Document Understanding of Graphical Content in Natively Digital PDF Documents

Aysylu Gabdulkhakova
Department of Computing Mathematics and Cybernetics
Ufa State Aviation Technical University
K. Marx str. 12, 450000, Ufa, Russia
aysylu.gab@gmail.com

Tamir Hassan
Pattern Recognition and Image Processing Group
Technische Universität Wien
Favoritenstraße 9-11, 1040 Wien, Austria
tam@prip.tuwien.ac.at

ABSTRACT

This paper presents an object-based method for analysing the content drawn by graphical operators in natively digital PDF documents. We propose that graphical content in a document can be classified either as structural or non-structural and present an output model for our analysis result. Heuristic techniques are used to group the instructions into regions and determine their logical role in the document's structure. Experimental results demonstrate the effectiveness of the algorithm.

Categories and Subject Descriptors: I.7.5 [**Document and Text Processing**]: Document Capture—*document analysis*

Keywords: Document analysis, document understanding, PDF, PDF operator, natively digital, logical structure, structural, non-structural

1. INTRODUCTION

Graphic objects play an important role in documents. Figures, diagrams and images can concisely and intuitively represent information in such a way that no amount of text can. Lines and rectangles can also be used to visually separate different logical parts of the document. The increasing need to reuse or repurpose this information has led to the development of document analysis and understanding techniques to detect their logical structure.

In natively digital PDF documents (*PDF Normal* or *Formatted Text and Graphics*) embedded vector graphics and other graphical content are drawn on the page using low-level primitives such as lines, curves, rectangles and glyphs. For the goal of repurposing or indexing this content we need to group such primitives into higher-level logical entities. Therefore, the task of document understanding of graphical content in documents is of high importance.

Most techniques for document understanding, such as [6], focus on the textual content, detecting individual structural items such as headings, paragraphs, etc. Our goal is to determine the logical role of the graphical content and classify each item as *structural* (e.g. ruling lines or rectangles that visually separate logical blocks from each other) or *non-structural* (illustrative content; the objects are grouped into regions) as shown in Figure 1. Here, the illustrative regions are highlighted in yellow, the structural elements are marked in green and the text regions in red.

Figure 1: The desired output of our algorithm

This paper proposes an object-based method that considers the geometrical relations between the graphic objects that appear in predominantly textual documents such as newspapers. In Section 2 previous research work relating to this problem is presented. Section 3 describes the methods we have developed for document analysis and understanding of vector graphic elements. Section 4 presents our ground truthing and evaluation procedures, and experimental results obtained by using this methodology. Finally, Section 5 concludes the paper.

2. RELATED WORK

There are several papers that describe methods that work on natively digital PDF documents, i.e. use the individual operators or another low-level representation instead of a bitmap representation as the initial starting point. The majority of these approaches are focused on textual information, so that graphical regions are detected as non-textual rather than explicitly analysed.

Chao and Fan [1] propose a method that combines object-based and bitmap processing approaches for information extraction. Text and images are detected directly from the PDF content stream and form distinct logical components. For the vector graphics they first find the regions which are more likely to have path objects using a document image segmentation tool and heuristically compare these regions with the path objects taken from the PDF content stream to determine the grouping of the logical components.

In contrast, Déjean and Meunier [3] present a system which uses only the PDF operators and uses a combination

DocEng'12, September 4–7, 2012, Paris, France.

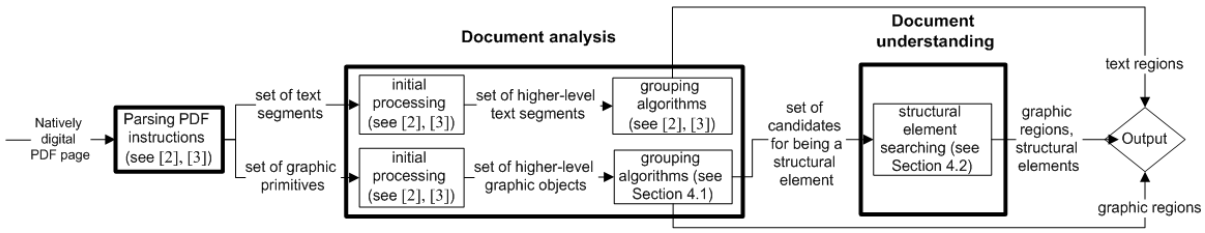

Figure 2: Stages of the whole algorithm for processing natively digital PDF page

of the X-Y cut algorithm and grouping heuristics to group textual and illustrative content into logical entities.

Hassan [4] also proposed an object-based method for the extraction of text and graphic objects directly from the PDF content stream. This method is included in the *PDF Extraction Toolkit (pdfXtk)*[1] which is built upon the *PDFBox* library and returns the graphical content as line, rectangle (filled and non-filled) and image objects, as well as text objects. Grouping algorithms are used to merge overlapping paths into regions for illustrative content.

In contrast to the above approaches, we determine the *logical role* of the vector graphic objects, i.e. we classify them as structural and non-structural. Using the output of pdfXtk as a starting point, our bottom-up approach uses geometrical properties and mutual arrangement of the given vector primitives to find illustrative regions and structural elements in the document.

3. METHODS

In order to represent a natively digital PDF document as a set of text regions, illustrative regions and structural elements, our system performs three processing phases. First, the page content is extracted from the PDF operators and transformed into Java object primitives. These primitives are defined by their rectangular bounding box coordinates in 2D Cartesian space. In pdfXtk the following types of primitive objects are returned: line segments, rectangles, bitmaps and text segments (corresponding to a single `Tj` or a partial `TJ` instruction).

Next, the grouping rules are applied in order to obtain higher-level graphic and text objects. In the final processing phase, we determine which of the graphical objects represent structural elements, as opposed to graphic regions. The diagram of the whole algorithm can be seen in Figure 2. The remainder of this section describes our grouping rules and our methods for determining whether a vector object is structural. The task of processing text blocks is not addressed in this paper (see [4, 5] for a description).

3.1 Grouping rules

The grouping rules that we have devised take into account geometrical properties of the graphic objects as well as their mutual spatial arrangement. We classify these rules into two categories: intersection-based and distance-based. When applied in combination with each other, they enable higher-level objects to be constructed, which usually correspond to distinct logical objects in the document's structure.

3.1.1 Intersection-based rules

Line and line. When the intersection between two lines is established, we group them together. For the next line, we check the intersection with each member of the group. The special case is when two lines construct a solid line, i.e. visually they are perceived as one. In this case we merge these line segments and no group is created.

[1]PDF Extraction Toolkit (pdfXtk): `http://pdfxtk.sourceforge.net`

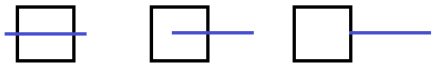

Figure 3: Three cases of intersection between line and rectangle objects

Rectangle and rectangle. This method is applicable not only to rectangles, but also to bitmap objects and complex figures (in the latter case, the bounding box is used). It is based on the assumption that two *structural* rectangle objects on the page are unlikely to intersect. Specifically, when the topmost coordinate of one rectangle is less than the bottommost coordinate of the other or when the leftmost coordinate of one rectangle is greater than the rightmost coordinate of the other.

Often, advertisements or other separated content is enclosed in structural rectangles, which are very close to each other or even overlap. Hence, before merging such elements using the above rule, we check whether they enclose other objects. If yes, then the given pair of rectangles is not grouped.

Line and rectangle. In predominantly textual documents, two types of intersection between line and rectangle objects can occur:

1. structural line intersects the rectangular object;
2. line segment intersects the rectangular object, both are part of a graphic region.

In order to avoid overmerging, three actions are sequentially performed:

a) the width ratio or height ratio, whichever is the larger, is compared to the given threshold;
b) we determine the type of intersection or, more precisely, the mutual arrangement of the intersecting objects (see Figure 3);
c) the ratio between both parts of the line, split at the intersection point, is compared to a given threshold.

Line and text. There are a variety of ways in which line segments and text fragments can intersect each other. In our research we focused on four cases that commonly occur in newspapers:

a) line segment underscores textual content;
b) line segment crosses a word;
c) line segments form the axes and text blocks represent labels (as in a chart or similar diagram);
d) text block is surrounded by lines which separate it from other parts of a document.

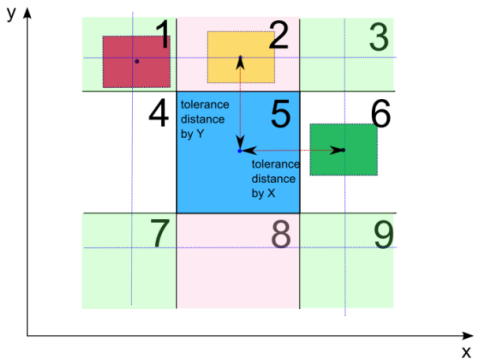

Figure 4: The grouping method for rectangles, based on the sizes and mutual arrangement of the given objects

Figure 5: Newspaper heading representation in *pdfXtk*

Cases a) and b) can be distinguished from each other by the distance between their centres projected on the Y-axis: if the line is closer to the centre of the text bounding box than to its border, then case a) applies; otherwise case b). Cases c) and d) can also be distinguished by the distance between their centres, but projected on the X-axis: if the line is touching or intersecting the text bounding box, then case c) applies; otherwise case d).

Rectangle and text. As in the previous situation, there are several possibilities of intersection between the given objects. More precisely:

a) rectangle encloses the text fragment;
b) rectangle intersects the text fragment;
c) rectangle touches (or partially overlaps) the boundary of the text fragment.

The last case occurs in tight layouts, where the rectangular bounding boxes of neighboring components often slightly overlap each other.

3.1.2 *Distance-based rules*

Line and line. Here the distance-based rules consider the possibility of dashed lines. Line segments represent small objects with the distance between them less than or equal to the element size.

Rectangle and rectangle. Two rectangles are considered as a single object if the distance between their centres is less than a given threshold. This threshold depends on two parameters: a granularity-level coefficient and the widths or heights of the rectangles. It is calculated by multiplying the first parameter with the sum of the second.

The granularity-level coefficient depends on a size ratio of the given rectangles and is divided into 3 types: high (0.55), normal (0.6) and low (0.65). These numerical values were obtained experimentally.

Next, the algorithm continues by detecting one of nine cases of mutual arrangement between two rectangles (as illustrated in Figure 4).

If the current rectangle is located in area 4 or 6 towards the rectangle being considered (green and blue rectangles respectively), the threshold distance uses the sum of both widths. For cases 2 or 8 (yellow and blue rectangles), the threshold distance depends on their heights. For the remaining cases 1, 3, 7, 9 (red and blue rectangles), two threshold distances are counted using the widths and heights.

Finally, the threshold distance is compared to the distance between the centres of the given objects projected on the appropriate axis. In cases 1, 3, 7 and 9 the comparison is conducted on both axes with the corresponding thresholds.

It is worth noting that our system represents composite objects by their rectangular bounding box. In such a manner, graphic glyphs that form parts of logos, newspaper headings etc. are introduced as rectangular objects. A vivid example of the above glyphs is the heading of newspapers such as *The Sydney Morning Herald*, *International Herald Tribune*, etc. (Figure 5). Here, low-level primitives are positioned sequentially in one row/column. In order to detect this case, we refer to the golden ratio font rules [2].

Errors can occur with advertisements that have the same size and are close to each other (Section 3.1.1, *Rectangle and Rectangle*). These advertisements differ from glyphs as they also contain text. Moreover, the advertising boxes are usually filled with objects as large as at least one third of their size, whereas glyphs have a negligibly small filled area.

3.2 Finding structural elements

Lines. Generally a structural line occurs as a horizontal/vertical line or rectangle that looks like a line, which does not intersect other non-structural objects (strokes, lines, rectangles, merged graphic regions, text fragments) and is not enclosed in any graphic region.

In Section 3.1.1, paragraph *Line and Text* we considered four cases in which line and text objects can intersect each other. Case a) is a special case, where the line is neither structural nor illustrative, but rather an integral part of the formatting of the text. In case b) the line is likely to form part of an illustrative region.

In cases c) and d) there is a pair of identical groups of straight lines with different semantics: in case c) these lines form part of a chart, whereas in case d) the lines are used as a "barrier" and serve the purpose of separating the text paragraph from the remaining page content. Thereby we conclude that c) is an example of non-structural lines and d) is an example of structural lines. In order to detect the correct variant, the closeness of line segments and text fragments is taken into account.

Rectangles. Generally, a structural rectangle occurs does not intersect other graphic primitives and regions, but can enclose them. The special case is that of "stand-alone" rectangle objects, which often look like and serve the same logical function as lines. Therefore, in Section 3.1.1, paragraph

	Total	Retrieved	Correct detections	Incorrect detections	Partial detections	Over-segmentations	Under-segmentations	False positives	Missed objects
Structural lines	847	875	710 (86%)	93 (11%)	0	0	28 (3%)	122	30
Structural rectangles	464	377	291 (69%)	119 (28%)	1 (<1%)	11 (2%)	4 (<1%)	68	42
Graphic regions	364	670	291 (82%)	55 (15%)	2 (<1%)	105 (29%)	32 (9%)	228	11

Table 1: Evaluation results

Rectangle and Text, case a), the rectangle can represent the bounding box of textual content and thus is more likely to be structural than in cases b) and c).

4. EVALUATION

We have tested our approach against manually generated ground-truth on 100 pages from 10 different newspapers taken from 15-17 April 2012: *Bresciaoggi*, *Il Tirreno*, *L'Eco di Bergamo*, *Nuovo Quotidiano di Rimini*, *El Mundo del Siglo XXI*, *Äripäev*, *China Daily*, *Die Tageszeitung*, *International Herald Tribune* and *Le Monde*.

The evaluation was performed on the bitmap level using the methods in [7]. We developed a graphical tool to align the manually generated ground truth and resultant images, enabling pixel-by-pixel comparison. As our approach goes further than current related work in this field – we do not just group the objects but distinguish between structural and illustrative items – we chose not to compare our results with those of other approaches.

First, the given PDF document is rasterized to create a binary image at a given resolution – 72 dpi, which is sufficient for this purpose. The ground truth is generated manually by colouring the black pixels in one of three different colours, one for each type of region: structural lines, structural rectangles and illustrative regions. In the same way, we automatically generate a result image from the output of our algorithm.

As soon as the resultant and ground-truth images are generated, pixel-by-pixel comparison is applied to compare the color values of the corresponding pixels.

As in [7], we use the following measures to determine the nature of overlapping between the regions:

- *correct detection* – regions are mainly overlapping;
- *partial detection* – some overlapping detected, but not sufficient as in the first case;
- *incorrect detection* – the types of region in ground truth and result of the algorithm are different (structural rectangle, structural line or graphic region);
- *false positives* – the region is marked by the algorithm, but does not occur in the ground truth;
- *missed objects* – the region is marked in the ground truth, but has not been detected by the algorithm.

The results of our evaluation are given in Table 1. The algorithm demonstrates a high performance on predominantly textual, natively digital PDF pages. The following paragraph discusses the possible causes of the errors that were encountered.

Limitations. One of the causes of some of the errors that have occurred is that the implementation of certain PDF operators in pdfXtk and PDFBox is incomplete. This can lead to invisible lines and rectangles, which may have been used for alignment purposes in typesetting the page, being output as rectangular objects, which in turn can lead to the entire region being detected as a graphical region.

Further limitations include the less common situations of using bitmap images or text elements to draw structural content, and using graphical elements to draw text. In the former case, some analysis steps on the bitmap rendition of the page are necessary to correctly detect its structure, whereas the output of an OCR product could be combined with our methods in the latter case.

Finally, it is worth mentioning that PDF has its roots in the page description language PostScript. As such, there is an almost limitless number of ways that a given visual result can be achieved, with completely different underlying operator structures. It is therefore possible to purposely format a PDF in such a way that it looks normal to the reader, but confuses algorithms that work directly on the operator level.

5. CONCLUSION

In this paper we have introduced a new approach to document analysis and understanding of vector graphic content in natively digital PDF documents. Our method includes grouping algorithms that take into account intersections, enclosures, overlapping and mutual arrangement of several types of objects. Moreover, we use a set of heuristic rules in order to define the purpose of the content: whether it is structural or illustrative. The experimental evaluation shows that our methods achieve good performance, and the causes of some of the errors have been determined.

Acknowledgement: This work was funded in part by the Austrian Federal Ministry of Transport, Innovation and Technology (Grant No. 829602).

6. REFERENCES

[1] H. Chao and J. Fan. Layout and content extraction for PDF documents. *DAS 2004: Proceedings of the International Workshop on Document Analysis Systems*, 3163:213–224, 2004.

[2] Y. Chernihov and N. Sobolev. *Composing Scripts*. Architektura-S, 2007.

[3] H. Déjean and J.-L. Meunier. A system for converting PDF documents into structured XML format. In *DAS 2006: Proceedings of the International Workshop on Document Analysis Systems*, pages 129–140, 2006.

[4] T. Hassan. Object-level document analysis of PDF files. In *DocEng 2009: Proceedings of the 9th ACM Symposium on Document Engineering*, pages 47–55, 2009.

[5] T. Hassan. *User-Guided Information Extraction from Print-Oriented Documents*. PhD thesis, Technische Universität Wien, 2010.

[6] S. Klink, A. Dengel, and T. Kieninger. Document structure analysis based on layout and textual features. In *DAS 2000: Proceedings of the International Workshop of Document Analysis Systems*, 2000.

[7] A. Shahab, F. Shafait, T. Kieninger, and A. Dengel. An open approach towards the benchmarking of table structure recognition systems. In *DAS 2010: Proceedings of the International Workshop on Document Analysis Systems*, pages 113–120.