Paper to PDA

Thomas M. Breuel *William C. Janssen* *Kris Popat*
Henry S. Baird
Palo Alto Research Center, Palo Alto, CA 94304 USA
Email: {tbreuel,janssen,popat,baird}@parc.com

Abstract

A system is described for the automatic analysis of a document image into atomic fragments (e.g. word images) that can be reconstructed or "reflowed" onto a display device of arbitrary size, depth, and aspect ratio. The main intent is to allow scans and other page-image documents to be viewed effectively on a limited-resolution hand-held computing device, without any errors and losses due to OCR and retypesetting. The methods of image analysis and representation are described.

1 Introduction

One of the familiar advantages of symbolic representations of documents (ASCII, HTML, etc) over page-image representations is the ability to reflow the text to fit odd-sized displays. Reflowing typically breaks or fills lines of text with words, and may rejustify column margins, so that the full width of the display is used and no manual 'panning' across the text is needed.

Yet many documents already exist, and in their best form, as images of pages: that is, as a specification of intensities and colors over a rectangular field, devoid of any explicit textual information. The canonical example of this is the scanned document image; another important example is a book created for on-demand printing. How can such material be displayed on screens that are the wrong size and shape? An extreme case is reading images of books on hand-held devices like portable digital assistants (PDAs).

The obvious thing to try would be to convert the document into symbolic form by performing optical character recognition (OCR). However, even the best OCR systems are prone to errors which are expensive to correct. Even perfect character recognition usually discards or confuses the typefaces and relative type sizes selected by the author and publisher. To mitigate these problems, it has been proposed that when recognized text is displayed, those characters that received low recognition confidence scores in the OCR process be replaced by character images extracted from the original document [3]. We can take this idea a step further by foregoing the recognition altogether. In particular, the idea is to locate the text elements without necessarily recognizing them, cut them out of the image, and arrange the resulting bitmaps in reading order so that they can be reflowed in the available display space. While this approach eliminates errors and losses due to OCR, it presents three significant challenges:

- Determination of reading order
- Locating the "atomic" text elements to be treated as tokens
- Representing the resulting stream of elements in a useful form

The first two of these are problems in image and layout analysis, and are considered in Section 2. The third depends on the target viewing device or platform; Sections 3 and 4 describe a representation suitable for a class of PDAs.

2 Image and Layout Analysis

Image and layout analysis transforms the raw document image into a form that is reflowable and can be more compactly represented on hand-held devices (for further references about many of the techniques described in this chapter, the reader is referred to [2]).

Image analysis begins with adaptive thresholding and binarization. For each pixel, we determine the maximum and minimum values within a region around the pixel using greyscale morphology. If the difference between these two values is smaller than a threshold (determined statistically), the region is judged to contain only white pixels. If the difference is above a threshold, the region contains both black and white pixels, and the minimum and maximum values represent the blank ink and white paper background values, respectively. In the first case, the pixel value is normalized by bringing the estimated white level up to the actual white level of the display. In the second case, the pixel value is normalized by expanding the range between the estimated white and black levels to the full range between the white level and the black level of the display. After this normalization process, a standard thresholding method can be applied.

In the thresholded image, connected components are labeled using a scan algorithm combined with an efficient union-find data structure. Then, a bounding box is determined for each connected component. This results in a collection of usually several thousand connected components per page. Each connected component may represent a single character, a character part, a collection of touching characters, background noise, or parts of a line drawing or image. These bounding boxes for connected components are the basis of the subsequent layout analysis.

For layout analysis, we are primarily interested in the bounding boxes corresponding to characters in the running text of the document, as well as in a few other page elements like headers, footers, and section headings. We are interested in these particular bounding boxes because they give us important information about the layout of the page that we need for reflowing it. In particular, these bounding boxes and their spatial arrangement can tell us page rotation and skew, where we find column boundaries, what tokens we should consider for token-based compression, what the reading order is, and how text should flow between different parts of the layout. Bounding boxes that are not found to represent "text" in this filtering operation are not lost, however. They can later be incorporated into the output from the system as graphical elements.

The dimensions of bounding boxes representing body text are found using a simple statistical procedure. If we consider the distribution of heights as a statistical mixture of various components, for most pages containing text, the largest mixture component is going to be from lower case letters at the predominant font size. We use this size to find the x-height of the predominant font and use this dimension to filter out bounding boxes that are either too small or too large to represent body text or standard headings.

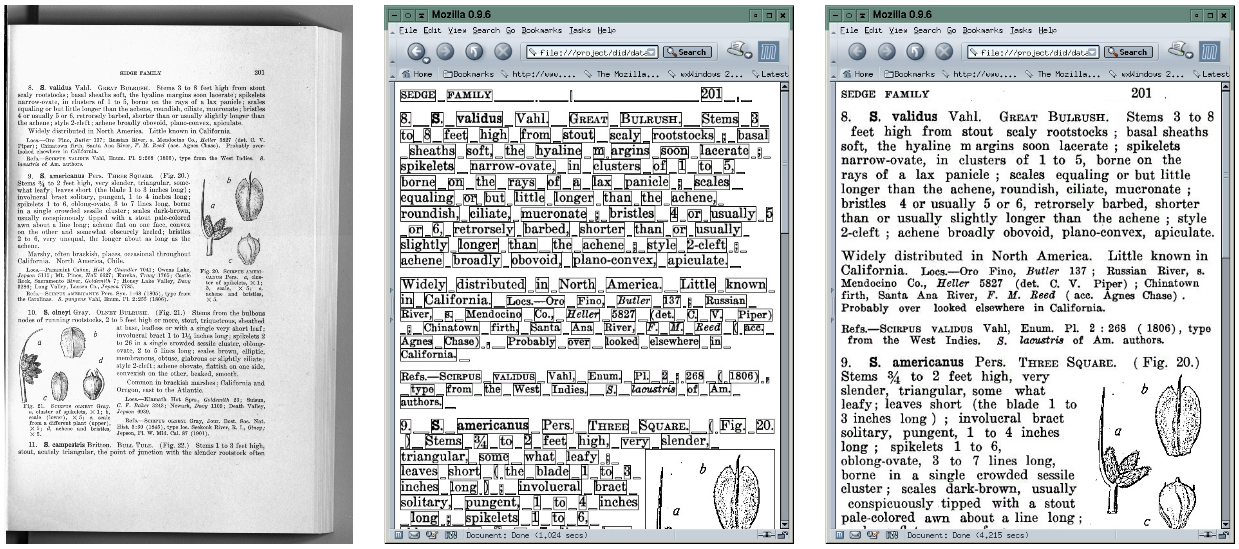
Given a collection of bounding boxes representing text, we are interested in finding text lines and column boundaries. The approach used in the prototype system for identifying text lines and column boundaries relies on a branch-and-bound algorithm that finds maximum likelihood matches against line models under a robust least square error model (equivalently, a Gaussian noise model in the presence of spurious background features) [1]. Text line models are described by three parameters: the angle and offset of the line, and the descender height. Bounding boxes whose alignment point, the center of their bottom side, rests either on the line or at a distance given by the descender height below it, are considered to match the line; matches are penalized by the square of their distance from the model, up to a threshold value ϵ, usually of the order of five pixels. After a text line has been found, the bounding box of all the connected components that participated in the match is computed, and all other connected components that fall within that bounding box are assigned to the same text line; this "sweeps up" punctuation marks, accents, and "i"-dots that would otherwise be missed. Within each text line, multiple bounding boxes whose projections onto the baseline overlaps are merged; this results in bounding boxes that predominantly contain one or more characters (as opposed to bounding boxes that contain character parts). The resulting bounding boxes are then ordered by the x-coordinate of their lower left corner to obtain a sequence of character images in reading order. Multiple text lines are found using a greedy strategy, in which first the top match is identified, the bounding boxes that participated in the match are removed from further consideration, and the next best text line is found, until no good text line matches can be identified anymore.

This approach to text line modeling has several advantages over the traditional projection or linking methods. First, different text lines can have different orientations. This is a common scanning artifact. Second, by taking into account both the baseline and the descender line, the technique can find text lines that are missed by other text line finders. Third, the matches returned by the method follow the individual text lines more accurately than most other methods[1].

Column boundaries are identified in an analogous manner, by finding globally optimal maximum likelihood matches of the center of the left side of bounding boxes against a line model. In order to reduce background noise, prior to applying the line finder to the column finding problem, statistics about the distribution of horizontal distances between bounding boxes are used to estimate the inter-character and inter-words spacing (the two largest components in the statistical distribution of horizontal bounding box distances), and bounding boxes for characters are merged into words. This reduces the number of bounding boxes that need to be considered for column matching severalfold and thereby improves the reliability of column boundary detection.

Based on the preceding analysis steps, the system now has a collection of text lines and column boundaries. Any connected components that are not part of a text line are grouped together and treated as images. For a single column document, by enumerating text lines and bounding boxes of images in order of their y-coordinates, we obtain a sequence of characters, whitespace, and images in reading order. For a double column document, the two columns are treated as if the right column were placed under the left column.

This simple layout analysis algorithm copes with a fairly wide number of commonly found layouts in printed docu-



Mozilla 0.9.6
File Edit View Search Go Bookmarks Tasks Help
file:///project/did/data
Search
Home Bookmarks http://www.... The Mozilla... wxWindows 2... Latest
Document: Done (1.024 secs)
Document: Done (4.215 secs)

SEDGE FAMILY 201

8. S. validus Vahl. Great Bulrush. Stems 3 to 8 feet high from stout scaly rootstocks ; basal sheaths soft, the hyaline m argins soon lacerate ; spikelets narrow-ovate, in clusters of 1 to 5, borne on the rays of a lax panicle ; scales equaling or but little longer than the achene, roundish, ciliate, mucronate ; bristles 4 or usually 5 or 6, retrorsely barbed, shorter than or usually slightly longer than the achene ; style 2-cleft ; achene broadly obovoid, plano-convex, apiculate.

Widely distributed in North America. Little known in California. Locs.—Oro Fino, *Butler* 137 ; Russian River, s. Mendocino Co., *Heller* 5827 (det. C. V. Piper) ; Chinatown firth, Santa Ana River, *F. M. Reed* (acc. Agnes Chase) . Probably over looked elsewhere in California.

Refs.—Scirpus validus Vahl, Enum. Pl. 2 : 268 (1806), type from the West Indies. *S. lacustris* of Am. authors.

9. S. americanus Pers. Three Square. (Fig. 20.) Stems ¾ to 2 feet high, very slender, triangular, some what leafy; leaves short (the blade 1 to 3 inches long) ; involucral bract solitary, pungent, 1 to 4 inches long ; spikelets 1 to 6, oblong-ovate, 3 to 7 lines long, borne in a single crowded sessile cluster ; scales dark-brown, usually conspicuously tipped with a stout pale-colored awn about a line long ;

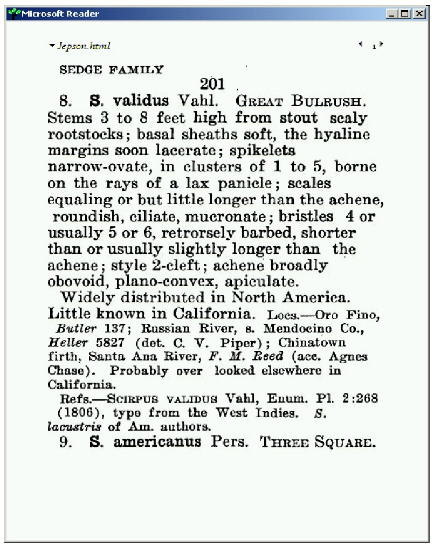
(a) (b) (c)

Figure 1. A page from Jepson's *A FLORA OF CALIFORNIA*

ments and transform them into a sequence of images that can be reflowed and displayed on a handheld device. In part, a simple algorithm works well in these applications because the requirements of reflowing for a handheld document reader are less stringent than for other layout analysis tasks, like rendering into a word processor. Since the output of the layout analysis will only be used for reflowing and not for editing, no semantic labels need to be attached to text blocks. Because the documents are reflowed on a small screen, there is also no user expectation that a rendering of the output of the layout analysis precisely match the layout of the input document. Furthermore, if page elements like headers, footers, or page numbers are incorporated into the output of the layout analysis, users can easily skip them during reading, and such elements may serve as convenient navigational signposts on the handheld device as well. Even some errors in layout analysis, like misattributing a figure caption to the body of a text, can be tolerated by readers. However, for very complex layouts, as found, for example, in magazines, more sophisticated document layout analysis techniques will have to be applied in order to arrive at a readable rendition on the handheld device. We are currently exploring the application of other layout analysis techniques developed in our lab to this problem.

3 Document Formats

The result of the decomposition process described above is a sequence of text images and illustrations, along with metainformation about formatting such as paragraph breaks and line justification. Many existing Web formats are well-



SEDGE FAMILY
201
8. S. validus Vahl. Great Bulrush. Stems 3 to 8 feet high from stout scaly rootstocks; basal sheaths soft, the hyaline margins soon lacerate; spikelets narrow-ovate, in clusters of 1 to 5, borne on the rays of a lax panicle; scales equaling or but little longer than the achene, roundish, ciliate, mucronate; bristles 4 or usually 5 or 6, retrorsely barbed, shorter than or usually slightly longer than the achene; style 2-cleft; achene broadly obovoid, plano-convex, apiculate.
Widely distributed in North America. Little known in California. Locs.—Oro Fino, *Butler* 137; Russian River, s. Mendocino Co., *Heller* 5827 (det. C. V. Piper); Chinatown firth, Santa Ana River, *F. M. Reed* (acc. Agnes Chase). Probably over looked elsewhere in California.
Refs.—Scirpus validus Vahl, Enum. Pl. 2:268 (1806), type from the West Indies. *S. lacustris* of Am. authors.
9. S. americanus Pers. Three Square.

Figure 2. The Jepson page rendered by Microsoft's Reader electronic book viewer

suited for representation of such data.

HTML[7], the standard for the World Wide Web, supports the layout in reading order of a sequence of image elements, along with formatting information. The successor to HTML, XHTML[8], uses the more rigorous XML syntax, but provides the same functionality. A group of commercial electronic-book interests are also defining the Open eBook Publication Structure[5], which also uses the XML syntax, incorporates the XHTML functionality, adds the ability to package multiple XHTML files into a single publication, and provides standards for addition of document metadata.

Figure 1(a) shows a sample page from Willis Linn Jepson's *A FLORA OF CALIFORNIA*[4], an important scholarly work in the field of botany, in which typeface and relative type size choices are significant. Once decomposed into image elements, the logical connections between those elements can be represented with HTML. Figure 1(b) shows an HTML representation of the decomposition of the Jepson page rendered in a standard Web browser, with boxes drawn around individual image elements.[1] Figure 1(c) shows that same page, but without the bounding boxes around the elements. Note that fonts and and some characteristics of the original page have been retained, which would not be the case for an OCR-based transition strategy.

4 Reader Applications

A problem with all of the Web formats is that a decomposed document will typically consist of hundreds of thousands of separate files. This configuration tends to strain the capabilities of underlying technology support platforms. Token-based compression schemes[6] can make this problem somewhat more manageable, but cannot really solve it. Most electronic-book document formats, however, alleviate this problem by packaging the image elements together with the layout directives in a single file.

Dozens of electronic book formats exist, among them Microsoft Reader, Adobe Acrobat Reader, Palm Reader, and Gemstar's RCA 1100 and 1200 formats. Our system of decomposed image elements can be supported by most (probably all) of these formats. Figure 2 shows an example of the Jepson page displayed in Microsoft's Reader viewer program. This was achieved by rendering the decomposed elements into Open eBook Publication Structure, then using the Overdrive Readerworks distiller for Reader to create the electronic book. Similar approaches seem to suffice for all of the other popular electronic book formats.

Common electronic book formats, and the associated viewer applications, are optimized for 'books' consisting mainly of text, with occasional images. This can lead to performance problems in both the time and space domains. Microsoft Reader version 1[2], for example, running on a 500 MHz Pentium III machine, takes approximately 20 seconds to lay out and display the first page of the Jepson sample shown above. But another system called 'Plucker'[3] provides timely layout and scrolling of the converted document, and averages only 70KB per converted page. We are currently investigating alternative storage formats and layout/display techniques optimized for this class of electronic book.

[1]The illustrations have been manually removed from the original, and manually re-inserted into this version. We are investigating ways of doing this automatically.

[2]see `http://www.microsoft.com/reader/`

[3]see `http://www.plkr.org/`

5 Summary and Conclusions

We began with the observation that documents that originate as high-resolution page images must be converted for display on hand-held devices, whose screens often differ from their desktop counterparts in size, depth, and aspect ratio. We have described a system for performing the conversion that is particularly effective when the document is predominantly textual.

Our system has several strengths. First, it achieves reflowing of the text without requiring the brittle and computationally expensive process of text recognition. Second, the transformed representation can be realized using existing description protocols, such as HTML and the popular electronic book formats. Third, by basing the representation on image fragments from the original, important visual aspects such as typeface are faithfully preserved. Finally, the approach is directly amenable to token-based compression schemes (future work), raising the prospect of greatly enlarging the amount of such material that may be stored on a single PDA.

References

[1] T. M. Breuel. Robust least square baseline finding using a branch and bound algorithm. In *Document Recognition and Retrieval VIII, SPIE, San Jose*, 2002.

[2] H. Bunke and P. S. P. Wang. *Handbook of Character Recognition and Document Image Analysis*. World Scientific, 1997.

[3] T. Hong and S. N. Srihari. Representing OCRed documents in HTML. In *Proceedings of the IAPR 1997 International Conference on Document Analysis and Recognition (ICDAR 1997)*, pages 831–834, Ulm, Germany, April 1997.

[4] W. L. Jepson. *A Flora of California*. University of California Press, http://ucjeps.berkeley.edu/jepson-project3.html, 1909-1943.

[5] John Alger et. al. Open eBook Publication Structure 1.0.1: Recommended Specification. Technical report, Open eBook Forum, http://www.openebook.org/, July 2001.

[6] Joint Bi-Level Image Experts Group (JBIG) Committee. Information technology – coded representation of picture and audio information – lossy/lossless coding of bi-level images. Technical Report 14492 FDC, ISO/IEC, July 1999.

[7] D. Raggett, A. L. Hors, and I. Jacobs. HTML 4.01 Specification. Technical report, World Wide Web Consortium, http://www.w3.org/TR/html4/, Dec 1999.

[8] Steven Pemberton et. al. XHTML 1.0: The Extensible Hyper-Text Markup Language. Technical report, World Wide Web Consortium, http://www.w3.org/TR/xhtml1/, Jan 2000.