The hOCR Embedded OCR Workflow and Output Format

Thomas Breuel (editor)

- December 2007 - initial release
- March 2010 - bug fixes, clarifications

1 Rationale

The purpose of this document is to define an open standard for representing OCR results. The goal is to reuse as much existing technology as possible, and to arrive at a representation that makes it easy to reuse OCR results.

2 Getting Started

This document describes many tags and a lot of information that can be output. However, getting started with hOCR is easy: you only need to output the tags and information you actually want to. For example, just outputting **ocr_line** tags with bounding boxes is already very useful for many applications. Just start simple and add more output information as the need arises.

3 Terminology and Representation

This document describes a representation of various aspects of OCR output in an XML-like format. That is, we define as set of tags containing text and other tags, together with attributes of those tags. However, since the content we are representing is formatted text, However, we are not actually using a new XML for the representation; instead embed the representation in XHTML (or HTML) because XHTML and XHTML processing already define many aspects of OCR output representation that would otherwise need additional, separate and ad-hoc definitions. These aspects include:

- standard representations for common logical structuring elements, including section headings, citations, tables, emphasis, line breaks, quotations, citations, and preformatted text
- standard representations for fonts, embedded images, embedded vector graphics, tables, languages, writing direction, colors
- standard representations for geometric layout and positioning
- output files that are understood without any further modification by widely used viewers

(browsers), editors, conversion tools, and indexing tools
- libraries for parsing and generating the content
- support for document metadata

We are embedding this information inside HTML by encoding it within valid tags and attributes inside HTML; We are going to use the terms "elements" and "properties" for referring to embedded markup.
Elements are defined by the class= attribute on an arbitrary HTML tag. All elements in this format have a class name of the form "ocr…_…".
Properties are defined by putting information into the "title=" attribute of an HTML tag. Properties in title attributes are of the form "name values…", and multiple properties are separated by semicolons.
Here is an example:

```
<div class="ocr_page" id="page_1">
  <div class="ocr_carea" id="column_2" title="bbox 313 324 733 1922">
    <div class="ocr_par" id="par_7"> ... </div>
    <div class="ocr_par" id="par_19"> ... </div>
  </div>
</div>
```

The following properties can apply to most elements (where it makes sense):

- bbox x0 y0 x1 y1 – the bounding box of the element relative to the binarized document image
  - use x_bboxes below for character bounding boxes
  - do not use bbox unless the bounding box of the layout component is, in fact, rectangular
  - some non-rectangular layout components may have rectangular bounding boxes if the non-rectangularity is caused by floating elements around which text flows

- textangle alpha - the angle in degrees by which textual content has been rotate relative to the rest of the page (if not present, the angle is assumed to be zero); rotations are counter-clockwise, so an angle of 90 degrees is vertical text running from bottom to top in Latin script; note that this is different from reading order, which should be indicated using standard HTML properties

The following properties can apply to most elements but should not be used unless there is no alternative:

- poly x0 y0 x1 y1 … – a closed polygon for elements with non-rectangular bounds
  - this property must not be used unless there is no other way of representing the layout of the page using rectangular bounding boxes, since most tools will simply not have the capability of dealing with non-rectangular layouts
  - note that the natural and correct representation of many non-rectangular layouts is in terms of rectangular content areas and rectangular floats
  - documents using polygonal borders anywhere must indicate this in the metadata
  - documents should attempt to provide a reasonable bbox equivalent as well
- order n – the reading order of the element (an integer)
  - this property must not be used unless there is no other way of representing the reading order of the page by element ordering within the page, since many tools will not be able to deal with content that is not in reading order
  - presence presence must be declared in the document meta data