Optimized XY-Cut for Determining a Page Reading Order

Jean-Luc Meunier
Xerox Research Centre Europe
6, chemin de Maupertuis
F-38240 Meylan
jean-luc.meunier@xrce.xerox.com

Abstract

In this paper, we propose a fast method for determining the human reading order of the layout elements of a document page. The proposal includes a computationally tractable optimization approach to the problem. We also report on the performance of the method and discuss it, in light of related work.

1. Introduction

The last few decades have seen the proliferation of electronically created documents and the increase of their usage in human activities. Unfortunately, not all of these electronic documents allow for easy use beyond screen viewing or paper printing. Reasons for this restriction include, among others, the unavailability of the document in native format, the deprecation or disappearance of the original authoring environment, but also the case of scanned paper documents. Many organizations are therefore looking for methods and tools for converting those particular electronic documents to a structured format in order to reuse or repurpose them in a cost-effective way.

We are interested here in one of the multiple problems that the conversion of print-, or display-, oriented documents to a structured format poses, namely the determination of the human reading order of the elements on each page of the document. Indeed the order in which the information appear in a print- or display- oriented file format may have more to do with optimizing the print process or the file size, for instance, than reflecting the logical order of the contained information. The conversion from the Adobe's PDF format to an XML format reflecting the logical structure of the document is our particular interest here. We did not found any tool able to determine in a systematic way the proper reading order in order to provide a properly ordered text from a PDF file. For instance, depending on how the PDF was generated, the consecutive lines from two adjacent columns may get interleaved, or all titles may get grouped at the end of the flow of normal text. And the proper ordering is instrumental for further automatic processing of the document, like for the recognition of paragraphs, titles, lists, and of great importance by itself.

In this paper, we will first define better the ordering problem we consider and then introduce a first ordering method derived from a page image segmentation method named XY-cut [1]. The next section takes an optimization perspective to improve on the previously described approach. We then report on the method evaluation and performance before discussing the proposed method in view of the alternatives.

2. The Ordering Problem

The ordering problem consists in inducing an order between layout objects positioned on a page in order to reflect it human reading order. Multiple approaches to this problem have been proposed in the literature [6], exploiting geometric or typographic features of the page objects, or going further in exploiting the content of objects, with or without a priori knowledge for a particular document class. We look here for a robust method only exploiting geometric features, in order to potentially apply on various classes of documents in various languages. This approach is similar to some previous works [2] [3] and shares with them a geometric approach issued from the XY-cut page segmentation method. The use of of-the-shelf PDF converters leads to consider here layout objects of various granularities, because they may contain one line, or one word, or part of it, or even only one letter. The pages considered here often contain few hundred of textual objects, hence this particular method.

More generally, the determination of the human reading order of a page is particularly difficult because there are cases were the geometric features leave open multiple possible valid ordering. It is then only by exploiting other features, for instance the textual content itself, that the correct order can be determined. In this paper, we however deliberately chose to exploit geometric features only, leaving those ambiguous cases unsolved for now.

3. XY-Cut Background

3.1. XY-Cut for Segmentation of Page Image

At its origin, the XY-Cut [4] is a page segmentation method working on a page image. Briefly, the method consists in finding the widest empty rectangle, or valley (a threshold gives some robustness to image noise), entirely crossing the page either vertically or horizontally. The page is then segmented in two blocks, which are shrinked to fit closely their content. The method applies again recursively to each block. It stops when no large-enough valley can be found in any of the created blocks, which become the final image segments.

Jaekyu Ha et al. [2] proposed to implement it using the bounding boxes of connected components of black pixels instead of using image pixels. After the connected components were obtained, the recursive XY-cut became much faster to compute. This applies well here since PDF converters provide bounding boxes. Also the XY-cut can exploit the graphical lines that often are used to visually separate different parts on a page, by both enforcing cuts along lines and discarding cut crossing those lines. PDF converters output those lines in a SVG format that is easy to exploit.

Figure 1 below illustrates the XY-cut method for the segmentation of a page image.

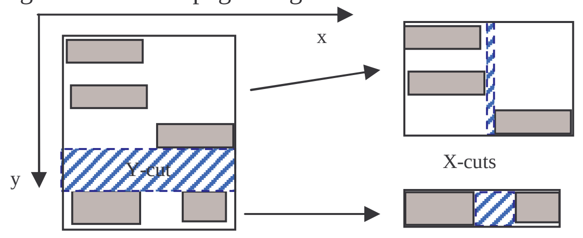

Figure 1: the XY-cut.

Gray rectangles represent some bounding boxes. The hatched rectangles indicate the largest valleys where to cut. The left sides shows the Horizontal cutting of the root block corresponding to the whole page image. On the right, the two formed blocks are then Vertical cut. The final blocks, or segments, will coincide with the gray rectangles.

3.2. XY-Cut for Text Ordering

Ishitani [3] proposed to leverage the hierarchy of blocks generated by the XY-cut algorithm to induce an order among blocks, in particular among leaf blocks. Typically, for a western conventional top-bottom left-to-right reading order, ordering the block hierarchy of block in a top-bottom left-to-right way leads to the expected order, as illustrated by figure 2 below.

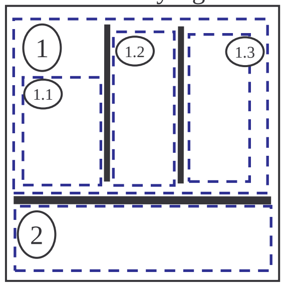

Figure 2: Ordering the Blocks

Inside blocks, a local order can be induced by looking for horizontal cuts, allowing thin valleys or even some overlapping. This is a way of forming lines in an elegant manner because with the same cutting model. Finally, the page ordering defines the complete document ordering.

3.3. Problems of the XY-Cut Ordering

There are three main problems with ordering the text of a page with the XY-cut algorithm: the "L-shapes", the threshold parameters and the cutting strategy.

Firstly, certain configurations, often called "L shapes", prevent the algorithm from segmenting properly the page. Two such cases appear below.

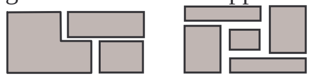

Figure 3: L-shapes defeating the XY-cut.

While those L-shapes occur rarely in technical documents, their frequency increases in newspapers where the page layout becomes more complex. Some workarounds seem to exist, consisting in discarding temporarily some layout objects inserting them later on. We however do not touch upon this problem in more depth here, since we mainly target technical documents.

Secondly, there is the need to define the minimum required widths of the horizontal and vertical cuts, or valleys. Those parameters directly influence the quality of segmentation, i.e. in paragraph, and are difficult to determine, in particular the nominal line spacing that may vary across pages or within pages. Segmenting is not the purpose but unfortunately some ordering methods, like [3], are sensitive to it because of their bottom-up ordering heuristic, in our opinion. We see this as a non-negligible problem.

Thirdly, the cutting strategy is critical with respect to the obtained order. Indeed, at each recursion step, there are often multiple possible, and possibly conflicting, cuts. In the original algorithm, the widest cut is selected at each recursion. This strategy fits for a page segmentation tasks but not likely for an ordering task. In two columns documents, like this one, an error can consist then in horizontally cutting a page before cutting vertically along the column separation.

This is why Ishitani [3] devised a bottom up approach using three heuristics that take into account local geometric features, text orientation and distance among vertically adjacent layout objects in order to merge some layout objects a priori. This aims at reducing the probability of having to face multiple cutting alternatives, but it does not truly prevent them from occurring. In addition, given the fine grain textual objects we inherit from the PDF document, this approach does not fit our needs, unless paragraphs would already be defined, which goes back to the second problems listed in this section.

In conclusion, we are interested here in devising a strategy that is both fully deterministic and efficient in presence of numerous fine-grain layout objects. We propose to view this problem as an optimization problem, i.e. what is the best series of cuts in view of some optimization criterion.

4. An Optimization Perspective

The control of the sequence of creations of blocks in the hierarchy is key to reflect the human reading order. From an optimization perspective, the problem of cutting a block can be seen as selecting a series of cuts in order to maximize a score function. In this section, we propose both a score function, which leads to the appropriate block ordering, and a method for finding the optimum in a computationally tractable way.

We assume a general model of document, where a document is made of pages, which contain rectangular layout objects geometrically positioned on the page with some coordinates (x, y, width, height).

Following the spirit of Ishitani's heuristics [3], the chosen cutting strategy and score function favor a reading by columns, since this is the conventional reading in most technical documents. But rather than relying on a bottom-up approach based on local geometric feature [3], the method consists in considering a block globally and in selecting among all possible cuttings the one that leads to the best set of columns. We therefore propose a cutting strategy that essentially aims at favoring vertical cuts against horizontals cuts. Figure 4 below illustrates how a horizontal cut can give precedence to a vertical cut to favor a column reading. The score function will best reward the largest cumulative height of created columns. Let us point out that it is not equivalent to simply choosing the longest possible vertical cut in the block since this choice may not be the optimal one, unless this cut does cross entirely the block, in which case both are equivalent.

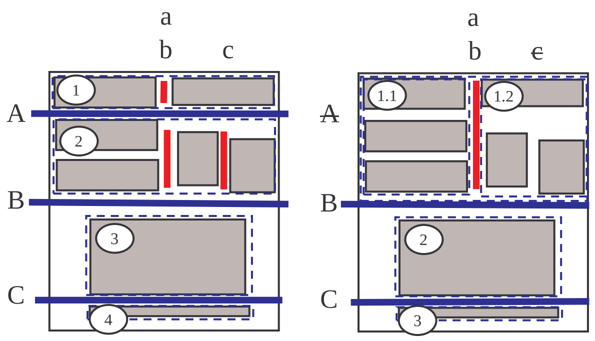

Figure 4: Optimized XY-cut

Figure 4 shows on the left hand three possible horizontal cuts (A. B, C) defining four possible blocks (1,2,3,4). Within each block, the possible vertical cuts are shown in red (a,b,c). Seven choices are offered here: taking only one of the horizontal cuts, two of them or all of them. The right hand part shows the preferred cuts (B,C to make a+b possible for first block).

In more details, the block cutting process becomes:

1. Given a block to segment, the method enumerates all possible horizontal cuts; (*A*, *B* and *C* on previous figure)
2. For each block potentially created by an enumerated horizontal cut, the method enumerates all possible vertical cuts inside it; (*(a) (b,c) ()*)
3. A (possibly empty) set of horizontal cut is chosen so as to make possible the best possible series of enumerated vertical cuts, given the score function.
4. The selected horizontal cuts are performed, and then for each created blocks the set of associated vertical cuts is performed as well.

5. Back to step 1 for each created blocks until no cut is anymore possible

The score function favors vertical cuts that span over multiple consecutive blocks. In practice, the score function is the sum of the height of the blocks that can be merged (1 and 2 in the left side of the example in figure 4). In addition, if a block can be merged with either the previous or the next one (because it can share a vertical cut with previous or with next block, but the two are not compatible, i.e. not vertically aligned), it is desirable to determine which vertical cut to favor. We therefore include in the score the inverse of the distance between merged blocks (the merge of blocks closer to each other is favored when the choice is left open).

The score function does not reward for the creation of too narrow columns. A parameter defines the minimal ratio of the column width to the page width, in practice one fifth of the page width in our tests. Otherwise columns of bullets could for instance appear.

The solving by enumeration of all possible combination of horizontal and vertical cuts is easy to express as a recursive algorithm but it can be so processing-intensive that it is not practically usable. It is however possible to exploit dynamic programming [5] techniques to greatly reduce the computation complexity. Briefly, dynamic programming consists in decomposing the main optimization problem into sub-problems that appear several times during the problem resolution. A simple memorization (called memoization in this context) of the sub-problem results drastically speeds up the whole resolution.

Dynamic programming applies here by recursively solving the following sub-problem: find best choice for the block number i (e.g. A, B, C) while complying with the given set of previous vertical cuts. So the sub-problems consists in choosing between sharing a vertical cut with previous block or rather stop any previous vertical cut by horizontally cutting before the current block. Let's note BC(i, X) the function that solve this sub-problem. BC(i, X) can be computed by computing the maximum among BC(i+1, X') and BC(i+1, ∅). X' is the intersection of the set of vertical cuts X with the set of possible vertical cuts in block i. ∅ denotes that no vertical cut is imposed to block i+1 (i.e. an horizontal cut separates block i and block i+1).

Here is the pseudo-code for computing the score function associated with BC (we do not show here how to also return the solution details, for sake of simplicity):

```
Function BC(i, X):
  if alreadyComputed(i, X):
        return memoization(i, X)
  #possible X-cuts in block i alone
  Xi = XCut(i)
  if X = ∅:
    #we are not required to share an X-cut
    # with previous block.
    #either we share the local Xi
    # vertical cut with next block
    sc1 = BC(i+1, Xi)
    if sc1 != 0:
      #if the sharing can be pursued,
      # reward for height of block i
      sc1 = sc1 + height(i)
    #or we horizontally cut between
    # block i and block i+1
    sc2 = BC(i+1, ∅)
    score = max( sc1, sc2 )
  else: #we are required to share a
        # vertical cut with previous block(s)
    X' = intersect(X, Xi)
    If X' = ∅ or tooNarrow(X'):
      score = 0
    else: #reward the making of columns
      share_score = 1/dist(i-1, i) + height(i)

      #pursue the sharing or not?
      sc1 = BC(i+1,X')#if pursuing the sharing
      sc2 = BC(i+1, ∅)#score if not pursuing

      score = share_score + max(sc1, sc2)
  memoize(i, X, score)
  return score
```

5. Evaluation

Our primary interest was to convert technical PDF documents into XML documents whose structure reflects the logical structure of the document in terms of sections, title, paragraphs, etc. We therefore evaluated the method on a series of PDF documents, most of them being technical documents in particular usage or repair manuals. After conversion to XML with an of-the-shelf converter, the page headers and footers were discarded thanks to a component that is not reported here.

The method is fast (below one second per page). Without the dynamic programming implementation, the processing of one page may take hours or days for complex pages.

A quantitative measure shows that 98% of the blocks are in proper order (the number of errors corresponds to an edit distance between the reference and test sets of lines).

Thanks to a SVG visualization tool, Figure 5 below, it was possible to validate manually many of the documents, totalizing 800 pages. The results are clearly satisfactory.

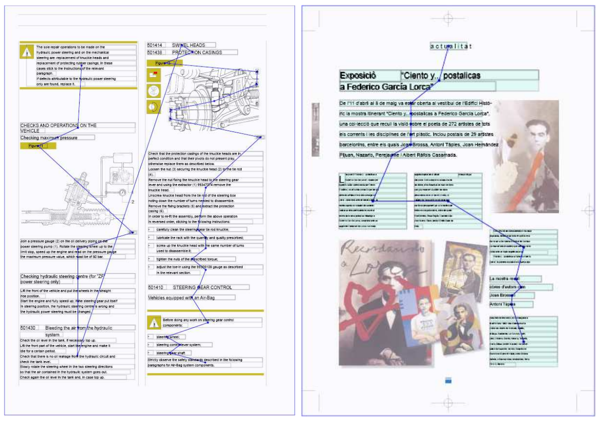

Figure 5: Visualization of the Block Order

Figure 5 shows the bounding box of the textual elements. Images were ignored in this case, but could be handled as well by their bounding box. The blocks formed by the method are also materialized by bounding boxes and their center is shown by a dot. The line links consecutive blocks.

Another set of measures was done by collecting a variety of MS-Word documents on the Internet. Each document was saved as text using MS-Word to generate a reference ordered text flow, and in parallel converted to PDF, then XML, then reordered and finally converted to text. This measure also gave a similar success rate and led to the conclusion that the present method was effective, and of general use (i.e. not specialized for the first test collection).

The method showed robustness to non-uniform layouts, like the page above on the right, with a varying number of columns and line space. Actually, because of the global approach to cut a block, the sensitivity to the minimal horizontal cut width (i.e. line space) is relaxed. Setting its value so low that the page is segmented into lines does not prevent from cutting the correct columns. This is an advantage when handling documents with a varying line space. The additional parameter that defines the minimal width for a column does not need a fine tuning, since tolerating columns with one fifth of the page width did not lead to the creation of erroneous columns. Overall, setting correct parameter values is eased, compared to other XY-cut based methods.

6. Conclusion

We have described a geometric method for ordering the layout objects positioned on a page. This method grounds on the XY-cut method, initially designed to segment page images.

This work shares with previous works, Ha et al. [2] and Ishitani [3], the difficulty in handling the so-called “L-shape”, weakness inherent to the XY-cut approach and we do not offer here better workaround to it. Similarly, we acknowledge the limitations inherent to a pure geometric approach. There will be geometrically ambiguous pages that will defeat the method. We conjecture that those pages require to dig into textual content and other features for a better processing, at the possible price of a less general applicability, in term of document class or language. However, we experimentally observed that the presence of such ambiguous page remains exceptional in business and technical documents that do not pertain to graphic arts. They also often incorporate separator lines to help the human reader and those lines also conduct to the appropriate cutting and hence order.

We believe to improve on previous work in three important manners. Firstly, the proposed method is less sensible to the proper setting of the XY-cut parameters. Secondly, the optimization approach allows us to split a block into sub-blocks in an optimal manner, while encapsulating the guiding heuristic in a score function. Under the same approach, this function may eventually include additional relevant features, non-geometric ones in particular. Thirdly, to take into account the relatively large number of layout object we extract from PDF, we devised a dynamic programming implementation that makes the optimization approach computationally tractable.

7. References

[1] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," International conference on Pattern Recognition, 1984

[2] Jaekyu Ha, R.M. Haralick, I.T. Phillips, Recursive X-Y cut using bounding boxes of connected components, International Conference on Document Analysis and Recognition, ICDAR 1995

[3] Yasuto Ishitani, Document Transformation System from Papers to XML Data Based on Pivot XML Document Method, International conference on document analysis and recognition, ICDAR 2003

[4] A. K. Jain, M. N. Myrthy, and P. J. Flynn. Data clustering: A survey. ACM Computing Survey, 31(3):264--323, 1999.

[5] Dynamic Programming, http://en.wikipedia.org/wiki/Dynamic_programming

[6] R. Cattoni, T. Coianiz, S. Messelodi, C.M. Modena: Geometric Layout Analysis Techniques for Document Image Understanding: a Review, ITC-IRST Technical Report #9703-09