The RightPages Image-Based Electronic Library for Alerting and Browsing

Guy A. Story, Lawrence O'Gorman, David Fox,*
Louise Levy Schaper,* and H.V. Jagadish

AT&T Bell Laboratories

This first phase in a system to give users full on-line library services comprises electronic "stacks" of journal images whose utility is augmented through image analysis and processing methods.

The objective of an electronic library is to bring the library to the user. This entails more than transmitting materials like books, journals, or even nonprint media such as audio and video. It also means providing services like those a research librarian offers.

We cannot duplicate the traditional library environment in each user's home or office, but we can create an electronic analog of it. The RightPages electronic library prototype is the first phase of a long-range plan to do this. The prototype takes advantage of fast hardware, multimedia workstations, and broadband networks to process scientific and technical journals for users at AT&T Bell Laboratories and to offer a service that

(1) alerts them to the arrival of new journal articles matching their interest profiles,
(2) lets them immediately examine images of pages in the alerted articles and browse through other articles in the database, and
(3) enables them to order paper copies of any articles in the database.

We designed the user interface to appear as a conventional, albeit electronic, library. It shows "stacks" containing images of journal covers that users can view in a way analogous to viewing journal covers on library shelves. To examine the contents, the user selects a journal with a mouse, and the system displays an image of the table of contents. Users can select an article simply by pointing to it in the table of contents. They also have the option of perusing the issue page by page.

We maintained this traditional "look and feel" in the belief that libraries and journals have evolved over centuries into forms that are both comfortable and efficient. However, on top of these traditional structures, RightPages includes

*Since collaborating on this article, Fox has left AT&T to pursue his PhD at New York University, and Schaper has accepted a position at the University of California, San Diego.

0018-9162/92/0900-0017$03.00 © 1992 IEEE

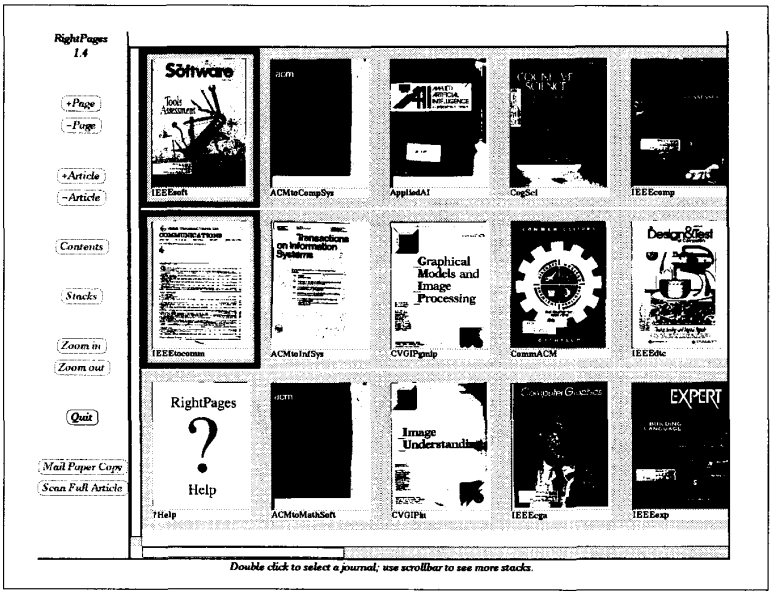

Figure 1. A portion of the stacks. The user views additional covers by sliding the lower horizontal bar to the right. The two journal covers at the top left are highlighted by contrasting borders, indicating profile matches.

features that are possible only through the computer. For example, users can point to a reference in the text — say, a reference to a figure on another page — and the system will immediately bring up the image of that page.

System overview

The RightPages prototype system has the following features:

- It obtains journal tables of contents and article pages in image form.
- It either uses publisher-provided ASCII text and page layout information or obtains these from the image through optical character recognition (OCR) and pattern-recognition techniques. This information is spatially associated with contents of the page images.
- It searches ASCII representations to determine matches with library profiles of individual user interests.
- It alerts users to these matches and gives pointers to the table-of-contents and article pages where matches occur. It also gives users access to all other journals and articles in the system.
- It lets users order full electronic or paper copies of articles for viewing.

The system runs on a local area network that connects one or more scanning stations, a centralized document database server, and multiple user stations running X Windows servers. The RightPages interface runs as an X Windows application on Sun workstations or X terminals.

New journal issues are entered every few days as they arrive. Most issues are scanned in. The current graphical scanning interface processes images at 300 dots per inch and one bit per pixel. We expect to upgrade the system to provide gray-scale images in the near future. For each periodical, the scanning operator indicates the title, volume, issue, date, and each article's page numbers as the issue is scanned.

Once a page image has been captured, the system processes it to remove noise and compresses it for efficient storage. Three "representations" are created for each page: a bitmap image of the page, the text content (including position information), and the location of logical fields, such as the table-of-contents entries. The representations are stored as separate Unix files, which are combined into a single (C++) object when the interface reads a page. The system matches text from the article pages with user keyword profiles and alerts users to matches via e-mail.

We recently completed the first phase of the service to a community of between 50 and 100 electrical engineering and computer science researchers at Bell Labs. In this article, we describe the system and aspects of the research that have gone into its implementation — in particular, the user interface and the processing methods that give this image-based electronic library its utility.

User interface

We built the RightPages interface with the InterViews graphical user interface development package,[1] which provides familiar window, mouse, and button features for display on X Windows interfaces. When in use, RightPages displays a control panel of buttons on the left side of the screen and banners for text messages along the top and bottom. Most of the screen, however, is an image area for journal covers, tables of contents, and article pages.

When the user first invokes the RightPages interface, the image area contains a grid arrangement of icons, called *stacks*. Each icon consists of a small image of a current journal cover with a label below (Figure 1). The user "activates" an icon with the mouse to access either another screen in a hierarchy of icons (for example, one showing all issues of that journal title) or the image of a particular issue's table-of-contents page.

Each user has a personal profile of keywords that the system matches with the text from the article pages. Matches generate *alerts*. After the system alerts a user to articles via electronic mail and the user, in turn, invokes the interface, the icon for each alerted journal is highlighted in the stacks and the alerted articles are highlighted in their tables of contents, as shown in Figure 2. The user can remove an alert by choosing it with the mouse and deleting it via a pop-up menu selection. In a similar fashion, the user can also add new alerts to mark articles for future reference.

This design is also meant to encourage browsing. Rather than having the user submit search requests and then showing a textual list of retrieved items, the default behavior starts at a view of the entire stacks, then gives graphical pointers to alerted material.

When the user activates an icon representing a particular issue of a journal, its table of contents is displayed on the

screen as a scanned image rather than ASCII text. This is true for article pages as well. While we do use OCR to obtain the text for searches, the OCR results are never visible to the user. The results are, however, spatially associated with the location of the text on each page image. This association is supported by page-layout analyses that segment and identify regions of a page (for instance, the title block, paragraphs, figures, and equations). Thus, we model each page as three planes of information: the image, OCR text, and page layout.

The main reason for displaying the image and not the ASCII is that most readers are already familiar with general graphical layout conventions, especially those used in journals they have read before. The displayed page images let readers rely on this familiarity when they scan for content. A second, practical reason is that OCR and page-layout-analysis results are not guaranteed to be flawless. Rather than displaying OCR errors to the users, the system sidesteps the problem by showing only the image and "hiding" the associated OCR text and layout planes.

When the reader chooses a table-of-contents entry by clicking in its region with the mouse, the entry is displayed in inverse video. Activating the entry brings up the first page of the article. The user can request a paper copy, which is sent through internal mail or picked up at a local printer.

The system maintains a log of the pages accessed, requests to scan arti-

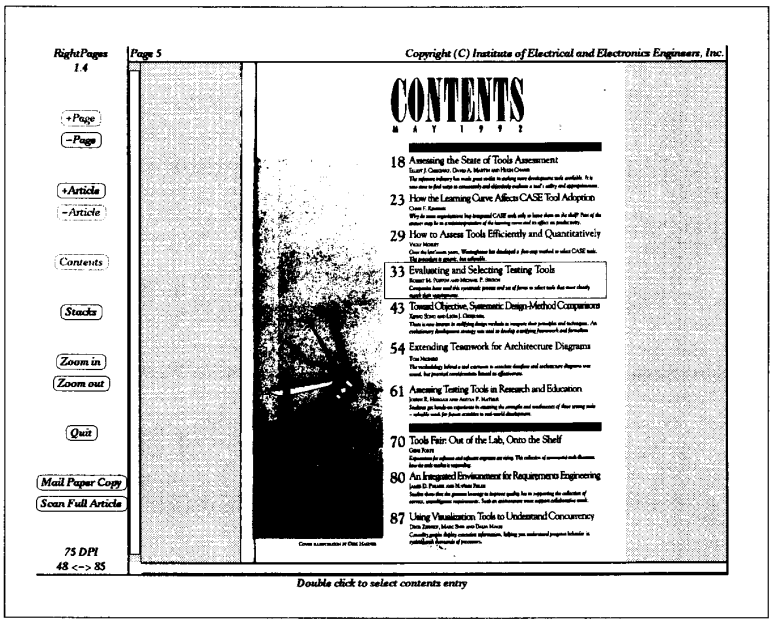

Figure 2. Table of contents with article entry highlighted.

Library of the future

For decades, improved information access via the "library of the future" has been the holy grail for far-thinking librarians, writers, and computer scientists. This century has produced many such visions. Examples include the world information monopoly presented in 1936 by H.G. Wells in *World Brain*[1]; Memex described in 1945 by Vannevar Bush in his classic article, "As We May Think"[2]; and F.W. Lancaster's paperless information system.[3]

Since the first library catalog, librarians have sought to provide increasingly accurate, easy-to-use retrieval tools. Card catalogs, controlled vocabularies, thesauri, abstracting and indexing services, and on-line and CD-ROM bibliographic databases are tools to facilitate information retrieval. They are precursors to electronic libraries. The past 10 years have introduced full-text databases such as Nexis and NewsNet, which are powerful but lack graphical expressiveness and a comfortable environment for retrieval and browsing — both goals of a truly electronic library.

Commercial image databases such as those offered on CD-ROM provide what is closest to the original material (that is, page images), but are limited by their reliance on a manual abstracting and indexing process to provide keyword searches.

Current experimental electronic library efforts include the CORE (Chemistry On-line Retrieval Experiment) project at Cornell University. This is a collaboration of the American Chemical Society (ACS), Bellcore, Chemical Abstracts Service, Cornell University, and OCLC,[4] where text and graphics from ACS journals are being stored and made available to users. Project Mercury at Carnegie Mellon University[5] is a plan to make information readily available at the desktop computer of each student and faculty member. The Bibliotheque Nationale is a national initiative in France to centralize and make library material electronically available.

References

1. H.G. Wells, *World Brain*, Doubleday, New York, 1938.
2. V. Bush, "As We May Think," *Atlantic Monthly*, July 1945, pp. 101-108.
3. F.W. Lancaster, *Toward Paperless Information Systems*, Academic Press, New York, 1978.
4. M. Lesk, "Full Text Retrieval with Graphics," *Bridging the Communication Gap (AGARD-CP-487)*, AGARD, Neuilly sur Seine, France, 1990, pp. 5/1-13.
5. W.Y. Arms et al., "The Mercury Electronic Library and Information System II, the First Three Years," Carnegie Mellon Univ. Mercury Tech. Reports Series, No. 6, 1992.

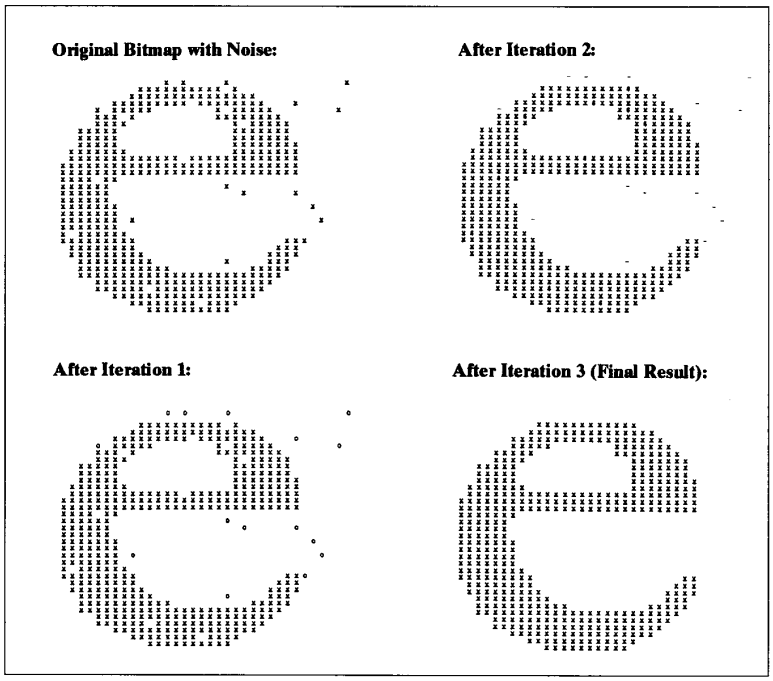

Figure 3. Applying kFill to an image of the letter "e." The sequence is top left, top right, bottom left, bottom right. Characters represent individual pixels: "X" is an On value, "-" represents an On value that has been changed to an Off, and "*" represents an Off value changed to an On.

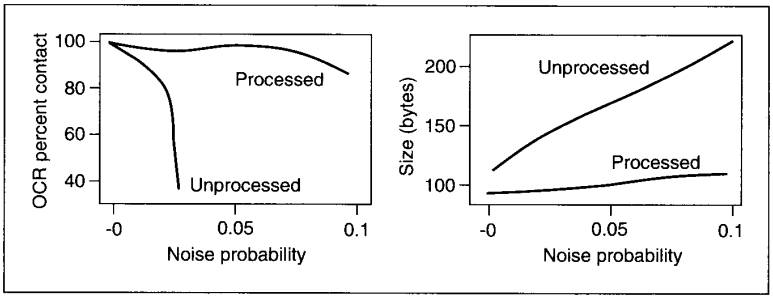

Figure 4. Effect of noise on OCR rate and compression size, with and without kFill processing.

cles, and requests to print articles for each user session. This log is used to fulfill those requests, generate statistical usage reports for the publishers, and guide development of the system.

Image and document processing

Input for page images requires multiple levels of descriptions. An image consists only of an array of data points representing pixel values. The system must process these pixels to obtain higher level descriptions.

Noise reduction is performed on each image after image capture. This process aids subsequent steps by reducing both the amount of data (thus, the processing time) and the artifacts that might otherwise impede the recognition of true features.

Document layout analysis and logical labeling are then performed on the cleaned images. These processes first segment the regions of each page into text blocks, then label them. For instance, just as a person reading a table of contents recognizes text fields containing the journal title, issue number, date, and each article field, so should the computer. This analysis also enables the implementation of a hypertext feature that links each article entry in the table of contents to the first page of that article.

OCR is performed in parallel with the layout analysis and labeling. While current OCR technology is highly accurate on clean printed material, errors still occur. Therefore, our OCR post-processing includes a method to recognize and correct many of the errors.

Because the size of the captured page image is larger than most monitors can fully display, it must be reduced. We do this by subsampling. To avoid the degradation in image quality that usually comes with size reduction, we have developed a filtering and subsampling technique that maintains maximum text readability.

In the following sections, we describe our document-processing methods in more detail. We include the processing times for 300-dpi images of 8.5 × 11-inch journal pages (typically 2,550 × 3,300 pixels) on a Sparcstation 2 machine that runs 24 million instructions per second.

Noise reduction. The first step after scanning is to reduce image noise. For text images in which the information is binary, salt-and-pepper noise (also called impulse and speckle noise, or just dirt) is the most prevalent. This noise appears as isolated pixels or pixel regions of On noise in Off backgrounds or Off noise (holes) within characters and other foreground On regions. The process of removing this is called "filling."

There are several causes of salt-and-pepper noise. A journal may have lightly colored background regions to highlight text, and the binarized result of these will generate a salt-and-pepper background. The limits of the scanning procedure also result in imperfections. For instance, when a journal is smaller than the scan region, the background outside the journal often contains this noise. The gutter region along the spine of thickly bound journals can also yield noise. Whatever its cause, we wish to reduce noise from this initial point of processing as much as possible to minimize its effect on the results of further analysis and compression steps.

The design of a noise-reduction filter involves a trade-off between noise re-

duction and maintenance of the signal. In this case, our signal is text and the objective is to maintain and enhance "readability." The kFill filter is designed to reduce salt-and-pepper noise while maintaining readability.[2] It is a conservative filter, erring on the side of maintaining text features versus reducing noise when those two conflict. To maintain text quality, the filter retains corners on text of 90 degrees or less, reducing rounding that occurs for other low-pass spatial filters. The filter has a k parameter (the "k" in kFill) that enables adjustment for different text sizes and image resolutions, thus supporting retention of small features such as periods and the stick ends of characters.

Filling operations are performed within a $k \times k$-sized window that is applied in raster-scan order, centered on each image pixel. This window comprises an inside $(k-2) \times (k-2)$ region, called the *core*, and the $4(k-1)$ pixels on the window perimeter, called the *neighborhood*. The filling operation entails setting all values of the core to On or Off, depending on pixel values in the neighborhood. The decision on whether or not to fill with On (Off) requires that all core values must be Off (On), and depends on two variables, determined from the neighborhood. For a fill-value equal to On (Off), the n variable is the number of On- (Off-) pixels in the neighborhood, and c is the number of connected groups of On-pixels in the neighborhood.

Filling occurs only when n is greater than a threshold, N, and c is equal to 1. The value of N is set as a function of window size, $N = 3k - 4$, to retain the text features described above. The stipulation that $c = 1$ ensures that filling does not change connectivity (that is, does not join two letters together or separate two parts of the same connected letter).

Noise reduction is performed iteratively on the image. Each iteration consists of two subiterations, one performing On-fills and the other Off-fills. When no filling occurs on two consecutive subiterations, the process stops automatically.

Figure 3 shows the results of applying kFill to random salt-and-pepper noise in a letter *e*. Depending on the amount of noise, the application of kFill in the preprocessing step may significantly improve the results of compression and OCR. Figure 4 shows how these results

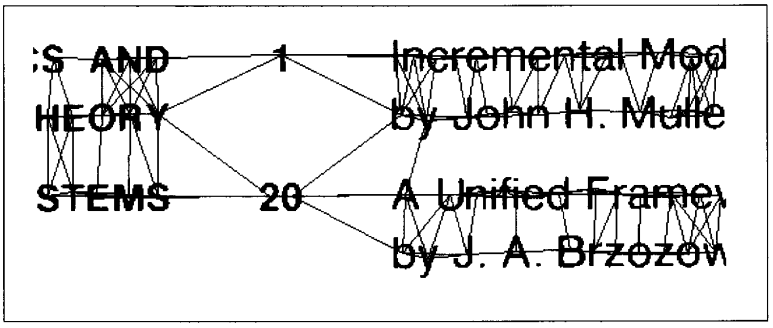

Figure 5. Intermediate results of docstrum analysis show the nearest neighbor connections on a text portion taken from the table of contents shown in Figure 7 ($k = 5$ in this example).

are affected by noise, both with and without the application of the filter. The time required for this operation varies with the amount of noise; a typical time, however, is between one and two minutes for a one-page image.

Document layout analysis. The results of OCR only partially describe each page. To interpret the text fully it is also important to have information on where it occurs within the page layout. Page-layout-analysis methods segment each page into logical entities and label them as title, authors, headings, subheadings, paragraphs, equations, words, etc. The RightPages prototype segments the page by using the "document spectrum" (or simply "docstrum") method.[3] The docstrum outputs segmented blocks, which are labeled using a two-dimensional grammar that establishes rules determined by particular, journal-dependent page formats.

The docstrum represents page layout via groupings of its lowest level primitives, namely, nearest neighbor pairings between character elements of the page (Figure 5). Each nearest neighbor pair is described by a tuple, the distance and angle between centroids of elements of the pair. For example, two characters in the same word form a nearest neighbor pair whose distance is relatively small and whose angle is close to the angle of its resident text line. For each page element, k nearest neighbors are found, where k is usually between three and five. Therefore, for $k = 5$, a character might make three pairings within a word and across word boundaries within the same line, and two pairings with characters on upper and lower lines. The interline pairings have larger distances than intraword pairs and their angles are approximately 90 degrees apart.

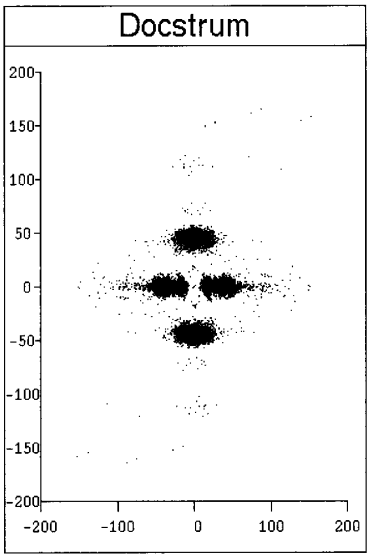

Figure 6. The docstrum plot corresponding to the page shown in Figure 7.

The docstrum is the plot of distance and angle for all k nearest neighbor coordinate pairs on a page (Figure 6). The docstrum is a polar plot with origin at the center; radial distance from the center is the nearest neighbor distance, and counter-clockwise angle from the horizontal is the nearest neighbor angle. The docstrum is so termed because of its similarity in appearance to the 2D power spectrum and because of its analogous utility in globally describing an image — in this case, a document image.

The prototype system determines page features by examining the docstrum plot. A symmetric pair of clusters will exist

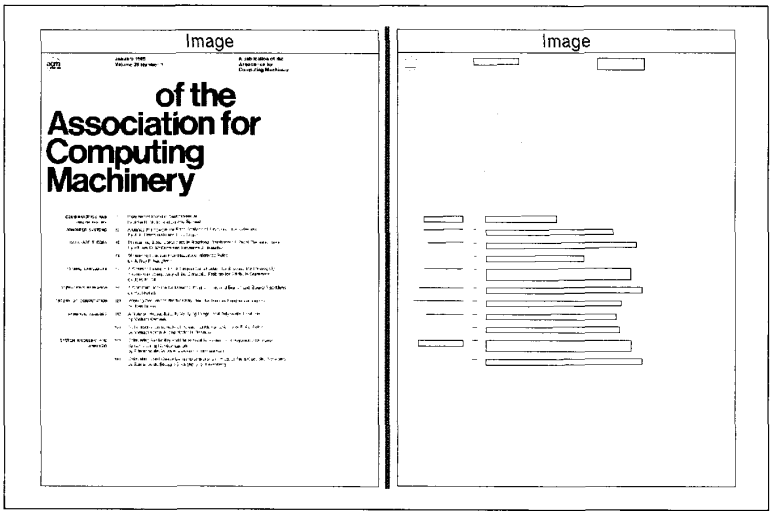

Figure 7. Final results of docstrum analysis show lines drawn through character centroids within text blocks and page number blocks. The larger characters of the journal title have been filtered out on the basis of their size. (The word "Journal" in the title was lost in binarization because it is white on a colored background, versus the rest of the text, which is black.)

for both intraword and interword character spacing, and at 90 degrees to these for interline character spacing. The radial distance from the origin gives the average spacing for each of these features. The angle that this cross-pattern of features deviates from the zero degrees of an upright page gives the skew angle, or orientation of the text lines.

To ensure the separability of clusters and to maximize the precision of locating their centers, the system performs clustering and centroid measurement, first, by integrating the docstrum over the distance variable to yield the angle histogram. The peak of the angle histogram is usually found easily and indicates the skew angle. The docstrum is then integrated over the angle variable to yield the histogram for the nearest neighbor distance. Knowing the skew angle, the system can determine character and line spacing from this histogram. Average intercharacter spacing is at the lower peak on the nearest-neighbor-distance histogram for pairs within an angular tolerance to the skew angle. Line spacing is found at the higher peak in the nearest-neighbor-distance histogram for angles within an angular tolerance of the perpendicular to the skew angle.

From this docstrum information, text blocks are determined in a bottom-up manner. Characters are joined to words using the skew angle and intercharacter spacing. Words are joined into lines. Regression lines are fit through character centroids of each text line, and the average of these over the page is a precise estimation of skew. Finally, a bottom-up line-merging technique forms blocks of text lines. Figure 7 shows the resulting blocks of text from this docstrum analysis. The docstrum steps described here require 8 to 15 seconds of processing time for table-of-contents pages.

Logical labeling of page parts. Because the user can access information by pointing to specific blocks on a page image, the system must determine what these blocks represent. For instance, on the title page of a journal, we would like to identify the date, volume number, and page number for a particular article. Furthermore, we would like to build logical groupings of related items, such as title, author name, and page number for a particular article. We achieve this by building a parse tree for the page.

This parse tree is constructed according to a grammar that is specified for each different journal type. The grammar uses information about the relative positions of the blocks to determine their semantic nature. Standard parsing theory has been developed for grammars on symbol strings in which a single sequencing relationship exists between the language symbols. In our case, each block is considered a terminal symbol, so there are two relationships of interest: a left-right relationship and an above-below relationship.

We have developed a theory of *partial order grammars* for this purpose. Such a grammar can be defined over any set of symbols that have one or more partial orders defined on them, subject to a few technical conditions easily satisfied by most page layouts. In our case, the left-right relationship is a partial order, since for any pair of blocks, A and B, we may have A to the left of B, B to the left of A, or neither (for instance, if A is directly below B). Similarly, the above-below relationship is a partial order. For partial order grammars, it is possible to develop polynomial time LR-parsers similar to those for string grammars. We have written a tool called pocc, similar to the Unix utility called yacc, that takes a grammar specification and generates a parser for it.

Thus, we can specify a grammar for each different type of page in a particular journal title. In this initial implementation, we do this for the different table-of-contents formats. This specification is converted into a parser through pocc. Thus, the parser is used to determine the structure of each new table-of-contents page, using a grammar specific to the particular journal title.

The parse tree provides us with a logical structure for the information on the page. For instance, there is an association between the title of an article and its page number (since these blocks will be grouped together to form an article block). This association is exploited to determine the page number automatically when a user clicks on the title of an article and thereafter to retrieve and display the appropriate page. This logical structure is also valuable in creating a structured object from a page. This composite object consists of smaller objects and fields with certain values. Such objects can be stored in an object-oriented database, where they can be retrieved and manipulated effectively.

Text processing. As seen in previous sections, a major goal is to add functionality to the bitmapped page images presented to the user. In addition to the facilities enabled by knowledge of the page's logical structure, two functions require machine-readable text. One is

the alerting service, which matches text against user profiles. The second is an on-line interactive text search facility that is under development.

The system generates text from the page images by first performing OCR (currently with a commercial system) and then postprocessing the OCR output in an attempt to remove any remaining errors. The OCR process, in addition to providing ASCII characters, gives information about the position of the text on the page. The system uses this information to spatially correlate search hits with the page image. Since the user always sees the page image and not the OCR output, we do not require pristine output from the recognition process. Instead, we wish only to improve its accuracy and in turn increase the number of hits found in profile matching and in on-line search.

The output of the OCR is postprocessed by the JSB (Jones, Story, and Ballard) system.[4] This system integrates a Bayesian treatment of predictions from a variety of knowledge sources. The JSB knowledge sources can be broadly classified as those that characterize the recognition device and those that characterize the documents. The document characterizations include both explicit frequencies and signature tables for the letter and word n-grams found in texts representative of the journals in the database. Device knowledge is expressed as a set of character rewrite rules with frequencies and is referred to as the DM (for device mapping). The DM describes the recognition errors made by the OCR device during training.

The postprocessing algorithm operates in three phases. Phase 1 builds a list of candidates for each word (space-delimited string) of the input by considering each word in isolation. Phase 2 merges words to undo the effect of apparent word-splitting errors. Phase 3 reorders candidates by considering their context (that is, adjacent words). During this final phase, additional candidates may also be proposed. These phases, while processing progressively higher level representations, are not strictly serial, since proposed analyses are fed back into earlier phases.

For each space-delimited string, Y, output by the OCR process, phase 1 proposes one or more candidate strings, X_i, with associated probabilities, $p(X_i|Y)$. Phase 1 estimates are derived from the DM statistics and the a priori probabilities of the X_i. These a priori values are based on word frequency for known words and character digrams for unknown words. Analytically, phase 1 computes the following (normalized) probabilities:

$$p(X_i|Y) = \frac{p(X_i)\ p(X_i \rightarrow Y)}{\sum_{j=1}^{q} p(X_j)\ p(X_j \rightarrow Y)}$$

The probability $p(X_i \rightarrow Y)$ is the likelihood that the Y string resulted from X_i, given the DM statistics; it is the product of the probabilities associated with the rewrite rules applied to get X_i from Y. If the Y string is not in the dictionary or if dictionary words are being reconsidered, then we enter it as the first candidate string and begin the search for known words.

In phase 3, we reorder the candidates for each string based on word digram probabilities. In addition to reordering the X candidates, word digram conditional probabilities also suggest additional X candidates, which are scored by phase 1 and added to the list before renormalization.

Display processing. A standard image resolution for today's scanners is 300 dpi. For an 8.5 × 11-inch document page, the scanned size is then 2,550 × 3,300 pixels. Bitmap monitor sizes vary greatly, but very few will display a full page at this resolution. To fit the image on the monitor, it must be subsampled by the smaller ratio of dimensions (for this example, by 1,280/3,300, or 39 percent). Though a reduction of this size will clearly reduce image quality, we can minimize its effect by applying a low-pass filter on the original image to reduce high-frequency content above the new sampling frequency.

A common image-filtering method is to obtain each subsampled pixel as the average in a $k \times k$-sized region around the corresponding pixel in the input image. For gray-scale output, this result is normalized by the maximum gray-scale value. For binary output, this result is compared with a threshold and set to a value of 1 or 0.

Filter parameters that maintain maximum "readability," or text quality, in the subsampled image are presented in O'Gorman.[2] A simple method is given

Copyright compliance efforts

Because established standard methods for complying with copyright in the electronic environment do not exist, the RightPages service is serving as a testbed for compliance processes, as well as a barometer of scientific and technical publishers' readiness to license their publications for a networked, image-based electronic library. Preliminary results indicate that an increasing number of publishers are strategically ready to authorize their works for electronic distribution, but slow to make agreements because they lack a familiar legal and administrative infrastructure.[1]

Currently, the RightPages service includes 64 journals, representing 10 publishers. We have made site-license arrangements with these publishers authorizing unlimited use within specified locations and parameters. Building on the process for authorizing paper photocopying, we entered into an agreement with the Copyright Clearance Center to manage the process of obtaining authorizations. Where the CCC has been unsuccessful in obtaining authorizations, we have attempted to obtain them from the publishers directly. There are three of these bilateral agreements at the time of this writing.

Part of the agreement requires semi-annual reports of usage. These include information from each user access of the system, such as access time, issues viewed, articles ordered, and interface features used. The publishers will use this information to assess this trial in electronic journal distribution, and we will use it to assess system performance, utility, and features.

Reference

1. L.L. Schaper and W.T. Kawecki, "Inwards Compliance: How One Global Corporation Complies with Copyright Law," *Online*, Vol. 15, No. 2, Mar. 1991, pp. 15-21.

```
# RIGHTPAGES PROFILE: file contains keywords, phrases, and
# journal names upon which RightPages search is done.

(machine OR computer) vision
library WITHIN(10) electronic
(document OR image) AND processing
pattern recognition AND NOT (JOURNAL Pattern Recognition OR
  JOURNAL \
IEEE Transactions on Pattern Analysis and Machine Intelligence)

JOURNAL Pattern Recognition
JOURNAL IEEE Transactions on Pattern Analysis and Machine
  Intelligence
JOURNAL Proceedings of the IEEE
```

Figure 8. A sample user profile.

for subsampling by noninteger rates so that subsampled images can fit into the maximum screen space available. Also, a method is described that adaptively reduces an image size; this approach reduces regions of white space and other "low-information" rows and columns, while maintaining quality in the higher information text. The filtering and subsampling requires about 10 seconds per page for reduction to 33 percent.

Document database

The result of the processing steps is a document in which journal pages are C++ objects, each of which contains multiple representations that are themselves objects. Conversion to an object-oriented database is planned to speed access and to improve programming ease and versatility. In the meantime, the various page representations are stored as Unix files.

The first kind of representation contains page images in a variety of resolutions and in depths of one and eight planes to accommodate the various monitors in use. The second representation type contains the results of OCR on the page (that is, the ASCII text with positional information). The third is the page *logical content*. A logical content file consists of a set of labeled rectangular regions, possibly hierarchically arranged. Each region can optionally link to an entire page (represented by its page attributes) or to another region. In the latter case, the second region can be on another page and is represented by its label and its page's attributes. This kind of link is used to connect article entries on the table-of-contents pages to the article pages.

Profile matching

Each user has a personal profile (see Figure 8), a file that resides in his or her home directory and indicates technical interests as well as parameters for customized interaction. The personal profile regulates alerting of new material as it arrives in the system. Alerting is done by sending e-mail to the user, indicating the number of hits and the journals in which they occur and giving instructions for running the system. The profile is composed of two distinct sections. One contains a list of journal names, called a subscription list, and the other contains Boolean combinations of keywords. For the subscription list, the user is alerted when a particular journal enters the system. This lets the user peruse the table of contents and articles much the same as if a conventional paper subscription had been received.

The keyword section is more selective, alerting the user only to articles that match keyword combinations. The keyword profile is meant to cover the user's secondary journals — journals that would be surveyed only irregularly for specific topics — and tertiary journals that the user would rarely read, but that occasionally include an article related to the user's interests. The RightPages prototype uses the slimsearch package to search keywords.[5] The slimsearch package produces an inverted index of terms for efficient searching.

The Boolean operators and syntax of the profile for keyword expressions are *w*1 AND *w*2, *w*1 OR *w*2, *w*1 WITHIN(*n*) *w*2, *w*1 *w*2, NOT *w*1. The *w*1 and *w*2 are keyword operands and can be single words or Boolean combinations of words. The AND operator forces a match if a search space contains two keyword operands. The search space is the full first page of an article. The OR operator forces a match if the search space contains either or both keyword operands. The WITHIN(*n*) operator matches if both keyword operands are within *n* words of each other. A space between expressions is equivalent to a WITHIN(1) connection. Finally, a match may be negated if the search space contains the expression following the unary NOT operator. A statement may consist of combinations of Boolean expressions.

Besides keyword expressions, journal names can also be used in the profile following the unary Journal operator. When journal names are used, they mean that a match is effected if it occurs in text from that journal. Two journal names can be joined only by the OR operator. A journal expression can be joined to keyword expressions by AND or by OR and can be negated by NOT. For example, *w*1 AND NOT (Journal journal_X OR Journal journal_Y) means that a match occurs if the keyword expression, *w*1, occurs in any journals excluding journal_X and journal_Y. The order of binding precedence for both keyword and journal operators is (...), Journal, blank, WITHIN(*n*), NOT, AND, OR.

Phase 1 results

We scan new journal issues and requested articles into the system during the day to undergo the bulk of their processing at night. Text and image processing are performed on a Sparcstation 2 with 32 megabytes of main memory and 100 megabytes of magnetic disk memory for swapping.

Though use of the system is increasing as the number of journals and users increases and as users become more comfortable with it, recent usage statistics averaged over a month show the following approximate data. Three issues are entered per working day, 15

pages per issue. (Note that during the first phase, we scan only the cover, table of contents, and first page of each article. Upon user request, we provide the full article in either paper or electronic form.) Two articles are requested per working day at 12 pages per article. These numbers total 1,380 pages per month and 16,560 pages per year.

The total storage size for the compressed images and other planes of text and layout information for each page is approximately one-fourth of the original bitmap size — that is, about 250 kilobytes per page. At current usage, this would result in 4.14 gigabytes of data per year. The daily processing time varies widely with the number of pages and amount of text, noise, matches, etc. However, all processing is usually complete in under two hours.

Because the system performs image and text processing off line, system performance at the interface level is reasonably fast. Table-of-contents pages and article pages can be viewed with less than a second delay. When a user requests a full electronic copy of an article, the request is logged, usually scanned the next day, and available to the user after the night's processing. Users who want full hard copy of an article can print it at their local printer with the same delay. Once an article has been requested, any user can view it without scanning delay.

The system has been operating on a small test scale since May 1991, and with wider availability at Bell Labs since February 1992. Upon writing this article, only statistics from the first phase were available. This phase had about 40 registered users. Of these, about half used the system at a frequency of once a week or more. A similar number of requests for paper and electronic copies had been made, about 22 each. There had been 103 journals perused and 1,440 articles read. Of the articles, 567 were unique. The average time that each page was on a user's screen was two minutes.

Though the user statistics are not yet definitive, we have received many user comments. In general, they have been positive. Some users have reported that the alerting and browsing features have led them to articles that proved important to their research in journals they normally do not read. The negative comments have come mostly from users who are uncomfortable reading the scanned text on the screen.

Planned additions to the RightPages system fall roughly into three categories: new interface features, enhancements of the underlying architecture, and extensions to other kinds of documents.

In the first category, we have demonstrated a prototype that lets users attach personal notes to or "draw" on the journal pages. The user can control both author and reader lists for these annotations. Such features should encourage on-line "discussion" and alerting among colleagues. In this prototype, the user can also highlight a region of a page image and have a text-to-speech system "read" aloud the text in that region. This feature is, of course, facilitated by the correlation of the ASCII text representation of the page with the bitmap used for display. A user who has been alerted to a new profile-matching article might have the introduction of the article read to him or her while busy with other activities in the office.

Interface enhancements in the planning stage include audio annotations, the ability to group the database contents into user-defined folders and notebooks, and graphical aids for indicating the user's traversal through the database during a session.

A number of enhancements to the underlying architecture should improve performance or enrich the database contents. One such enhancement will provide a text-search facility that is tolerant of OCR errors, using techniques derived from the JSB OCR postprocessor. Other enhancements will require the results of ongoing research. A goal of the layout analysis is to identify and label as many logical structures on the pages as possible. This would, for example, allow a user to restrict search or display to just the title and abstract or just the figures. In addition, the database provides a testbed for experimenting with and evaluating different text and document classification methods. As the database grows, such techniques will be critical in helping users find and manage information. Finally, we are investigating networking architectures for providing a RightPages-like service to wide area networks.

Libraries are not restricted to text-based materials, and we plan to apply the ideas used in representing journal pages to other media, including pictures, audio, and video. In the long run, of course, more and more published material will be prepared and distributed in electronic form. We intend to support these purely electronic documents, which are represented in a variety of markup languages. ■

References

1. M.A. Linton, P.R. Calder, and J.M. Vlissides, "InterViews: A C++ Graphical Interface Toolkit," Tech. Report CSL-TR-88-358, Stanford Univ., July 1988.
2. L. O'Gorman, "Image and Document Processing Techniques for the RightPages Electronic Library System," *Proc. 11th IAPR Int'l Conf. Pattern Recognition, Vol. II*, IEEE CS Press, Los Alamitos, Calif., Order No. 2915, 1992, pp. 260-263.
3. L. O'Gorman, "The Document Spectrum for Page Layout Analysis," accepted by the IAPR Int'l Workshop on Structural and Syntactic Pattern Recognition, Bern, Switzerland, Aug. 1992.
4. M.A. Jones, G.A. Story, and B. Ballard, "Integrating Multiple Knowledge Sources in a Bayesian OCR Post-Processor," *Proc. First Int'l Conf. Document Analysis and Recognition*, INRIA, Paris, France, 1991, pp. 925-933.
5. R.K. Waldstein, "Slimmer — A Unix-System-Based Information Retrieval System, *Reference Services Rev.*, Vol. 16, No. 1-2, 1988, pp. 69-76.

Guy A. Story has been on the technical staff of the Computing Systems Research Laboratory at AT&T Bell Laboratories, Murray Hill, New Jersey, since 1985. His current research is interactive multimedia systems and user interfaces, although he also has interests in natural-language processing and knowledge representation.

Story received a BS in electrical engineering from Rice University in 1974, and completed his MS in computer science from New York University in 1980. He is a member of the IEEE Computer Society and the ACM.

Lawrence O'Gorman has worked in the Computing Systems Research Laboratory at AT&T Bell Laboratories, Murray Hill, New Jersey, since 1984. His research interests include image processing, pattern recognition, document image analysis, and machine vision precision. He has lately been involved in the design of the RightPages electronic library project.

O'Gorman received the BASc degree from the University of Ottawa in 1978, the MS degree from the University of Washington in 1980, and the PhD degree from Carnegie Mellon University in 1983, all in electrical engineering.

Readers can contact Guy A. Story at AT&T Bell Laboratories, Room No. 3C-420A, PO Box 636, Murray Hill, NJ 07974-0636; e-mail story@research.att.com.

David Fox is a PhD student at the Courant Institute of Mathematical Sciences, New York University. From 1983 to 1991, he worked in the Computing Systems Research Laboratory at AT&T Bell Laboratories, Murray Hill, New Jersey. His research interests include multiscale multimedia interactive systems, computer cartography, issues of privacy in computer communication, and the process of computer programming.

Fox received his BSc degree from Brown University in 1983, and the MS degree from Columbia University in 1989.

Louise Levy Schaper is head of the Systems Department in the University Library at the University of California, San Diego. She manages the library's information technology infrastructure, including data/voice transport, library and office automation applications, and a campus information system. She was previously with AT&T Bell Laboratories, where she developed and marketed electronic and paper-based information services.

Schaper received MSW and MLS degrees from Syracuse University.

H.V. Jagadish is with the Computing Systems Research Laboratory at AT&T Bell Laboratories, Murray Hill, New Jersey. His research interests encompass various aspects of data storage, retrieval, and use. In particular, he is working on object-oriented data models, active databases, and the nontraditional use of databases, especially with respect to image processing systems. He is also interested in the architecture of computer systems oriented toward data manipulation rather than computing.

Jagadish received his PhD from Stanford University in 1985.