A Distance-based Technique for non-Manhattan Layout Analysis

Stefano Ferilli Marenglen Biba Floriana Esposito
Teresa M.A. Basile
Università degli Studi di Bari-Dipartimento di Informatica
Via Orabona, 4 - 70126 Bari - Italy
{ferilli, biba, esposito, basile}@di.uniba.it

Abstract

Layout analysis is a fundamental step in automatic document processing. Many different techniques have been proposed in literature to perform this task. These are broadly divided in two main categories according to the approach they follow: the top-down methods start by identifying the high level components of the page structure and then recursively split them until basic blocks are found. On the other hand, bottom-up approaches start with the smallest elements (e.g., the pixels in case of digitized document) and then recursively merge them into higher level components. A first limitation of such methods is that most of them are designed to deal only with digitized documents and hence are not applicable to the general case of native digital documents which are mainly diffused. Furthermore, top-down and most of bottom-up methods are able to process Manhattan layout documents only. In this work, we propose a general bottom-up strategy to tackle the layout analysis of (possibly) non-Manhattan documents, and two specializations of it to handle the different cases of bitmaps and PS/PDF sources.

1 Introduction

Automatic document processing is a hot topic in the current computer science landscape [8], since the enormous and ever-increasing amount of available documents cannot be tackled by manual expert work. Document layout analysis is a fundamental step in the document processing workflow, that aims at identifying the relevant component in the document (often called *frames*), that deserve further and specialized processing. Thus, the quality of the layout analysis task outcome can determine the quality and even the feasibility of the whole document processing activity. Document analysis is carried out on (a representation of) the source document, that can be provided either in the form of a digitized document, and hence as a bitmap, or in the form of a native digital document (such as a word processing file, a PostScript or PDF file, etc.), and hence in the form of basic components (which we call *blocks*). The two cases are very different as regards the abstraction level of the input, and hence are typically faced by different algorithms.

Layout analysis algorithms are generally divided into two main categories: bottom-up ones, that start from elementary components in the document description and progressively group them into higher-level components (see [7, 11, 15]), and top-down ones, that start from the entire document page and split it into high-level components (see [1, 9, 12]). The latter are typically more efficient, but also less effective in handling documents with uncommon layout. Indeed, many algorithms assume that the document under processing has a so-called Manhattan layout, in which significant content blocks are surrounded by perpendicular background areas. This assumption holds for many typeset documents, and significantly simplifies the processing activity, but cannot be assumed in general. Previous work has agreed on identifying in bottom-up techniques the best candidates for handling the non-Manhattan case, since basic components grouping can ignore high level regularities in identifying significant aggregates. This work proposes a general bottom-up strategy to tackle the layout analysis of (possibly) non-Manhattan documents, and two specializations of it to handle the different cases of bitmaps and PS/PDF sources.

Well-known top-down strategies for Manhattan layout analysis are those proposed in [3], that finds the document background as a composition of progressively smaller background rectangles, in [10], that progressively divides the document layout according to alternate horizontal and vertical splits between disjoint components, and the Run-Length Smoothing Algorithm (RLSA) [16], that identifies runs of white pixel in the document image and fills them with black pixels whenever they are shorter than a given threshold. In particular, the RLSA works in four steps:

1. horizontal smoothing, carried out row by row on the image with threshold t_h;

2. vertical smoothing, carried out column by column on the image with threshold t_v;

3. logical AND of the images obtained in steps 1 and 2 (the outcome has a black pixel in positions in which both input images have one, or a white pixel otherwise);

4. new horizontal smoothing with threshold t_a on the image obtained in step 3, to fill white runs inside the discovered blocks.

2 Distance-based Technique

Much work in the literature is based on RLSA, exploiting it or trying to improve its performance by properly settings its t_h, t_v and t_a parameters [6] or by modifying it [4, 14]. Other works believe that the bottom-up strategy can be more effective, and propose algorithms to improve its efficiency by limiting the number of block merges, such as in CLiDE [13], inspired to well-known greedy algorithms such as the minimum spanning tree by Kruskal. Other works completely change the document representation, such as in the DOCSTRUM [11], where the distance between connected components is exploited as a basic feature and studied to identify, by means of a connection strategy based on the k-Nearest Neighbour technique, higher level layout components in the document.

Here, we propose a novel approach that borrows ideas from these works and add new ones to obtain an original result. It can be generally defined as bottom-up and distance-based, in that it exploits the distance among adjacent basic components of the page (pixels in the case of scanned images, basic blocks in the case of digital documents) to properly group them into consistent frames. In the case of scanned images, the white run lenghts embody such a distance, while in the case of digital documents the distance between components proposed in [13] is exploited. Indeed, horizontal/vertical white runs connect horizontally/vertically adjacent pixels, while the distance in CLiDE refers to horizontally/vertically adjacent blocks where, differently from pixels, a block may have many adjacent blocks on the same side, according to their projections as explained in [13]. Differently from DOCSTRUM, in our approach we do not impose a limit on the number of neighbours to be considered (indeed, in our case the blocks are not limited to single characters, but can include a variable number of characters, which would affect the k-Nearest Neighbour correct behaviour), rather we impose horizontal/vertical distance thresholds under which adjacent components are to be merged in the same frame. Hence, our technique could be assimilated to a single-link clustering procedure [2], where however the number of desired clusters is not fixed in advance, but automatically derives from the number of components whose distance falls below the given thresholds.

2.1 Application to scanned images

In the case of digitized images of documents, we adopt a simplified version of the RLSA, that we call RLSO (Run-Lengh Smoothing with OR), and that works as follows:

1. horizontal smoothing, carried out row by row on the image with threshold t_h;

2. vertical smoothing, carried out column by column on the image with threshold t_v;

3. logical OR of the images obtained in steps 1 and 2 (the outcome has a black pixel in positions in at least one of the input images have one, or a white pixel otherwise).

Now, each connected component is considered a frame: in turn all such components are considered, all the others are filtered out and a logical AND with the original image returns the frame content, to be further processed (e.g., by passing it to an OCR module if it is text, or to an image processing subsystem if it is a graphical component). Since the final outcome is obtained as a disjunction of the two smoothed images, this algorithm provides some advantages over the original RLSA: first, no final horizontal smoothing is required, since the OR operation, differently from the AND, does not lose anything from the original smoothed images; second, since the aim here is linking together small original connected components (such as characters) into larger ones (such as frames), shorter thresholds are sufficient (to fill inter-character, inter-word or inter-line spaces), and hence less runs will be filled and the algorithm will be more efficient; third, instead of performing the pure OR of the two smoothed images, efficiency can be improved by avoiding the third step and applying vertical smoothing directly on the horizontally smoothed image rather than the original one (this will not significantly affect the outcome, since it should not introduce further connections among components, and should even improve the quality of the result, since possible cases in which many rows have inter-word spacings vertically aligned, and hence not captured by the pure vertical smoothing, should be avoided). By performing logical OR, a single filled run is sufficient to link together different components, which is in general an advantage but must be managed carefully in order to avoid problems in cases in which logically different components are very close to each other and could be merged together.

Figure 1 shows the partial steps of processing on a scanned document image: (a) is the original document, (b) is the result of RLSO, (c) is as (b) but uses different gray levels to highlight different connected components, (d) (e)

and (f) are the texts extracted in turn from each connected component.

2.2 Application to natively digital documents

In the case of PS/PDF images, basic blocks in the source document (often corresponding to single characters, fragments of words, halftone images or simple geometrical elements such as rectangles and lines) can be regarded as playing the role of black runs in digitized document images, and the distance between adjacent components as the white runs. Thus, we can transpose the RLSO in this case by progressively merging into higher-level components adjacent basic components whose (horizontal or vertical) distance is below given thresholds. The algorithm works as follows:

1. build a frame for each basic block, such that the corresponding basic block is the only element of the frame;
2. compute the list H of all possible triples $(d_h, b1, b2)$ where $b1$ and $b2$ are horizontally adjacent basic blocks of the document and d_h is the horizontal distance between them;
3. sort H by increasing distance;
4. while H is not empty and the first element of the list has distance below a given horizontal threshold t_h
   (a) merge in a single frame the frames to which the two basic blocks in the first element belong
   (b) remove the first element from H
5. compute the list V of all possible triples $(d_v, b1, b2)$ where $b1$ and $b2$ are vertically adjacent basic blocks of the document and d_v is the vertical distance between them;
6. sort V by increasing distance;
7. while V is not empty and the first element of the list has distance below a given vertical threshold t_v
   (a) merge in a single frame the frames to which the two basic blocks in the first element belong
   (b) remove the first element from V

Steps 2-4 correspond to horizontal smoothing in RLSO, while steps 5-7 correspond to vertical smoothing. Efficiency of the procedure can be improved by preliminarily organizing the basic components into a structure where they are stored top-down in rows and left-to-right inside these rows, so that considering in turn each of them from top to bottom and from left to right is sufficient for identifying all couples of adjacent blocks: the horizontally adjacent block is the nearest subsequent block on the same row whose initial coordinate is greater than the final coordinate of the block being considered, according to the distance in [13], while the vertically adjacent blocks are the blocks in the nearest subsequent row (according to a distance similar to the previous one but computed vertically) that have a non-empty intersection in their vertical projection, defined again in [13] (as soon as the first non-overlapping block having initial coordinate greater than the final coordinate of the block being considered is encountered in each row, the rest of the row can be ignored).

This method resembles that proposed in [13], with the difference that no graph is built in this case, and more efficiency is obtained by considering adjacent components only. The list-of-rows structure allows to efficiently find adjacent blocks by limiting the number of useless comparisons. The "nearest first" grouping is mid-way between the minimum spanning tree technique of [13] and a single-link clustering technique. Compared to the ideas in [11], we exploit the distance between component borders rather than between component centers. Indeed, the latter option, adopted in DOCSTRUM, cannot be exploited in our case since the starting basic components can range from single characters to (fragments of) words, and this would cause a lack in regularity that would affect the proper nearest neighbour identification. After obtaining the final frames, each of them can be handled separately by reconstructing the proper top-down left-to-right ordering of the basic blocks it includes.

Figure 2 shows a digital document and the corresponding output of the proposed algorithm, with the discovered frames highlighted.

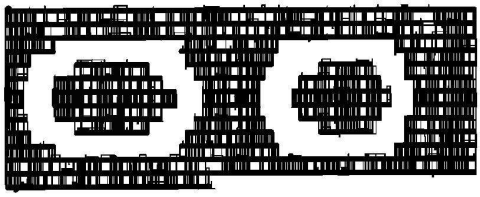
3 Experiments

Through experimental evaluation we wanted to check whether the proposed method can separate internal frames without merging them with the surrounding text. In order to answer such a question, we performed experiments on a collection of 100 natively digital documents whose layout included at least one 'internal' frame (i.e., a frame belonging to the non-Manhattan case). Specifically, 28 documents have two internal frames to be found (as in Figure 2), 71 documents have one internal frame and one document has three such frames. The 1-page documents were extracted from digital journals and magazines with the goal of buiding a testbed dataset with very different layout structures. All layouts show different structural characteristics in terms of: position and size (height and width) of internal text blocks; font size for the internal blocks and the surrounding text; distance between the internal text and the other part of the document; other elements in the page such as lines, figures, etc; titles (relative font and position) and paragraphs. The t_h and t_v thresholds were set to the average horizontal and

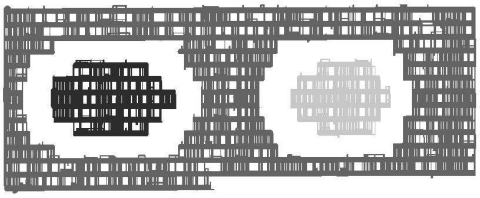
(a)

Questa è una prova di layout non-manhattan che il sistema automatico dovrebbe riconoscere separando i frame distinti anche se non formano dei quadrati. Da qui in poi il testo è tagliato e incollato a caso da altri documenti. Questa è una prova di layout non-manhattan che il sistema automatico dovrebbe riconoscere prova separando i frame distinti anche se prova non formano dei quadrati. Da qui prova in poi il testo è tagliato e prova incollato a caso da altri prova non non documenti. Non Questa è una prova di non layout prova non-manhattan che il prova sistema Non Questa è una prova di non layout prova non-manhattan che il prova sistema poi il testo è tagliato e prova incollato a caso da altri prova non non documenti. frame distinti anche se prova non formano dei quadrati. Da qui prova in automatico dovrebbe riconoscere prova separando i formano dei quadrati. Da qui in poi il testo è tagliato e incollato a caso da altri documenti. Questa è una prova di layout non-manhattan che il sistema automatico dovrebbe riconoscere separando i tagliato e incollato a caso da altri documenti.

(b)



(c)



(d)

Questa è una prova di layout non-manhattan che il sistema automatico dovrebbe riconoscere separando i frame distinti anche se non formano dei quadrati. Da qui in poi il testo è tagliato e incollato a caso da altri documenti. Questa è una prova di layout non-manhattan che il sistema automatico dovrebbe riconoscere prova separando i frame prova non formano Da qui prova in poi il incollato a caso non documenti. Non layout prova prova sistema Non layout prova prova sistema poi il incollato a caso non documenti. frame prova non formano Da qui prova in automatico dovrebbe riconoscere prova separando i formano dei quadrati. Da qui in poi il testo è tagliato e incollato a caso da altri documenti. Questa è una prova di layout non-manhattan che il sistema automatico dovrebbe riconoscere separando i tagliato e incollato a caso da altri documenti.

(e)

dei quadrati.
da altri prova non
non-manhattan che il
non-manhattan che il
da altri prova non
dei quadrati.

(f)

distinti anche se
testo è tagliato e prova
Questa è una prova di non
Questa è una prova di non
testo è tagliato e prova
distinti anche se

Figure 1. Processing steps on scanned documents

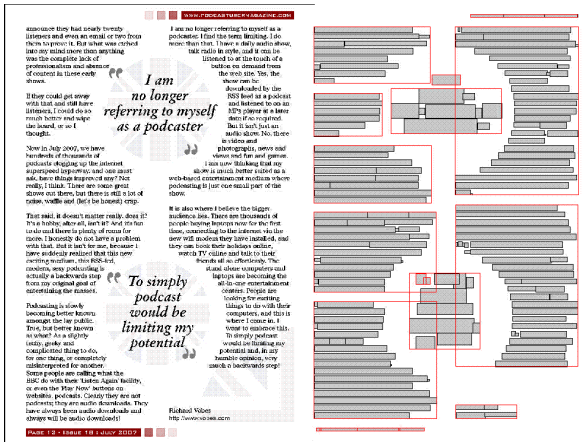


Figure 2. Processing steps on natively digital documents

vertical distance between adjacent blocks, respectively, and an additional check was included that avoided to merge text blocks having font size difference greater than 1.5 (empirically set). The proposed method successfully processed the document with three internal frames, and produced the following errors on the other documents: on 4 documents out of the 28 with two internal frames it was able to properly separate only one of the two internal frames, while on 6 of the 71 documents with only one internal frame, it was not able to properly separate the internal frame from the rest of the text. Hence, it showed an overall accuracy of 90% on the documents and of 92.3% on the internal frames to be found. Thus, the proposed method is in general able to group the internal text blocks on a very heterogeneous collection of layout configurations. An analysis revealed that at least 5 of the 10 faulty cases concern text blocks having larger fonts, and hence inter-word and inter-line spacings, than the rest of the text, that exceed the average thresholds.

4 Discussion

The overall layout analysis outcome can be improved by coupling this basic procedure with other ones aimed at handling peculiar features of some documents, such as the presence of horizontal or vertical lines, or the co-existence of different text sizes (and hence spacings) within the same document. These features are explicitly represented in natively digital documents, and hence can be straightforwardly exploited and considered as prioritary; in the case of digital documents, they are implicit and require proper pre-processing to be derived. As to lines, it is possible to avoid merging two frames if the nearest blocks considered in a step are interleaved by a line. This approach has been taken also in other works (e.g., for scanned images, in

[5] for the X-Y tree technique). As to the case of multiple text sizes/spacings, a technique could be developed to identify different horizontal/vertical thresholds for different portions of the document having omogeneous text size/spacing, and selectively applying them on the corresponding portion only.

The proposed strategy specifically focusses on identification of frames in non-Manhattan layouts. However, usually documents do show a Manhattan layout, and non-Manhattan behaviour is limited to particular document areas/components. For this reason, in order to improve efficiency, a hybrid approach can be exploited: first, basic components that can be easily grouped bottom-up are handled (e.g., in the case of PS/PDF documents, overlapping/adjacent basic blocks could be immediately grouped into words); then, a top-down approach could be exploited to quickly identify Manhattan zones in the page (e.g., exploiting RLSA on scanned documents or Breuel's technique on digital documents); lastly, the technique proposed in this paper can be applied separately to each Manhattan zone (possibly selected by heuristics that indicate non-Manhattan behaviour therein) to refine the outcome.

Further improvements suggested by the experimental outcomes concern separate handling of text blocks having different local spacings, which can be identified by a proper analysis of the spacings distributions in the page layout.

5 Conclusion

Document layout analysis is a fundamental step in the document processing workflow. The layout analysis algorithms presented in literature are generally divided into two main categories, bottom-up and top-down according to their approach to the problem, and they have been mainly developed to deal with digitized documents. However, document analysis has to be carried out on both digitized document and native digital document. Furthermore, most of them are able to process only Manhattan layout documents. Thus, the existing algorithms showed a limitation in deal some situations. This paper presented a general bottom-up strategy to tackle the layout analysis of (possibly) non-Manhattan documents, and two specializations of it to handle the different cases of digitized and native digital documents. Experiments on a real-world dataset of digital documents have been presented and discussed, that confirm the validity of the proposed approaches and suggest directions for further improvements.

References

[1] H. Baird, S. Jones, and S. Fortune. Image segmentation by shape-directed covers. In *Proceedings of International Conference on Pattern Recognition*, pages 820–825, Atlantic City, NJ, 1990.

[2] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[3] T. M. Breuel. Two geometric algorithms for layout analysis. In D. P. Lopresti, J. Hu, and R. S. Kashi, editors, *Document Analysis Systems*, volume 2423 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2002.

[4] H. Cao, R. Prasad, P. Natarajan, and E. MacRostie. Robust page segmentation based on smearing and error correction unifying top-down and bottom-up approaches. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 1*, pages 392–396, Washington, DC, USA, 2007. IEEE Computer Society.

[5] F. Cesarini, S. Marinai, G. Soda, and M. Gori. Structured document segmentation and representation by the modified x-y tree. In *ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition*, page 563, Washington, DC, USA, 1999. IEEE Computer Society.

[6] B. Gatos, S. L. Mantzaris, S. J. Perantonis, and A. Tsigris. Automatic page analysis for the creation of a digital library from newspaper archives. *International Journal on Digital Libraries (IJODL*, 3:77–84, 2000.

[7] K. Kise, A. Sato, and M. Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision Image Understanding*, 70(3):370–382, 1998.

[8] G. Nagy. Twenty years of document image analysis in pami. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):38–62, 2000.

[9] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.

[10] G. Nagy and S. C. Seth. Hierarchical representation of optically scanned documents. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 347–349, 1984.

[11] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, 1993.

[12] T. Pavlidis and J. Zhou. Page segmentation by white streams. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 945–953, Saint-Malo, France, 1991.

[13] A. Simon, J.-C. Pret, and A. P. Johnson. A fast algorithm for bottom-up document layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):273–277, 1997.

[14] H.-M. Sun. Page segmentation for manhattan and non-manhattan layout documents via selective CRLA. In *Eighth International Conference on Document Analysis and Recognition (ICDAR 2005)*, pages 116–120. IEEE Computer Society, 2005.

[15] F. Wahl, K. Wong, and R. Casey. Block segmentation and text extraction in mixed text/image documents. *Graphical Models and Image Processing*, 20:375390, 1982.

[16] K. Y. Wong, R. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26:647–656, 1982.