2009 10th International Conference on Document Analysis and Recognition

OCD: An Optimized and Canonical Document Format

Jean-Luc Bloechle, Denis Lalanne and Rolf Ingold
DIVA Group, Department of Informatics
University of Fribourg, Switzerland
{firstname.lastname}@unifr.ch

Abstract

Revealing and being able to manipulate the structured content of PDF documents is a difficult task, requiring pre-processing and reverse engineering techniques. In this paper, we present OCD, an optimized, easy-to-process and canonical format for representing structured electronic documents. The system and methods used for reverse engineering PDF documents into the OCD format are presented as well as the techniques to optimize it. We finally expose concrete evaluations of our OCD format compactness and restructuring performances.

1. Introduction

Since Adobe published the complete specification of its Portable Document Format in 1993 [1], PDF has become a de facto standard for electronic information exchange and archiving (cf. "Why PDF is Everywhere" [6]). PDF can be considered as a universal document format, since it is able to reproduce and preserve the original document appearance as well as any kind of printable information including text, drawings, business charts, and photos.

Despite the fact that the latest PDF specifications make possible to embed structural information about the content, most PDF producers do not make use of such features and focus only on the preservation of the original document appearance. As a consequence, lots of interesting features based on physical and logical structures are lost, although they were originally known and controlled by the document authoring software. Document re-usability, for instance, is limited to copy-paste operation of raw text data. In the case of complex multi-column layout, even the copy-paste operation is not guaranteed to work correctly when the selected text spans over more than one column. At worst, for poorly encoded PDF files, simple character sequences within a text line may not be respected at all.

Such limited functionalities have several drawbacks for document processing [5]. PDF documents can not be reedited, restyled, or re-flowed easily. For instance, it is not possible to change the general presentation of a document, to adapt it to another style, or to perform a logical text-to-speech for visually impaired people. Additionally, the copy-paste operation does not preserve logical labeling of headers, titles, or figure captures. Thus, reusing text excerpts of existing PDF documents often requires the restyling to be done manually.

To overcome these limitations, our research group has developed a complete and standalone structured electronic document format called OCD (Optimized Canonical Document). This file format is an optimized and enhanced version of our previous XCDF format and is supported by two tools: XED (eXploring Electronic Documents) extracts the physical structure from PDF documents and store the restructured content in the OCD format; Dolores (Document Logical Restructuring) uses OCD as input in order to recover the logical structures of electronic documents. The final goal is to convert a PDF file into a logically structured format that enables reapplying all kinds of editing operations offered by common text processing systems.

This paper is organized around 5 main sections. Section 2 emphasizes the weaknesses of our previous format, XCDF. Section 3 presents our new optimized canonical document format (OCD). Section 4 describes the processes leading to the physical restructuring of PDF documents. Section 5 presents the OCD format performance results and Section 6 concludes the paper.

2. XCDF File Format Weaknesses

XCDF is a complete and structured XML-based format developed in our research group that aims at representing static documents in a canonical way [4]. XCDF guarantees the full respect of the following principles:

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.159

- All the graphical primitives contained in the original document must be represented in an easy, concise and non-ambiguous way;
- The textual content must be hierarchically structured: homogeneous text blocks containing lines themselves divided into character sequences;
- The format must be user-friendly, i.e., easy to read/write and handle.

Dolores is an interactive tool including learning capabilities that is able to recover the logical structure of documents such as newspapers, e-books, scientific papers, journals, etc. [3]. Dolores' logical structure recovering is entirely based upon XCDF, our format that stores the physical structure, the content, and layout of static documents.

Practical uses of XCDF with Dolores recently showed major weaknesses: prohibitive file sizes and under-segmented text blocks. Indeed, Dolores is now able to proceed entire PDF newspaper documents holding dozens of pages, each of them containing lots of pictures, graphics and texts. Thus, XCDF files describing newspaper documents became really heavy, i.e., hundreds of megabytes. Such over-sized XCDF files led to very long read/write times and even out of memory problems.

Moreover, recovering the logical structure with Dolores thanks to the physical structure represented in the XCDF format is clearly not satisfying when dealing with documents having complex layouts. Indeed, the XCDF segmentation algorithm lacks a concrete definition, it does not specify clearly the segmentation level targeted and thus is not precise enough, i.e., paragraphs are not detected. XCDF text block under-segmentation prevents the recovering of precise logical structures with Dolores.

To overcome these drawbacks, we decided to move on toward a complete revision of XCDF. The new format should take benefit of the strengths of the old one while removing its weaknesses. Thus two different aspects have been addressed: first the XML representation itself has been completely reconsidered in order to optimize its storage size. Then the canonization process has also been reconsidered and greatly enhanced, in order to generate physical structures with finer adequate granularities.

3. OCD File Format

The OCD (Optimized Canonical Document) file format still keeps the fundamental objectives of XCDF: preserving the visual rendering of static documents while keeping a clean, canonical and structured internal representation. OCD solves the XCDF over-sized file problem by removing information redundancy and optimizing the internal representation. Further, as described in the next sub-sections, OCD is a good trade-off between structuring, readability and compactness.

3.1. The OCD Paradigms

Similarly to XCDF, three types of graphical objects exist in the OCD format: texts, vectorial graphics and raster images. Figure 1 shows the DTD of the OCD format. As in XCDF, font definition as well as clipping path descriptions are handled as document resources. In contrast, OCD adds the notion of a pool of objects in order to remove redundant images or graphics, OCD also uses far more compact data descriptions than XCDF. Furthermore, it is complete and standalone since all data are entirely embedded and external resource references are strictly forbidden.

OCD compactness is mainly achieved through information redundancy removal and character stream compression, thanks to the GZIP standard.

```
<!ELEMENT document (resources, pages)>
<!ELEMENT resources (fonts, clips, pool)>
<!ELEMENT fonts (font*)>
<!ELEMENT font (glyph+)>
<!ELEMENT clips (clip*)>
<!ELEMENT pool (path | image)*>
<!ELEMENT pages (page+)>
<!ELEMENT page (images, graphics, texts)>
<!ELEMENT images (image*)>
<!ELEMENT image (raster+)>
<!ELEMENT graphics (graphic*)>
<!ELEMENT graphic (path+)>
<!ELEMENT texts (block*)>
<!ELEMENT block (line+)>
<!ELEMENT line (token+)>
```

Figure 1. The partial OCD DTD

3.2 OCD Resources and Pool of Objects

The OCD file format begins with a description of the document resources, i.e., fonts, clipping paths and a pool of graphical objects. The advantage of defining document resources is that every page shares the same resources, thus avoiding the unnecessary duplication of information, e.g., a font used in several document pages is defined only once, the same is true for the clipping paths. In OCD, fonts and clipping paths are nothing more than graphical paths coming with their own attributes and referenced by means of ids.

Paths and images are often recurrent along a multi-page document, e.g., background images, logos, etc. Thereby, the OCD format defines a new concept called *pool of objects*: paths and images are compared each other and identical objects are stored only once in the pool of objects, and then referenced through pool ids.

3.3 Text Representation

Text representation is the most important part of our OCD file format and it is also the most elaborated one. In essence, text representation greatly benefits from our canonical text blocks generation (see Section 4). As text blocks are homogeneous chunks of text, their description needs very little information.

In XCDF text blocks do not take advantage of information redundancy. Every text attribute is written in the output file, whereas positions are expressed thanks to absolute coordinates. OCD uses relative positioning, it does not store redundant data and as such is far more space saving.

In OCD, each single page contains a unique internal text state (the same is true for images and graphics), This avoids a lot of information redundancy since an attribute is written only if its value has changed. Character codes are represented in Unicode thanks to hexadecimal values. Positions are then computed for each character glyph thanks to the current text state: *character space* (cs), *token space* (ts) and *white space* (ws) are combined with the *character width* (available in the font resources) and the current text *transform matrix*. Interline is expressed thanks to the *line space* attribute (ls). Figure 2 shows an example of a text block extracted from the newspaper "Le Monde" and containing the following text "Jacques Chirac\npresent sur\ntous les fronts".

```
<block x="705" y="252">
   <line>
     <token cs="-.36">4a 61 63 71 75 65
73</token>
     <token ws="5.184"/>
     <token>43 68 69 72 61 63</token>
   </line>
   <line ls="24">
     <token>70 72 e9 73 65 6e 74</token>
     <token/>
     <token>73 75 72</token>
   </line>
   <line>
     <token>74 6f 75 73</token>
     <token/>
     <token>6c 65 73</token>
     <token/>
     <token>66 72 6f 6e 74 73</token>
   </line>
 </block>
```

Figure 2. An OCD text block element

3.4 Graphic Representation

In XCDF a path element is represented with the <*path*> tag, itself containing elements such as <*line*>, <*cubic*> and <*quadratic*>. Each of these elements is described with attributes in absolute coordinates. In contrast, an OCD path is represented in a much more compact way, it is similar to an SVG path where sequential sub-path commands describe the overall path thanks to relative coordinates. Existing sub-path commands are *m*, *l*, *c*, *q* and *z* standing for *move*, *line*, *cubic*, *quadratic* and *close*, respectively. Each command's coordinates are relative to the previous command's coordinates, except the first *move*. Thus, every path must begin with an absolute *move* command. Cubic (Bézier) and quadratic curves allow complex curved paths to be represented accurately [2]. The *z* command closes a path by drawing a straight line back to the last *m* command.

3.5 Image Representation

In XCDF, raster images are represented by PNG files referenced by relative file paths. The OCD file format does no more allow external resources such as image files. Lossless image compression is handled with PNG whereas lossy image compression is handled with JPEG 2000. Image streams are directly embedded in OCD as byte streams encoded in hexadecimal.

4. The OCD Format Builder

An XCDF file is generated from a PDF document by merging its text primitives into text lines themselves merged into text blocks, resulting in under-segmented text (paragraphs are not segmented).

Figure 3. OCD vs XCDF text segmentation

Recent practical uses of our canonical format with complex document classes such as newspapers showed that XCDF text segmentation was clearly too sparse. Therefore, we decided to redesign our canonical document format builder in order to fill in this gap. As our previous text segmentation heuristics used only a

bottom-up scheme, we naturally came to add a top-down phase in order to split our previous XCDF text blocks into new finer ones, i.e., homogeneous paragraphs. Figure 3 compares the XCDF text segmentation (red dashed bounds) versus the OCD one (continuous bounds).

4.1 The OCD Physical Restructuring Process

The PDF reverse engineering process leading to the OCD file format is handled with XED. It proceeds in two main steps: firstly, it reads and converts the PDF document in a normalized internal structure called the *virtual document*. Secondly, this virtual document is analyzed in order to recover its physical structures and represent them in our OCD file format. The bottom-up phase proceeds in 6 key steps and has already been described in a previous paper [4]:

- Clean and trim all text primitives;
- Create a different layer for each text orientation;
- Merge layer text primitives into *tokens*;
- Merge layer tokens into text lines;
- Merge layer text lines into text blocks;
- Apply retroactive merging, i.e., parse all text blocks again and merge over-segmented lines.

The OCD top-down phase adds two new steps based on interline and text alignment.

First, sudden interline changes are detected in each text block. If the difference between two consecutive interlines is greater than a dynamic threshold (relative to the font size), the wider interline is used to split the text block in two pieces.

Then, text alignment is verified. The idea consists in splitting under-segmented text blocks into paragraphs thanks to their left/right alignments. For instance, first lines of paragraphs are often indented and therefore it is possible to use this information to split an XCDF text block into several OCD text blocks, i.e., paragraphs. This step is more tricky and subtle than it may appear at first sight.

Primarily attempts used the text block bounds as left and right alignment references. Results were clearly not satisfying with documents having complex structures. For instance, overlapping text block bounds are common in newspapers (see Figure 3), therefore real text bounds may be quite distant from the rectangular shaped text block bounds.

We then decided to detect the relative changes in text lines left and right alignments. Three rules have been implemented. Let i be the current text line, then h is the previous one and j the next one, and so on:

- If (g, h) are not right-aligned and (i, j) are strictly right-aligned and h is indented relatively to i, split the text block after h;
- If (g, h) are strictly left-aligned and (i, j) are not left-aligned, and i is indented relatively to j, split the text block after h;
- if (g, h) are not right-aligned and (i, j) are not right aligned too and the right alignment distance between g and h or h and i is greater than the first token width of i, then split the text block after h.

The first two rules work pretty well over fully justified text blocks, whereas the last rule has been added to specifically address non right-aligned text blocks. The text block splitting takes place only if it occurs between a line ending and a line beginning: a line ends/begins if its last/first character is a symbol or an uppercase character, or if the text font changes.

These restructuring steps are specifically adapted for Latin languages. Arabic and oriental languages are not currently supported. A relevant feature of our system is its ability to generate the OCD format over any PDF file without customization. Indeed, thresholds are not static values, they are ratios of dynamic values, such as font size. Thresholds have been refined iteratively over an heterogeneous PDF corpus including e-books, newspapers, scientific papers and journals. Moreover, used ratios (thus thresholds) tend to be minimal, over-segmentation being preferable than under-segmentation.

An evaluation of the OCD text block segmentations has been performed on three considerable documents: two newspapers and one e-book. More than 4000 text blocks have been evaluated. Table 1 shows the percentage of correct text block segmentations, i.e., heterogeneous paragraphs, in respect to human judgment.

Document title	# text blocks	% correctness
Le Monde 2009/01/16	1827	98.08%
La Liberté 2009/01/16	1410	98.87%
Alice's Adventures in W.	1108	99.37%

Table 1. Evaluation of the OCD segmentation

5 OCD Performance

The aim of OCD is to have a canonical, structured, compact and easy to handle file format that preserves exactly the original document appearance. Since OCD is produced by XED which has been proven to preserve exactly the document appearance [3], our evaluation has focused on its compactness. OCD has been compared against two electronic document standard, i.e., PDF from Adobe and XPS from Microsoft. We did our tests with 3 different classes of documents, i.e., e-books, city maps and newspapers.

Several non-tagged PDF e-books have been downloaded from PlanetPDF [7] and exported to XPS

and OCD. As they contain nearly only textual primitives, it is possible to grasp the compactness of the textual encoding scheme (Table 2).

Book Title (# of pages)	PDF	XPS	OCD
Around the W. in 80 Days (339)	766 KB	1'912 KB	427 KB
The Last of the Mohicans (698)	1'605 KB	3'944 KB	924 KB
Ulysses (1305)	2'953 KB	7'334 KB	1'743 KB

Table 2. E-books file sizes comparison

City maps have been acquired directly from official Internet city web sites. As they contain a lot of vectorial graphics they are a good testbed to determine the graphical encoding performances (Table 3).

Filename (# of paths)	PDF	XPS	OCD
Ascona City Map (3981)	822 KB	606 KB	416 KB
Estavayer Cadastre (107799)	1'544 KB	1'818 KB	1'306 KB
Manhattan Bus Map (4571)	265 KB	-	260 KB

Table 3. Maps file sizes comparison

Newspapers have complex document structures. They contain lots of text, graphics and images. This class of document reflects the general encoding performances of our file format (Table 4). As images in OCD are encoded using PNG and JPEG 2000, newspaper file size compactness is less relevant.

Newspaper	PDF	XPS	OCD
La Liberté 2005/02/08	11'203 KB	16'547 KB	8'586 KB
La Liberté 2009/01/16	5'422 KB	8'899 KB	4'620 KB
Le Monde 2009/01/16	9'093 KB	12'379 KB	10'664 KB

Table 4. Newspapers file sizes comparison

Tables 2, 3 and 4 show that even though OCD is a strongly structured and XML-based format its compactness is efficient compared to PDF and XPS. Moreover OCD improves PDF compactness when dealing with textual or/and graphical documents while still structuring its content. Indeed, the canonical format ensures that textual primitives are organized in homogeneous text chunks, thus allowing information redundancy to be greatly reduced, leading to a better compression. Moreover, redundant graphical primitives are described only once in a document pool.

The Microsoft XPS standard gives acceptable results. But it is important to notice that in XPS, images are most of the time down-sampled. Moreover vectorial graphics may sometimes be replaced by raster images, as it is the case for the Manhattan Bus Map (Table 3).

The file size reduction from XCDF to OCD is drastic. XCDF file sizes haven't been reported in the above tables for concision purposes. For instance, the XCDF file size of some of the previous documents are: "The Last of the Mohicans" e-book has a file size of 80'896 KB, the "Estavayer Cadastre" file size is 21'924 KB, whereas the newspaper "La Liberté 2009/01/16" has a file size of 77'619 KB (including the non embedded PNG images).

6. Conclusion

This paper presented OCD, an Optimized and Canonical Document format based on XCDF. Motivated by the heaviness of XCDF, our previous file format, OCD has proven not only to increase drastically its compactness, but also its readability and segmentation granularity as shown in the Section 5.

In the future, our investigations will again address the logical level (Dolores). Logical information will be directly encapsulated in OCD while keeping its current internal representation: logical structures will be added as new nodes that could simply be ignored by standard OCD readers, while enriching document indexing with logical information, allowing dynamic re-styling, document accessibility by visually impaired people, etc. Last, not least, OCD generation does **not** rely on the PDF internal structures and representation and thus may be generated directly by any document processing software, and vice-versa.

7. References

[1] Adobe Systems Incorporated, “PDF Reference, sixth edition: Adobe Portable Document Format version 1.7”, http://www.adobe.com/devnet/pdf/pdf_reference.html, 2006.

[2] Dokken, T., Dæhlen, M., Lyche, T., and Mørken, K, “Good approximation of circles by curvature-continuous Bézier curves”, Comput. Aided Geom. Des. 7, 1-4, 1990.

[3] J.-L. Bloechle, C. Pugin and R. Ingold, “Dolores: An Interactive and Class-Free Approach for Document Logical Restructuring”, In 8th Int. Workshop on Document Analysis Systems (DAS’08), 644–652, 2008.

[4] J.-L. Bloechle, M. Rigamonti, K. Hadjar, D. Lalanne, and R. Ingold, “XCDF: A canonical and structured document format”, In 7th Int. Workshop on Document Analysis Systems (DAS’06), 141–152, 2006.

[5] W. S. Lovegrove and D. F. Brailsford, “Document analysis of pdf files: methods, results and implications”, Electronic Publishing – Origination, Dissemination and Design, 8(3):207–220, 1995.

[6] T. McKinley, “Why PDF is Everywhere”, Inform, the journal of AIIM, 11(8), 1997.

[7] Planet PDF. http://www.planetpdf.com.