PDF Document Layout Study with Page Elements and Bounding Boxes

Hui Chao, Giordano Beretta, Henry Sang
Hewlett-Packard Labs, Imaging Systems Laboratory
Email: hchao@hpl.hp.com, berettag@hpl.hp.com, hsang@hpl.hp.com

Abstract

The Portable Document Format (PDF) has been mostly used for posting the final form of documents. The aim of our project is to analyze the layout, to modify the layout or to re-use elements of PDF documents for different media. Using PDFEdit in Adobe SDK, we built tool to study the layout of documents and tool to select page elements to compose a new page. We demonstrate problems we encountered and propose possible solutions.

Introduction

The Portable Document Format (PDF) is a file format for representing documents in a manner independent of the application software, hardware, and operating system used to create them and of the output device on which they are displayed or printed [[1]]. In Fig. 1 we illustrate the role of PDF in a publishing workflow.

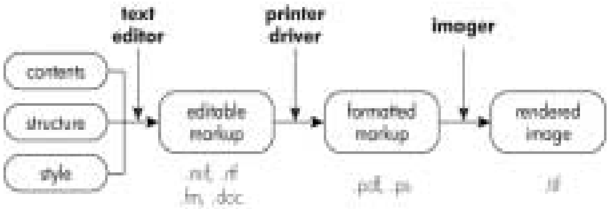


Figure1. Document creation workflow from the creation of editable (flowable) presentation markup through formatting (layout) to rendering to a specific device.

Authors start with a text editor and produce either a portable editable file such as Rich Text File Format (RTF) or an application specific editable file such as Microsoft Word. A printer driver is then used to either create directly a formatted PDF file, or indirectly by first creating a PostScript file which is then distilled into PDF.

As indicated by the uni-directionality of the arrows, the PDF is a document that is laid out for final presentation. Adobe Acrobat provides very limited editing functions [3, 4, 5]. While PDF1.4 allows for re-flow and some reformatting when structural information is preserved through the formatting, it does not generally support sharing and re-purposing the content. Indeed, very few PDF documents are created with structural information included.

The aim of our project is to analyze the layout of PDF documents to re-use elements of PDF documents or to modify the layout of documents for different media. The simple solution to the element extraction problem is to grab a portion of the bitmap from a screen shot of the rendered document. Directly working on PDF documents instead of the bitmaps of the screen shot has some advantages, i.e., the resulting materials preserve attributes and the look and feel of the original elements.

Methods

PDFEdit in Adobe Acrobat Software Development Kit (SDK) provides an access to PDF page contents with PDEContent class. Page contents can be treated as a list of objects whose values and attributes can be modified with PDFEdit methods. PDFEdit objects are independent of each other. Each object has all the relevant information about itself. By *elements* we refer to PDEElements in PDFEdit, a base class of page elements such as text, image, form, path etc. PDEText object is a PDEElement that represents text. It contains text as show strings or individual characters. Text consists of text runs, which are runs of characters in PDF file with the same attribute. A *text run element* represents a text run. Path is a graphics element, an arbitrary shape made up of straight lines, rectangles and cubic Bezier curves. Each PDEElement has an associated bounding box, which is guaranteed to encompass the element, but not necessarily the smallest box that could contain the element. Using the PDFEdit in SDK toolkit, we built tool to display the bounding boxes of page elements and selectively display the bounding boxes of text run elements; we built tool to acquire selected image and tool to extract and place page elements on a new PDF page from a user-defined region of interest (ROI). We collected PDF files from various source and generated PDF files with various methods for this study.

Results and discussions

We studied text, graphic and image layout of various sources, created single image document from a selected image in a PDF file, extracted page elements from ROI. An example is showed in Figure 2, in which part of page elements are extracted from a PDF page. We studied page elements layout by looking at the bounding boxes. We found that although elements, sub-elements and their associated bounding boxes in PDF document can be used in understanding a document's layout, it can also be misleading, inaccurate and complex in term of logical connection. Figure 3 shows a "*missing logical connection*" problem. An original PDF page generated by Adobe Acrobat PDFWriter 3.02 for Windows NT is shown in Figure 3(a), the bounding boxes of the text elements are also displayed. The element enclosed in the bounding box is then extracted and placed on a new PDF page in Figure 3(b). As can be seen in Figure 3(b), the two groups of text in this text element are not structurally grouped; a body of text gets divided.

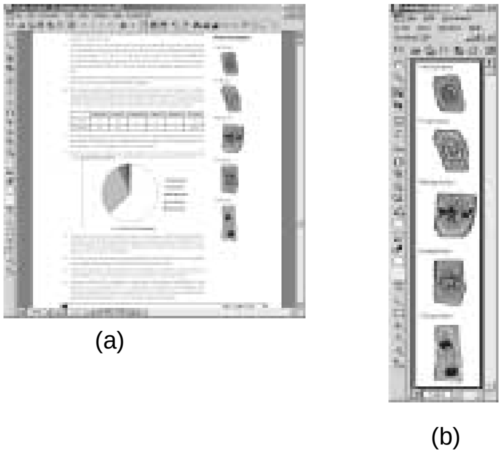


Figure 2. Page elements are extracted and form a new PDF page. (a) Original PDF page. (b) New PDF page from extracted elements.

Lacking structural information in the PDF file, text elements cannot be reliably used in understanding the structural layout of the document. In our experiments, the bounding box locations of text run elements were used. New text element consists of text run elements within user defined ROI were created for new document. Text-run bounding boxes are not the smallest boxes enclosing elements; they can only give an approximation of the location and size. We also found that the location and size of bounding boxes depend on the processes used in generating the PDF documents. Figure 4 shows two PDF documents that come from same original Microsoft Word document, (a) is converted with Adobe Distiller and (b) is converted with Adobe PDFWriter. In both Figure 4(a) and 4(b), bounding boxes for text elements and text run elements are displayed. In both figures the bounding boxes of PDF text run elements do not precisely define the position and size of the elements. They often extend beyond the actual elements and overlap with each other. Bounding boxes layouts are different in (a) and (b), that shows bounding boxes depend on not only the size and position of the actual elements but also the processes and tools used for the document formatting. When creating a new document from ROI, rules have to be imposed in identifying user's intention in selecting text. Rules should be created based on the application. It could be selecting any text-run elements whose bounding boxes intersect the ROI or only text run elements whose bounding boxes are within the ROI. More complicated rules could be made in selecting elements based on the ratio of overlap area of ROI with bounding box to the size of bounding box.

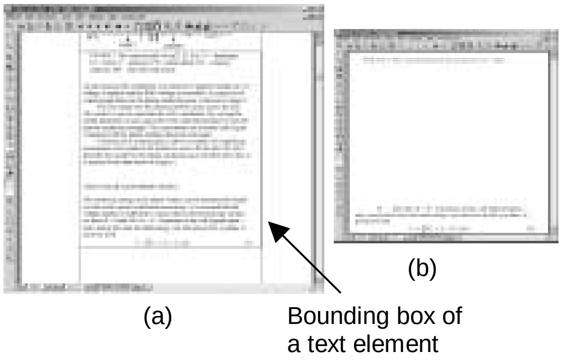
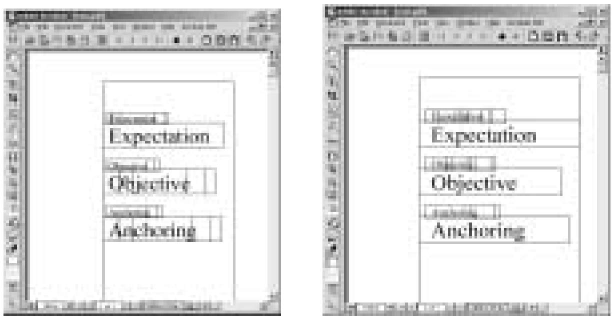


Figure 3. (a) The original page with bounding box of the text elements displayed. (b) A text element extracted from the original page with bounding box shown as the rectangular box in original page in (a).

Figures in a document should be represented in the PDF file by a single object or a group such as illustration, bitmap, etc. Figure 5 show a PDF document with a bar chart. Bounding boxes for all elements are displayed. From these bounding boxes, we can see that each bar, rule, and tick mark is an independent element, so is the background rectangle. The illustration we see is actually a group of seemly-unrelated graphics primitives. Re-flowing unstructured graphics can be difficult. Image vision or segmentation tools can be used in regaining document structure information.

Although page elements, sub-elements and bounding boxes can be used to interpret document layout, they are not reliable tools for recovering a documents structure. More information is needed to re-flow the document, change its layout, or extract elements for reuse in other documents. Using image-processing tool, studying the bitmap of the page, bounding boxes can be modified to be the smallest boxes enclosing the elements. Using image segmentation technique, geometric position analysis of text and image blocks [6,7] the logical structural information could be obtained. However Adobe SDK doesn't provide enough methods for developers to modify the attributes of page elements and its associated bounding boxes. PDF library will be required in that case.



(a) (b)

Figure 4. (a) A PDF document created by converting a Microsoft Word file to PDF with Adobe Distiller. (b) A PDF document created by converting a Microsoft Word file to PDF with Adobe PDFWriter. The rectangular boxes are the bounding boxes of text elements or text run elements.

Figure 5. Bounding boxes are displayed for graphic elements in the page. A bar chart is actually represented by graphic primitives.

Conclusions

In a typical workflow, PDF is sitting at the end of the formatter. In order to make the PDF files as small as possible the PDF format was originally designed to represent the final form only, the structural information has been generally discarded at each step in deriving the PDF. Although some text editors like Adobe FrameMaker have provisions to retain structural information through the formatting step, few documents make use of this capability. Indeed, even the manual of the recently released Adobe Acrobat 5 product has been formatted without structural information, despite having been authored with Adobe's own FrameMaker editor.

For reusing document elements in new documents, or reflowing documents for different devices (such as personal digital assistants), we have to reconstruct a document's original structure. We have discussed some problems we have encountered and outlined a possible solution, namely to regain document structural information from page elements bounding boxes location information, image segmentation results and geometric position analysis of text and image blocks.

References

[1] "*PDF Reference*" Second Edition, Adobe Systems Incorporated

[2] "*Acrobat Core API Reference*" Adobe Developer Technologies

[3] Walter, M. Seybold ,"Acrobat 4: Adobe's bid to make it more than a viewer" , Report on Internet Publishing. Vol. 3, no. 7.

[4] Venedek, A. " Is PDF set to become the universal file format?" British printer, Vol. 111, no. 8.

[5] Mckinley, T. " Why PDF is everywhere", Inform, Vol. 11, no. 8.

[6] ", Lovegrove, W.S., Brailsford, D.F., " Document analysis of PDF files; method, results and implicationsElectronic publishing: origination, dissemination and Design, Vol. 8 no. 2-3.

[7] Smith, P.N., Brailsford, D.F. , "Towards structured, block-based PDF", Electronic publishing: origination, dissemination and Design, Vol. 8 no. 2-3.