Text Segmentation of Consumer Magazines in PDF Format

Jian Fan
Hewlett-Packard Laboratories
Palo Alto, California, USA
jian.fan@hp.com

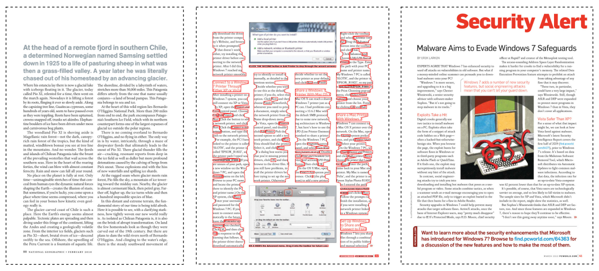
***Abstract*— Text segmentation is usually the first step taken towards the reuse and repurposing of PDF documents. Through experimental evaluation, we found that the leading text segmentation algorithms have limitations for contemporary consumer magazines. We propose a new local homogeneity measure based on line space, and incorporate this new feature into a region growing algorithm. Using a fixed set of parameters, our algorithm achieved robust performance on PDF magazines with wide-ranging layouts and styles.**

Keywords-page segmentation; text segmentation; PDF analysis

I. Introduction

Portable document format (PDF) accurately preserves the visual appearance of electronic documents across application software, hardware, and operating systems, making it a widely used format for document sharing and archiving. However, most PDF documents do not maintain logical structures of document content, such as text lines, paragraphs, titles, and captions. The lack of structural information makes it difficult to reuse and repurpose the digital content represented by a PDF document. Extracting logical structures from PDF documents has been an active research problem with many real applications [1][2][3].

Text segmentation is usually the first step taken towards logical structure extraction. The goal is to group low-level text entities into lines and homogeneous blocks. Most previous work targeted PDF documents of simple style and layout. Lovegrove and Brailsford grouped text lines only if they have the same font name, point size, and line space [1]. Chao and Fan required additional homogeneity regarding color [2]. Although the strict conditions on font name, size, and color may be valid for most technical documents, they are often not true for contemporary consumer magazines. Figure 1(L) is a page from National Geographic Magazine, Feb. 2010. The font size of the lead paragraph gradually changes, line by line. In addition, many such magazines freely use various color and font families to highlight URLs and other items. The strict homogeneity requirement may result in severe oversegmentation. Figure 1(C) shows the result using the method of [2]. Another limitation of the above cited papers is that they both assumed that a key grouping criterion, the line space, is a constant associated one-to-one with a particular font on a global (page) scale. In the context of determining reading order, Meunier proposed an optimized XY-cut for text segmentation [4]. However, as the author himself pointed out in the paper, XY-cut suffers from two significant drawbacks. First, it is very sensitive to two parameters specifying the minimal width/height of a cut, and their automatic determination is difficult. Second, XY-cut cannot handle the L-shaped text layouts that are common in consumer magazines. Figure 1 (R) is such an example.


At the head of a remote fjord in southern Chile, a determined Norwegian named Samsing settled down in 1925 to a life of pasturing sheep in what was then a grass-filled valley. A year later he was literally chased out of his homestead by an advancing glacier.

Security Alert

Malware Aims to Evade Windows 7 Safeguards

Want to learn more about the security enhancements that Microsoft has introduced for Windows 7? Browse to find.pcworld.com/64363 for a discussion of the new features and how to make the most of them.

Figure 1. Examples of consumer magazine pages and a text segmentation result using a previous work. Left: a page from National Geographic Magazine Feb. 2010. Center: text segmentation result using a strict homogeneity criterion. Right: a page from PC Magazine Mar. 2010.

To the best of the author's knowledge, the most complete treatment to date comes from recent publications by Bloechle [5][6]. Bloechle presented a bottom-up text grouping method with a relaxed set of homogeneity criteria and a follow-up top-down step to correct possible undersegmentation. However, this hybrid approach has some critical limitations that we will discuss later.

In this paper, we propose an improved algorithm. Our main contributions include a novel homogeneity measure based directly on line space and a bottom-up region growing approach utilizing both the line space and font size measures. The rest of the paper is organized as follows: Section II describes the details of the proposed method and discusses the main limitation of Bloechle's method. Section III presents some experimental results. Section IV concludes the paper.

II. Proposed Method

The goal of text segmentation is to group text into visually homogeneous blocks. In this paper, we limit ourselves to true PDF documents, in contrast to PDF files embedded with scanned document images. We assume that text can be separated from image and graphic components using existing PDF libraries. We further assume that text follows horizontal reading order and is laid out as strictly horizontal lines. We do not assume local consistency between rendering order and reading order [7]. Our method includes three stages: text info retrieval, the merging of words into text lines, and the grouping of text lines into text blocks.

For the convenience of subsequent description, we define as notation the *relative difference* of two non-negative values v_1 and v_2 as the following:

$$\Delta(v_1, v_2) = \begin{cases} 0, & \text{if } v_1 = 0 \text{ and } v_2 = 0 \\ \infty, & \text{if } (v_1 \cdot v_2) = 0 \text{ and } v_1 \neq v_2 \\ |v_1 - v_2| / \min(v_1, v_2), & \text{otherwise} \end{cases} \quad (1)$$

A. Preprocessing and text attribute retrieval

We relied on the Adobe PDF Library (from Datalogics) for rendering and retrieving text attributes. A given PDF page is opened and a WordFinder (`PDWordFinder`) is created. Words (`PDWord`) and quads[1] (`ASFixedQuad`) are then accessed via the WordFinder. Visual attributes that can be retrieved include font family, font size, color and bounding box. An example of bounding boxes of quads retrieved using `PDWordGetNthQuad` is shown in Figure 2. Notice that the height of the bounding boxes varies significantly within the paragraph and even within a single text line due to differences in fonts. It can be seen in Figure 2(R) that the vertical center positions in most cases fluctuate less in a center line than either the top or bottom position of the bounding boxes.

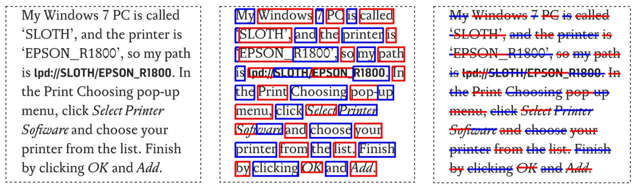

My Windows 7 PC is called 'SLOTH', and the printer is 'EPSON_R1800', so my path is lpd://SLOTH/EPSON_R1800. In the Print Choosing pop-up menu, click *Select Printer Software* and choose your printer from the list. Finish by clicking *OK* and *Add*.

Figure 2. Left: a paragraph from a PDF version of PC World March 2010 p. 72. Center: bounding boxes of text quads retrieved using PDF Library's WordFinder. Right: vertical center lines computed from bounding boxes.

B. Merging words/quads of text into line segments

The goal of this step is to merge words/quads into line segments. As it will become clear shortly, a line segment will not necessarily be a logical text line.

Since we do not assume that the rendering order is the same as the reading order, we must rely on the font size and spatial attributes. We start by sorting all quads in the order of top-down and left-to-right based on the vertical center position of the bounding boxes. Although it is not guaranteed that sorted order agrees with reading order, the sorting reduces the search range for neighboring quads.

The line-forming process proceeds by picking up a quad that has not been assigned a line id to start a new line segment. It then extends the line both left and right by adding *qualified* quads to the line. When no qualified quad can be added to the line, a new line is started until all quads are assigned a line id. The criteria we adopted for judging if two quads can be merged are similar to Bloechle's [5]:

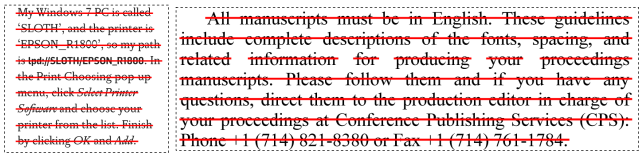
1. Overlap. The vertical overlap between two bounding boxes should be large enough such that $O(q_i, q_j) > k_o \cdot \min(h_i, h_j)$, where O is the vertical overlap, h is the height of a quad and k_o is a threshold.
2. Font size. The font size difference between the two quads should be small enough such that $\Delta(f_i, f_j) < k_{fh}$, where f is the font size and k_{fh} is a threshold.
3. Space. The space between the two quads should be small enough such that $d_{i,j} < k_{dq} \cdot \min(f_i, f_j)$, where f is the font size, $d_{i,j}$ is the horizontal distance between two quads and k_{dq} is a threshold.

We would like to point out that for text with horizontal reading order, text merging in the horizontal direction should be performed first. This is because we can safely merge two words horizontally if their horizontal distance is closer than a threshold. However, we may not merge two quads vertically even if their vertical distance is very close.

We use font size and vertical center as the attributes of a line segment. Taking possible text variations within a line segment into account, these two attributes are computed using weighted averaging. The width of quads is used as the weight:

$$f_L = \left(\sum_i f_i \cdot w_i\right) \Big/ \sum_i w_i \quad \text{and} \quad y_L = \left(\sum_i y_i \cdot w_i\right) \Big/ \sum_i w_i ,$$

where f_i, y_i and w_i are font size, vertical center and width of the quad i, respectively. Figure 3 shows two examples where red lines are the center lines of resulting line segments. Notice the fragmentation of a logical text line for the paragraph on the right side.


~~My Windows 7 PC is called 'SLOTH', and the printer is 'EPSON_R1800', so my path is lpd://SLOTH/EPSON_R1800. In the Print Choosing pop-up menu, click *Select Printer Software* and choose your printer from the list. Finish by clicking *OK* and *Add*.~~

~~All manuscripts must be in English. These guidelines include complete descriptions of the fonts, spacing, and related information for producing your proceedings manuscripts. Please follow them and if you have any questions, direct them to the production editor in charge of your proceedings at Conference Publishing Services (CPS): Phone +1 (714) 821-8380 or Fax +1 (714) 761-1784.~~

Figure 3. Left: line segments from the paragraph in Figure 2 (L). Right: a paragraph from the formating guidelines of IEEE CS.

C. Growing line segments into blocks

The goal of this step is to merge text segments into homogeneous text blocks. This will also enable us to regroup fragmented line segments into logical lines. Therefore this is the core of our text segmentation algorithm.

Here we first review the region growing method proposed by Bloechle ([5], section 5.3) . Starting from a line i, a block recursively takes in a new line j with the following conditions:

1. The horizontal overlap between the two lines is non-zero.

[1] The Acrobat and PDF Library API Reference defines a quad as "a quadrilateral bounding a contiguous piece of a word. Every word has at least one quad. A word has more than one quad, for example, if it is hyphenated and split across multiple lines or if the word is set on a curve rather than on a straight line."

2. The font sizes of the two lines i and j satisfies $|f_i - f_j| < \sigma_{precision} \cdot \min(f_i, f_j)$, where $\sigma_{precision}$ is an algorithm parameter set to 0.25.
3. The vertical distance (line space) $d_{i,j}$ between the two lines i and j satisfies $d_{i,j} < \sigma_{clustering} \cdot \min(f_i, f_j)$, where $\sigma_{clustering}$ is another algorithm parameter set to 0.8.

However, Bloechle's method suffers from a critical drawback stemming from the third criterion. Due to the tie between line space and font size, we found that it is impossible to have a parameter $\sigma_{clustering}$ that works on wide-ranging document layouts and styles. This can be demonstrated with just two PDF magazine pages shown in Figure 4. The minimum value of $\sigma_{clustering}$ needed to correctly merge the lines of the first paragraph in Figure 4(L) is 2.5. However, for the page of Figure 4(R), $\sigma_{clustering} \geq 1.9$ would result in a serious segmentation error. In this case, the caption paragraph (below the image) will be merged with the lower portions of the left and middle columns as well as all lines of the right column. The error of merging lines across columns cannot be detected by the subsequent interline change detection step. Therefore, there is not a single value of $\sigma_{clustering}$ good for both magazines.

Figure 4. Two PDF pages. Left: National Geographic Magazine Feb. 2010 p. 70. Right: Maclean's Sept. 20, 2010 p. 84.

This problem with Bloechle's method arises due to merging across block boundaries. Our method avoids the pitfall by decoupling line space and font size and carefully detecting block boundaries during region growing. We deployed two measures in detecting block boundary:

1. Line space. Based on the observation that change of line space alone usually indicates a block boundary, we define a measure of relative difference between the two line spaces as $\Delta(d_{i,j}, d_{i,h})$ (Eq. 1 and Figure 5), which is independent of font size. The line space is defined as the distance between two vertical center lines as illustrated in Figure 5. We can then detect block boundary by comparing the relative line space difference with a threshold k_{dl}: line i is a block boundary if $\Delta(d_{i,j}, d_{i,h}) > k_{dl}$. This measure fundamentally differs from Bloechle's.
2. Font size. We also define a relative difference of font sizes $\Delta(f_1, f_2)$ (Eq. 1). Line i is a block boundary if $\Delta(f_i, f_j) > k_{fl}$ or $\Delta(f_i, f_h) > k_{fl}$, where f is the weighted average of font sizes within the line i.

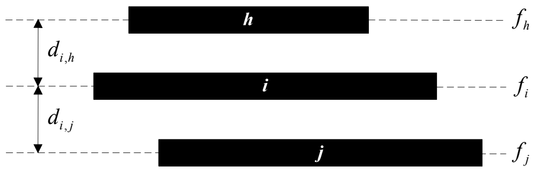
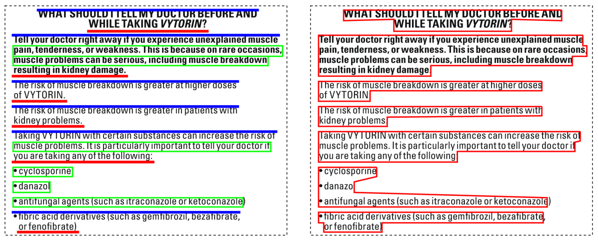


Figure 5. Line space and font size.

Using both line space and font size measures, we can detect not only the block boundary, but also the type of boundary as the following:

$$B_i = \begin{cases} 0, & \text{if}\left(\Delta(d_{i,j}, d_{i,h}) < k_{dl} \wedge \Delta(f_i, f_j) < k_{fl} \wedge \Delta(f_i, f_h) < k_{fl}\right) \\ 1, & \text{else if}\left(\hat{d}_{i,h} + w_f \cdot \Delta(f_i, f_h)\right) > \left(\hat{d}_{i,j} + w_f \cdot \Delta(f_i, f_j)\right) \\ -1, & \text{otherwise} \end{cases}$$

where B_i is a flag indicating whether line i is a boundary line and its type, w_f is a weight emphasizing either font size or line space, and $\hat{d}_{i,h}$ and $\hat{d}_{i,j}$ are normalized line spaces $d_{i,j}$ and $d_{h,i}$: $\hat{d}_{i,h} = d_{i,h} / \max(d_{i,h}, d_{i,j})$, $\hat{d}_{i,j} = d_{i,j} / \max(d_{i,h}, d_{i,j})$. Boundary type "1" means "top-down", or line i is closer to line j than to line h. On the other hand, boundary type "-1" means "bottom-up", or line i is closer to line h than to line j. We would like to point out that the above boundary detection method is analogous to edge detection in bitmap images. An example of boundary detection and the resulting segmentation is shown in Figure 6. In the image on the left, blue and red lines indicate "top-down" and "bottom-up" boundaries, respectively, while green boxes indicate non-boundary lines. In Figure 6(R), red polygons represent text blocks obtained from our line growing algorithm, to be presented next.


WHAT SHOULD I TELL MY DOCTOR BEFORE AND WHILE TAKING *VYTORIN*?

Tell your doctor right away if you experience unexplained muscle pain, tenderness, or weakness. This is because on rare occasions, muscle problems can be serious, including muscle breakdown resulting in kidney damage.

The risk of muscle breakdown is greater at higher doses of VYTORIN.

The risk of muscle breakdown is greater in patients with kidney problems.

Taking VYTORIN with certain substances can increase the risk of muscle problems. It is particularly important to tell your doctor if you are taking any of the following:

• cyclosporine

• danazol

• antifungal agents (such as itraconazole or ketoconazole)

• fibric acid derivatives (such as gemfibrozil, bezafibrate, or fenofibrate)

WHAT SHOULD I TELL MY DOCTOR BEFORE AND WHILE TAKING *VYTORIN*?

Tell your doctor right away if you experience unexplained muscle pain, tenderness, or weakness. This is because on rare occasions, muscle problems can be serious, including muscle breakdown resulting in kidney damage.

The risk of muscle breakdown is greater at higher doses of VYTORIN.

The risk of muscle breakdown is greater in patients with kidney problems.

Taking VYTORIN with certain substances can increase the risk of muscle problems. It is particularly important to tell your doctor if you are taking any of the following:

• cyclosporine

• danazol

• antifungal agents (such as itraconazole or ketoconazole)

• fibric acid derivatives (such as gemfibrozil, bezafibrate, or fenofibrate)

Figure 6. An example of boundary detection (left) and segmentation (right).

Upon the completion of boundary detection, text segmentation is accomplished using region growing in the vertical direction (both up and down). We consider two neighboring lines i and j with horizontal overlap and no other text between them. Whether the two lines should be merged can be determined according to three possible scenarios:

1. Neither line *i* nor line *j* is a boundary line ($B_i = 0$ and $B_j = 0$). Of course, line *i* and *j* should be merged.
2. Only one of two lines *i* and *j* is a block boundary. This includes four possible cases based on the relative position of the boundary line and the type of the boundary. Only in two cases should the two lines be merged: the top line is a boundary line of the "top-down" type, or the bottom line is a boundary line of the "bottom-up" type. For the other two possibilities, the two lines must not be merged.
3. Both lines *i* and line *j* are boundary lines. This also includes four cases since each boundary line can have two types. The two lines should be merged only if the top line is the "top-down" type and the bottom line is the "bottom-up" type. In this case, because the text block will only have two lines, we may impose a stricter condition on the maximum line space, linking it to font size to avoid merging two lines very far apart.

A concrete example of the above method is shown in Figure 6(R) using the boundary detection result in Figure 6(L). We would like to point out that the layout of the bullet items is an example where text with the same font does not have the same line space globally. In this case, bullet items have the same font. But space between bullet items differs with line space of text within a single item. Our method achieved the correct segmentation, grouping text that belongs to a single item without splitting them.

A c-style pseudo-code for the line segment grouping is included to the right.

```
int GroupLineSegToBlocks(LineSeg *lines, int nlines) {
    Sort lines in top-down and left-right based on the geometric center point;
    For each line segment, identify its vertical neighbors above and below, and save the result with each line segment. Note that vertical neighbor implies horizontal overlap.
    Detect boundary lines and their type.

    Initialize bid of all line segments to -1;
    int bid = 0;
    for(i=0;i<nlines;i++) {
        if( lines[i].bid>=0 )
            continue;

        RegionGrow(lines,nlines,i,bid);
        bid++;
    }
    return bid;
}

void RegionGrow (LineSeg *lines, int nlines, int seed,int bid) {
    Queue q;    // a FIFO quaeue
    q.enqueue(seed);
    lines[seed].bid = bid;

    while( q.isEmpty()==false ) {
        int i = q.dequeue();
        for ( each neighbor line j above and below line i ) {
            if( lines[j].bid>=0 )
                continue;
            merge = check if line j should be merged;
            if ( merge==true ) {
                lines[j].bid = bid;
                q.enqueue(j);
            }
        }
    }
}
```

III. Experimental Results

The proposed algorithm was thoroughly tested on three consumer magazines: National Geographic Magazine, February 2010; PC World Magazine, March 2010; Maclean's (Canada) September 20, 2010. The three multi-page PDF files were downloaded from the Internet. The algorithm parameters are listed in Table I.

TABLE I. ALGORITHM PARAMETERS

Parameter	Value	Description
k_{fh}	0.4	Maximum relative font size difference for horizontal merge
k_{dq}	0.6	Maximum space between horizontal words to merge
k_o	0.4	Minimum vertical overlap to merge two words horizontally
k_{fl}	0.25	Maximum relative font size difference for line merging
k_{dl}	0.2	Maximum relative line space difference for line merging
w_f	2.0	Weight for computing boundary orientation

It should be noted that we had to set the threshold k_{dq} very low (60% of font size) in order to accommodate the very narrow column space presented in the Maclean's pages, which utilize lines as column separators. The low threshold caused more text lines to be fragmented. Nevertheless, the proposed algorithm achieved very satisfactory results on all three magazines.

As precise quantitative evaluation for the three magazines requires ground truth, which is very time-consuming and still involves some subjective judgments, we only attempted a coarse and preliminary assessment. We manually counted content text blocks and captions and inspected the corresponding segmentation results. We did not count advertisement pages due to the difficulty of interpretation. Nor did we count titles, tables and maps. For example, we counted 3, 10, 7, 5, 6, and 4 text blocks respectively, for the six pages in Figure 7 starting from the top left in clockwise order. The segmentation performance based on counted text blocks as well as the processing time is presented in Table II below.

TABLE II. QUANTITATIVE EVALUATION

Magazine	Num. Pages	Num. Text Blocks	Num. Segmented Blocks	Processing Time *(sec.)
National Geo.	155	271	284	8.9
PC World	107	430	432	10.2
Maclean's	100	391	393	10.1

*PC Intel Core2 Duo 2.93 GHz

Text segmentation results using six pages from the three magazines are shown in Figure 7. Four of the original pages are shown in previous sections.

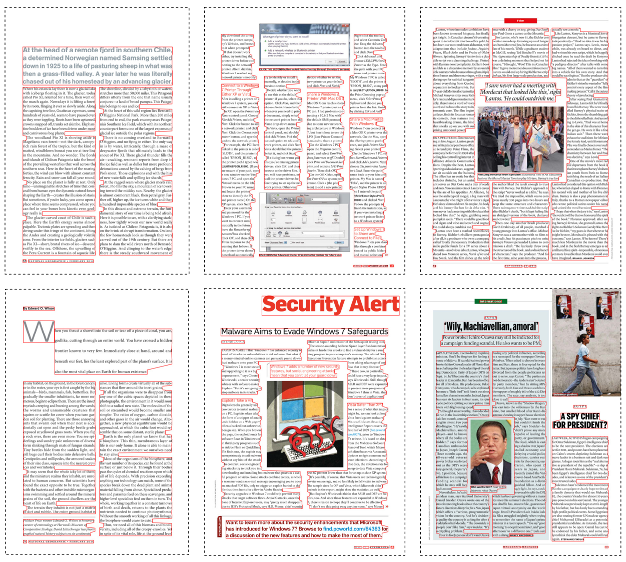


Figure 7. Selected text segmentation results. Left column: pages from National Geographic. Middle column: pages from PC World. Right column: pages from Maclean's.

In terms of counted text blocks, our method achieved over 99% accuracy. Only a very small number of errors were made. They may be detected and fixed in a post-processing step with some heuristics. However, as we expected, text segmentation based on purely visual properties has its limitations. We found a significant number of non-meaningful segmentations in titles, tables, lists and maps. Some examples of this are shown in Figure 8.

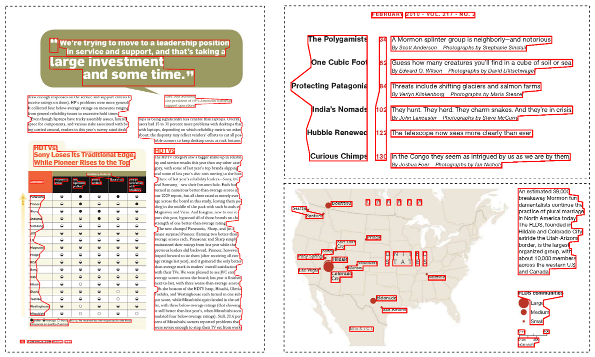


Figure 8. Examples of a title, table, list and map that require recognition and special handling.

IV. Conclusion and Future Work

We have presented a systematic method for text segmentation of PDF documents. The core of our algorithm lies in a novel measure of line space and a boundary detection method based on combined relative differences of font size and line space. Being localized in nature, our method overcomes some key limitations associated with global and top-down algorithms. We also demonstrated the robust performance of our proposed algorithm on three contemporary consumer magazines that contain complex layouts.

We also identified the limitations of our algorithm on titles, tables, lists and maps. Towards a complete PDF document understanding system, we plan to work on the following aspects:

1. Graphic recognition and integration with text segmentation. In many cases, graphic components such as lines and color background are used to separate text. The detection of graphic components and their integration with text segmentation will greatly improve performance.
2. List and table recognition.
3. Map recognition. Text belonging to map regions often has various orientations and excess character space. These are the most challenging cases for text segmentation.

Acknowledgment

The author would like to thank Dr. Hui Chao for providing her research codes and many helpful discussions.

References

[1] W. S. Lovegrove and D. F. Brailsford, "Document analysis of PDF files: methods, results and implications," Electronic Publishing, vol. 8(2 & 3), pp. 207–220, June & Sept. 1995

[2] Hui Chao and Jian Fan, "Layout and content extraction for PDF documents", S. Marinai and A. Dengel (Eds.): DAS 2004, LNCS 3163, pp. 213–224, 2004

[3] H. Déjean and J.-L. Meunier, "A system for converting PDF documents into structured XML format," H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 129 – 140, 2006

[4] J.-L. Meunier, "Optimized XY-Cut for determining a page reading order," Proc. ICDAR, pp. 347 - 351, vol. 1, 2005

[5] J. -L. Bloechle, "Physical and logical structure recognition of PDF documents", Doctoral thesis No. 1676, University of Fribourg (Switzerland), June 2010

[6] J. -L. Bloechle, D. Lalanne and R. Ingold, "OCD: An Optimized and Canonical Document Format," Proc. ICDAR, pp. 236 – 240, 2009

[7] J. Fang, Z. Tang, L. Gao, "Reflowing-driven paragraph recognition for electronic books in PDF," G. Agam, C. Viard-Gaudin (Eds.), Document Recognition and Retrieval XVIII, Proc. of SPIE-IS&T Electronic Imaging, vol. 7874, 2011