

Design and Implementation of a Universal Personal Digital Library System

William C. Janssen and Kris Popat

Palo Alto Research Center (PARC)
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
{janssen, popat}@parc.com

Abstract. We trace through the initial design and early evolution of a personal digital library system pieced together largely from open-source components (the current exception being the OCR engine, which is proprietary). The system consists of a repository, a search engine, and a Web interface. It is suitable for personal collections comprising tens of thousands of documents (including papers, books, photos, receipts, etc.), and provides for ease of document entry and access as well as basic security and privacy. Representation of documents in an image-domain format is taken as canonical, exploiting the fact that documents are ultimately presented as visual objects to a user to achieve some degree of “universality.” Provision is made for alternative representations existing alongside the image-domain representation, either stored or generated on demand. At the conference we will demo the document-access portion of the system implemented self-contained on an ultra-portable laptop holding about 2,000 documents including many scanned books.

1 Introduction

The Universal Personal Digital Library (U-PLIB) project is aimed at capturing all the documents involved in normal human life in an organized digital form such that:

- Documents can originate from any source and in any initial format;
- Documents can be retrieved using a natural combination of full-text retrieval, metadata and category information, and visual search among initially retrieved thumbnail arrays to achieve a comfortably tradeoff among precision, recall, and user involvement;
- The contents of large documents such as books can be random-accessed through a Web interface, without having to launch an application or plug-in with its attendant time- and space-overhead;
- Tens of thousands of documents can be accommodated; and
- A reasonable level of privacy and security can be assured

Among the documents we include credit card bills, family pictures, favorite books, letters of various kinds, receipts, tax forms, Web pages, and the other

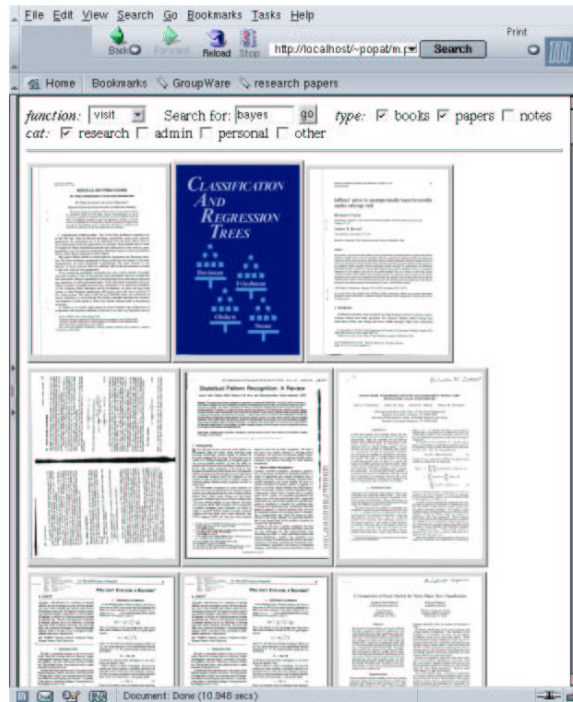


Fig. 1. Initial Web interface, implemented in PHP4 and calling *htsearch* through the *cgi-bin* mechanism. The search results from *htdig* are parsed by the PHP script to form the names of the precomputed thumbnail images, which are then used to make HTML4 image buttons for the retrieved documents. Clicking on one of the retrieved-document buttons has an effect determined by *function* (see Fig. 2.).

flotsam of modern life. As researchers, we also place particular emphasis on technical papers and notes, both ours and those of collaborators. Our intention is to deploy this library in multiple forms, including one that can be carried with you always in a small device like a PDA. Our approach is centered on document images, rather than text; we feel that the image domain presents certain unique opportunities not available in the symbolic domain, including a degree of universality and resistance to “bit-rot” or obsolescence of digital format.

This paper describes the two prototypes of a U-PLIB system that have been built, or are under construction, as part of the PARC Productive Reading effort. Each of these prototypes can be broadly divided into five functional stages: capture, normalization, storage, retrieval, and use. The following sections describe each of these stages.

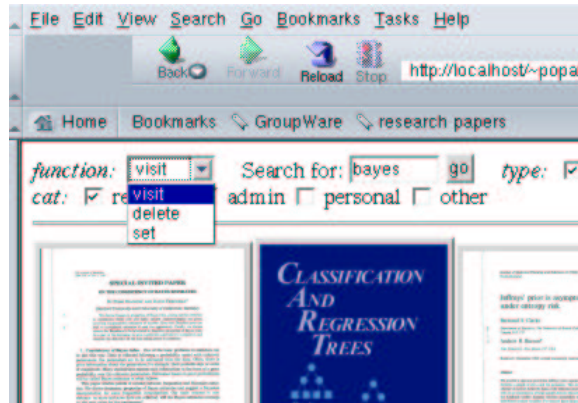


Fig. 2. Pull-down menu for *function* determines what clicking on a thumbnail in the retrieval array does. This is useful for quickly deleting visually-detected duplicates, or quickly re-categorizing incorrectly classified or unclassified documents that come up in a search. The function list can be trivially expanded by adding an item, then writing a corresponding PHP function of the same name to implement it.

2 Capture

Document capture can be done in a number of ways, and for a number of different purposes. In one study of information capture, 65% of the paper documents being captured were standard letter-size, and of those, 5/8 were only one page in length, while only 6% were longer than 3 pages in length [5]. Many of the other documents were handwritten notes or newspaper clippings. The same study proposes 10 reasons why documents are captured. Of these, half seem related to personal document libraries, while four of the remaining five are about collective document use. Only one, “capture for task management,” seems hard to understand in the role of document libraries, rather than for some form of personal information management. In addition, the increasing availability of documents in electronic form makes it clear that these formats have to be supported as well in a personal document library. Indeed, Web “clippings” may come to replace conventional newspaper clippings.

The proliferation of small cheap digital cameras suggest that they will also be used extensively for informal document capture of small paper documents, as well as for taking pictures and movies. Effective use of these documents requires a number of automated image enhancement steps, to de-warp and otherwise clean up the document image. A number of papers have already discussed these problems and potential solutions, such as the CamWorks paper [13].

Our current prototypes use both scanners, for paper documents, and document converters, for electronic documents. Scanning can be done either manually or automatically with systems such as Xerox’s Flowport. Document converters such as PARC’s documents.com allow us to convert PDF, HTML, Microsoft Word, and other formats to TIFF. Metadata available in the document can be

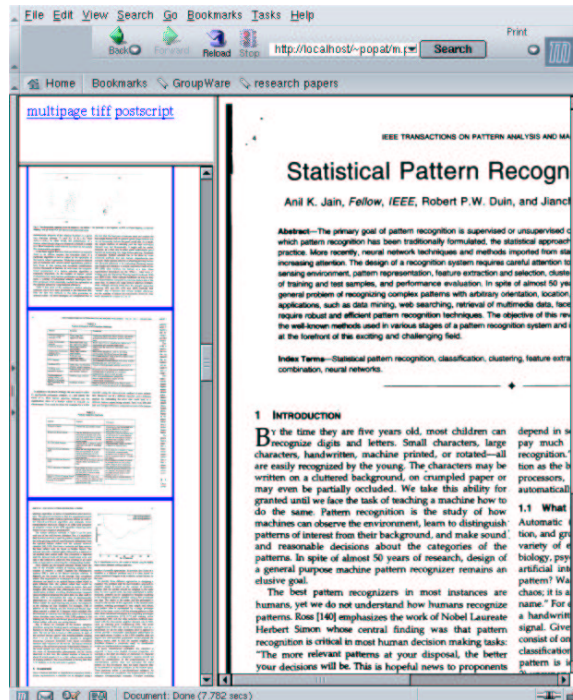


Fig. 3. An example of insufficient window size, as would be the case on a low-to-moderate-resolution display. The need to scroll, especially horizontally, severely impedes the reading process. As mentioned in the text, the UbiText project [4] may offer a solution, and may be adopted in U-PLIB for small-screen devices.

transferred to our metadata cache, as can textual content extractable from the document. The end result is a multi-page TIFF file containing the document, and one or more accessory files containing metadata about the document. Within the TIFF file, various encodings may be used on different pages; for instance G4 compression is typically used on purely bitonal pages. A directory having a unique name based on the current time, called the “document folder,” is automatically generated in a “staging area.” The files are placed in the document folder. If the TIFF file has been generated from another electronic form, a copy of the original document is also copied to the folder.

3 Normalization

For use, we require that each document be pre-rendered in several different forms. Our current system requires that a file containing the symbolic text of the document be present, that thumbnails of each page of the document (in two different sizes) be present, and that a standard set of HTML files for accessing the document be present in a subdirectory of the document folder. To accom-

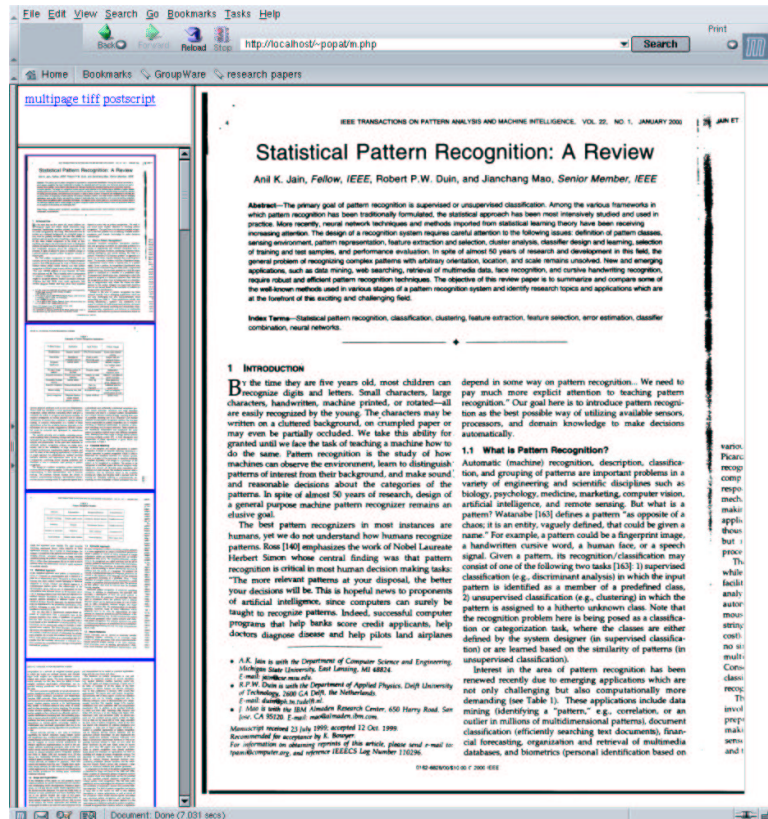


Fig. 4. Full page is viewable on a 1280 × 1024 pixel screen, provided that the image has been properly anti-aliased. In our system, appropriately anti-aliased images are precomputed at several resolutions. In principle, more resolutions could be generated on-demand through the *cgi-bin* mechanism, although this has not yet been implemented.

plish this, we use one or more daemons that watch the staging areas and take appropriate actions.

The first transformation is to generate the symbolic text, if none is already included in the metadata. An OCR service is called to try various OCR scans on the document. The output of these scans are compared to determine the best one by minimizing the empirical per-character entropy with respect to a character-5-gram language model (currently for English, but easily re-trainable on any other language representable in UTF-8). The output of that OCR scan is then added to the stored metadata for the document. A second transformation generates the thumbnails for each page of the document; in our current system a small thumbnail used for document overview is generated along with a full-screen image used for actual page presentation. The third transformation generates a

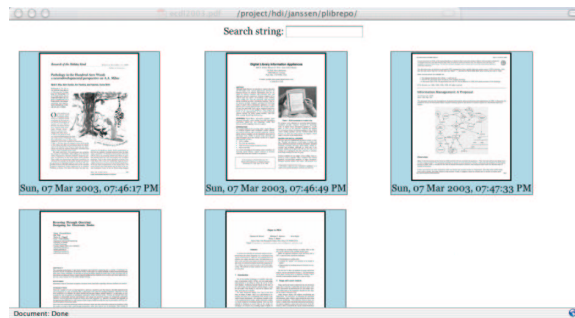


Fig. 5. Another thumbnail Web interface. The search form invokes a Python CGI script, which calls *Lucene*, a Java program, as a subprocess.

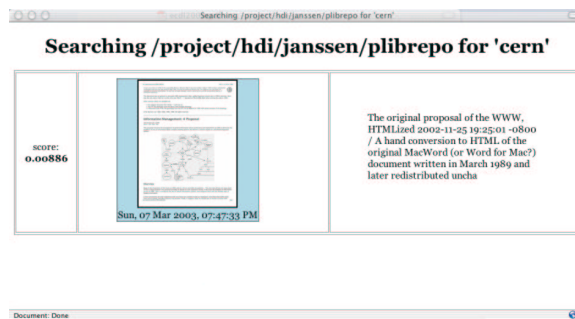


Fig. 6. The search results from *Lucene* are parsed by the Python script to form document descriptions consisting of the scores, thumbnails, and summaries of the documents, which also serve as HTML links to retrieve any of the documents.

set of HTML views of the document, drawing on the metadata and symbolic text for a summary, and on the thumbnails for page views. All of these elements are added to the document folder automatically.

Since each transformation works separately of the others, additional transformations can easily be added. We envision a library of possible analysis engines, each performing some metadata extraction or pre-use caching operation, that can continuously “rove” over the contents of the library, making opportunistic improvements. For example, as researchers, we’d like to integrate simple research paper metadata extraction similar to that used in CiteSeer [3], which would only run on documents that sufficiently resemble technical papers. Other transformations might perform URL extraction from Web pages, headline and dateline extraction from newspaper clippings, face recognition on family photos, or mailing-list classification on email messages.

4 Storage

When the document has been normalized, the document folder is automatically moved from the staging area to a protected volume for storage. This action also causes the text of the document to be indexed in a full-text retrieval system; our first prototype used ht://Dig [9] for this purpose, and our second uses Jakarta Lucene [15]. The protected volume is typically an encrypted disk image accessible via SSL-encrypted HTTP to a Web server on an otherwise-isolated server machine. We are exploring a lighter-weight deployment mechanism for our second prototype, but we feel that this level of security is important to maintain for actual use, and that the full-text index must be protected with the same level of security. Public or group repositories may be created with less stringent privacy protection, but we view these as special cases rather than norms. When the U-PLIB is accessed as a self-contained system on a laptop, the documents are stored on an encrypted partition that is unmounted upon inactivity-timeout or power-down.

5 Retrieval

Once a document has been stored in the repository, most access to it is with a Web browser over SSL-encrypted connections. Currently, there are two major interfaces to the repository: a visual presentation of all the documents in the repository, in thumbnail form, and searching over the textual index.

Examples of possible thumbnail overviews can be seen in Figures 1 and 5. This is convenient for a small number of documents, and tractable for a somewhat larger number. However, more sophisticated information management techniques need to be used for the number of documents we envision a typical repository as having. There is a great deal of work going on in this area which we expect to benefit from. We currently plan to incorporate a time-based organization scheme, such as a *perspective wall* [11], or perhaps a hierarchical clustering scheme [2], [8].

However, in a large database of documents, searching may be the primary way of finding documents. Our prototypes allow a search string to be specified by the user, using the particular query syntax supported by the underlying full-text indexing engine. This string generates a set of document matches, and the system then needs to indicate those matches to the user in some way. Figure 6 shows one way of doing this, implemented in our second prototype. Each result consists of three parts: the Lucene relevance score, the thumbnail image of the first page of the document, and a summary of the text of the document. Clicking on any of these takes one to the actual document.

6 Use

Currently, the primary use supported is viewing or reading the document, with provision to provide a printable copy (PostScript or PDF) on demand. When a document is presented, it consists of a column of thumbnails of all the pages,

along with a larger view of the current page. A small control area allows a PDF or text form of the document to be retrieved. We've found the image presentation to be remarkably effective, even of documents that are originally not in "virtual paper" form, i.e. Web pages. Typically, once it is determined that a document is of interest, it is printed out for more comfortable reading, then the printed out version is eventually discarded for recycling.

The second author has used the first prototype of this system to eliminate one large 4-drawer filing cabinet from his home, and one of about the same size from his office. He currently has about 2,000 documents in his U-PLIB, and accesses it daily. On average, he adds about 2.5 new documents to the system per day.

An important benefit of the U-PLIB is that it allows paper to be discarded. Before an important paper document that has been entered into the system can be thrown away or shredded, the digital version must be verified for quality and retrievability. This verification stage is by far the most time-consuming aspect of entering documents into the system, but our experience has shown that it is not prohibitively burdensome. For retrievability, an end-to-end check is made for each newly entered document by attempting to retrieve using the query terms that come to mind, and if the retrieval fails, then those query terms are artificially added to the end of the text-version of the document. Missing pages are checked for in large documents by inspecting every 10 or 20 pages, and making sure that the page numbers are incrementing by the expected amount. Scanning problems such as excessive skew or bad exposure are checked for by quickly scrolling through the thumbnail bar.

Most documents entered into the system will never be accessed, but some will, and it is important that they all be retrievable so that the paper versions can be thrown away. Our personal information needs cannot be predicted perfectly in advance, so it is safer (and now feasible) to store virtually everything, taking advantage of plummeting hard disk costs, and shifting the burden of filing onto a well-designed retrieval process. So in addition to the day-to-day tasks of looking up research papers and notes, examples of unanticipated question easily answerable with the U-PLIB include "what diameter contact lenses did my optometrist in Massachusetts prescribe?" and "how much did we pay in un-reimbursed medical expenses in 2002?" The potential effect on filing practices and information availability of adopting the U-PLIB into daily work-flow appears to be significant.

7 Architecture

There are a number of ways to build systems of this type. Our current focus is on a client-server model, with a thin client built mainly with the HTML+JavaScript user-interface "toolkit" communicating over an secure RPC channel to a server back-end that performs task-specific actions. The client will thus run in any modern Web browser, which we see as a significant advantage. The server is implemented as a daemon running on a user's computer with the user's iden-

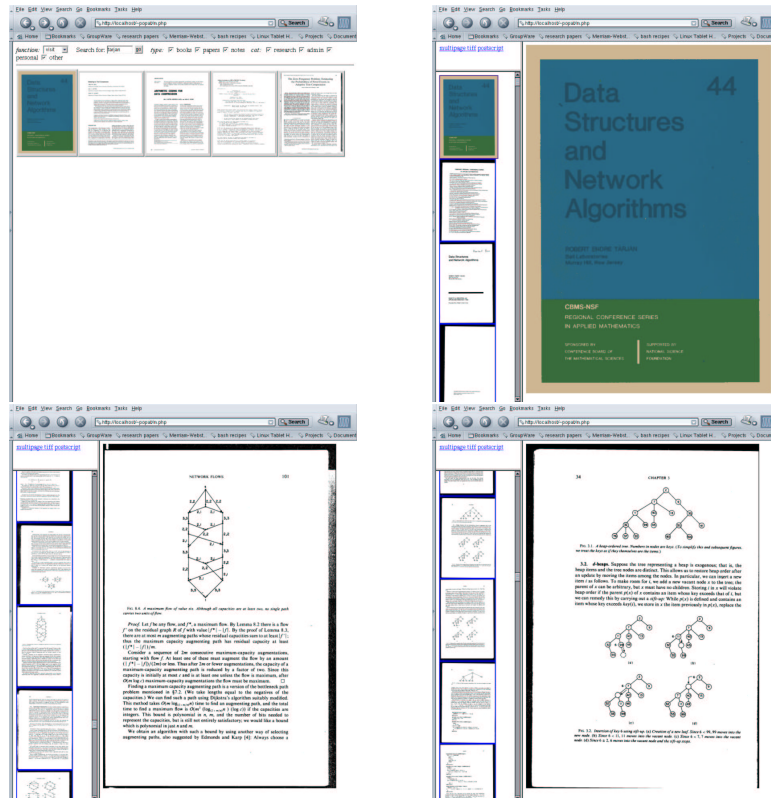


Fig. 7. Retrieving and browsing a book. On a 600 MHz Intel Pentium III machine, each page displays in the order of 200 msec, without prior caching, irrespective of position in the book. Top left: search results for “Tarjan.” Remaining images: page views as the thumbnail bar to the left is browsed. The functionality is similar to that provided by Adobe Acrobat Reader, but somewhat faster and without the need for a plug-in or separate viewing application.

tity, exporting functionality to the client via the standard HTTP/SSL protocol. In addition to servicing RPC calls from the client (or clients), it also periodically scans directories designated as *drop-off points*, moves documents or partial folders found in those directories to the staging area, and performs the various normalization tasks described above.

8 Future Work

An alternative to the HTML/JavaScript front end supported by a Web browser is one written specifically for our application, either as a stand-alone program, or as a Java applet. Our intention is to avoid this if possible, both to reduce barriers to adoption and to provide resistance to obsolescence. Another possibility is to

use HTML/JavaScript in conjunction with a custom Web browser, such as the highly portable Multivalent Browser [14] being developed at Berkeley, which supports a greater range of document interactions. Annotation is important for example, and the Multivalent Browser supports this.

We'd also like to support use of documents in smaller display contexts. The availability of small PDA devices with high-resolution color screens and Bluetooth automatic connectivity, such as the Palm Tungsten T, and other devices with very large storage capacity, such as the Apple iPod, suggest that pocket-size devices with both of these capabilities are likely in the near future. A pocket-size wirelessly available "container" holding all of your personal documents, as well as serving as an MP3 player and PDA, seems a likely outcome of these trends, and we intend to run our U-PLIB on these devices when they become available.

But interacting with documents on these devices presents special challenges [12]. Effective use of personal documents on these devices will require new ways of dealing with images, particularly images of text. One promising approach, PARC's Ubitext [4], offers a way of re-flowing (essentially re-typesetting) images of pieces of the text (characters and words), and we hope to use it in future versions of our system.

9 Conclusion

We have described the objectives, design, and implementation of a universal personal digital library that emphasizes image-domain representation of documents, and is largely based on open-source components. Specifically, the architecture and functionality of two prototypes has been sketched; more details and a demo will be presented at the conference. The first prototype of the U-PLIB system has been adopted into the daily work-flow of one of the authors over a period of about two years, providing an "existence proof" that the system is both usable and useful. Several colleagues have expressed a desire to adopt the system into their own work-flow. The second prototype is partially in response to this; it is intended to be more powerful, more robust, and easier to use by a novice. We expect that it will be adopted by a growing user community within PARC, and that it will continue to improve in usability and functionality as a result. In the longer term, we expect to release an open-source version of the system externally. Currently, the largest obstacle to doing this is the unavailability of a reliable open-source OCR engine.

References

1. Steven C. Bagley and Gary E. Kopec. Editing images of text. *Communications of the ACM*, 37(12):63–72, 1994.
2. Benjamin B. Bederson. Photomesa: a zoomable image browser using quantum treemaps and bubblemaps. In *Proceedings of the 14th annual ACM Symposium on User Interface Software and Technology*, pages 71–80. ACM Press, 2001.

3. Kurt Bollacker, Steve Lawrence, and C. Lee Giles. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In Kattia P. Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 116–123, New York, 1998. ACM Press.
4. Thomas M. Breuel, William C. Janssen, Kris Popat, and Henry S. Baird. Paper to PDA. In *Proceedings of the 16th IAPR International Conference on Pattern Recognition*.
5. Barry A. T. Brown, Abigail J. Sellen, and Kenton P. O'Hara. A diary study of information capture in working life. In *Proceedings of 2000 ACM Special Interest Group on Computer-Human Interaction (CHI2000)*, pages 438–445, 2000.
6. Stuart K. Card, George G. Robertson, and William York. The webbook and the web forager: An information workspace for the world-wide web. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'96*, 1996.
7. James R. Davis, Carl Lagoze, and Dean B. Krafft. Dienst: Building a production technical report server. In *Advances in Digital Libraries*, pages 259–271, 1995.
8. Adrian Graham, Hector Garcia-Molina, Andreas Paepcke, and Terry Winograd. Time as essence for photo browsing through personal digital libraries. In *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02)*, pages 326–335, Portland, Oregon, July 2002.
9. The ht://Dig Group. ht://Dig – Internet search engine software, 2003. See <http://www.htdig.org/>.
10. Wei Hao Lin and Alexander G. Hauptmann. A wearable digital library of personal conversations. In *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02)*, pages 277–278, Portland, Oregon, July 2002.
11. Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: detail and context smoothly integrated. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 173–176, New Orleans, Louisiana, 1991. ACM.
12. Catherine C. Marshall and Christine Ruotolo. Reading-in-the-small: A study of reading on small form factor devices. In *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02)*, pages 56–64, Portland, Oregon, July 2002.
13. William M. Newman, Christopher R. Dance, Alex S. Taylor, Stuart A. Taylor, Michael Taylor, and Tony Aldhous. Camworks: A video-based tool for efficient capture from paper source documents. In *Proceedings of the IEEE Conference on Multimedia Systems*, volume 2, pages 647–653, 1999.
14. Thomas A. Phelps and Robert Wilensky. The Multivalent browser: a platform for new ideas. In *Proceedings of the 2001 ACM Symposium on Document Engineering*, pages 58–67, Atlanta, Georgia, 2001. ACM. See also <http://www.cs.berkeley.edu/phelps/Multivalent/>.
15. The Apache Project. Jakarta Lucene Overview, 2003. See <http://jakarta.apache.org/lucene/docs/index.html>.