Image and Document Processing Techniques for the RightPages Electronic Library System

Lawrence O'Gorman

AT&T Bell Laboratories, Room 3D-455
Murray Hill, NJ, 07974, USA
email: log@research.att.com

Abstract

This paper describes some of the document processing techniques used in the RightPages electronic library system. Since the system deals with scanned images of document pages, these techniques are critical to the use and appearance of the system. We describe three techniques here. One is for noise reduction from binary document pages to improve page appearance and subsequent optical character recognition and compression. The second is for subsampling the text image to fit on the computer screen while maintaining readability. The third is a document layout analysis technique to determine text blocks. Due to the short length of this paper, more detail, discussion, and results on the techniques presented here are left for the given references.

1. Introduction — The RightPages System

The RightPages system is an image-based electronic library that is currently in research, as well as in use, at Bell Laboratories. The system was begun as a testbed to study many of the aspects associated with the electronic library, including document analysis, user-interfaces, multi-media databases, and electronic library usage. In this paper, we give an overview of the library system, and give more detailed description of some of the work in image and document processing.

The objective of the RightPages project is to bring an electronic analog of the library to a user's desktop computer. This facility includes both delivery of library material as well as a personal — albeit electronic — research librarian to alert the user to new library material matching the user's interest profile. One of the most characteristic features of the system is the same "look and feel" of traditional libraries. However this familiar basis is augmented with some of the features that computer imaging, networking, and databases enable. For instance, a user may browse the stacks, viewing journal covers, and choosing one to read based on the automatic alerting or just whatever catches the eye. When an article selection is made, its appearance is exactly as in the original — it is simply a scanned image of the original, but spatially associated with the scanned image are results of optical character recognition (OCR) and page layout analysis to enable searching of the article contents. We currently have copyright permission to offer about 60 technical journals, representing 10 publishers. The system is being used by a community of computer researchers at Bell Labs, and is being studied for usage patterns and for further modifications and improvements. For more detail on this system, see [1].

2. Image and Document Processing Techniques

In this section, we describe some of the image processing and document analysis techniques used in the RightPages system. Though other methods are used in the system, we choose to describe those that are novel or particularly applicable to the image-based electronic library. Where processing times are given, these are for a Sun SparcStation 2 workstation.

2.1 Noise Reduction

The first step after scanning is the reduction of noise in the image. For text images in which the information is binary, salt-and-pepper noise 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". It is important for a number of the techniques following preprocessing that the noise be reduced as much as possible. For storage of the images, noise reduction reduces the storage size. More importantly, for the recognition processes of OCR and page layout analysis, the application of noise reduction can mean a substantial improvement in recognition results. In this section, a filter is described to reduce salt-and-pepper noise with the special characteristic that it is designed for text images to retain their quality and maximum "readability."

The *kFill* filter is designed to reduce salt-and-pepper noise while maintaining readability. 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° 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 enabling 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 dependent upon 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 three 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, the c variable is the number of connected

0-8186-2915-0/92 $3.00 © 1992 IEEE

groups of ON-pixels in the neighborhood, and the r variable is the number of corner pixels that are ON (OFF). (The first two variables comprise a subset of those used in the $k \times k$ thinning algorithm [2], which also uses a variably sized window for processing. The window geometry and variables are described in more detail in that reference.) Filling occurs when the following conditions are met:

$$(c = 1) \wedge [(n > 3k - 4) \vee (n = 3k - 4) \wedge r = 2].$$

The conditions on n and r are set as functions of the window size k such that the text features described above are retained. The stipulation that $c = 1$ ensures that filling does not change connectivity (i.e. 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 sub-iterations, one performing ON-fills, and the other OFF-fills. When no filling occurs on two consecutive sub-iterations, the process stops automatically.

Depending on the amount of noise, the application of *kFill* in the preprocessing step may result in a significant improvement in the results of compression and OCR. Figure 1 shows how these results are affected by noise, both with and without application of the filter.

The time required for this processing varies with the amount of text and noise, however a typical time is around a minute to two minutes for a 2550×3300 pixel page image of a fairly dense page such as for IEEE Transaction articles. Our implementation performs run-length encoding of the input raster image on the first iteration, then subsequent iterations are more efficiently processed. Since each iteration is still costly, and since much less noise is removed on later iterations, a practical use of the method is to set a maximum number of iterations (through empirical experience with particular images), or to stop the iterations when noise is reduced a chosen percentage. With these speed enhancements, our processing typically takes about 40 seconds.

2.2 Display Processing

A standard image resolution for today's scanners is 300 dpi (dots per inch). For an 8.5×11 inch document page, the scanned size is then 2550×3300 pixels. Bitmap monitor sizes vary greatly, but very few will display a full page at this resolution. (For instance, the screen that this paper is being typed on is 1600×1280, considered fairly high resolution.) To fit the image on the monitor, it must be subsampled by the smaller ratio of dimensions, (for this example, by 1280/3300, or by 39%, thus the subsampling rate, r is 100/39 = 2.56). Though it is clear that a reduction of this size will reduce the quality of the image, we can minimize this by digital signal processing techniques. Standard practice before reducing the size of a digital signal — or subsampling — is to apply a low-pass filter on the original image to reduce high frequency content above the new sampling frequency. In this section, we describe filtering and subsampling techniques designed to maintain maximum readability of text.

There are a number of different subsampling cases, depending on the available quantization of the initially scanned image and the desired quantization of the final display image. The cases that we deal with are: binary-to-binary, binary-to-gray, and a spatially adaptive binary-to-binary technique. The first two methods are straightforward and not new. However, we have spent some time to examine the results of the methods for different parameters, and these empirically determined "best" parameters should prove useful to other researchers or practitioners. For more detail on these techniques, see [3].

For both non-adaptive methods, we first convolve the image with a Gaussian-weighted filter window to reduce high frequency. We use the minimum window size such that all pixels affect the output. For a subsample rate, r, the window width is $k = 2r - 1$. We set the variance of the Gaussian such that the coefficient weighting at distance $(k - 1) / 2$ from the center of the filter is $\exp(-2)$. That is, the variance is $\sigma = (k - 1) / 4$, and the unnormalized filter coefficients are $\exp(-(x / \sigma)^2 / 2)$, where x is the distance from the center of the filter.

For the binary-to-binary method, the Gaussian result at each pixel is compared with a threshold that is a chosen percentage of the maximum obtainable result. We have found that 40% gives good results for text images.

For the binary-to-gray method, the Gaussian result is divided by a contrast enhancement parameter, which is expressed as a percentage of the maximum obtainable result, then is normalized to a value from zero to 255 (the maximum intensity value). The contrast parameter that we have found to be effective is 70%. That is, if the output of the filter at a pixel is 70% or more than the maximum attainable result, it is set to 255; and lower outputs are scaled proportionately.

We do make one small modification of the output for a certain topological condition, that of an endline. The purpose of this is to reduce attenuation or shortening of endlines, such as the vertical line on a "b" or horizontal lines on "E". The rationale is that the endline should be treated the same way as a continued line (through the filter window) is treated; it is not treated this way because the window size is "too large" for the smaller endline feature. Therefore, when an endline is recognized, the filter output sum is augmented by the sum of the filter coefficients times the ON-valued border pixels within the window. This effectively doubles the effect of the border pixels and changes the output sum to that if the endline were really a continuous line. After this, the result is handled in the same way as described above.

Because the output image is subsampled from the original, we need not apply the filter to every input pixel. Instead, the filter is applied at the location of every output pixel. For instance, if the subsampling rate is 3, then the filter need only be applied to pixels on every 3rd row and column. This technique is not limited to integral subsampling rates. If it is desired to reduce the image size to 40%, then the subsampling rate is 2.5, and the filter is centered at locations ever 2.5 rows and columns. This requires coefficient values to be defined at subpixel intervals. In our implementation, we supersample the Gaussian filter window at a resolution 8 times that of the initial image rate to obtain adequate subpixel precision of the filter coefficients. As examples of processing speed, reduction of an image to 50% and 28% take approximately 17 seconds and 9 seconds respectively.

The spatially adaptive binary-to-binary subsampling technique is akin to silence removal techniques that are used to speed up recorded speech. Instead of silence recognition, we recognize "low-information" rows and columns in the image. Simple examples of this are the margins of most pages. However the low-information measures should be more

sophisticated than simply recognizing blank space, such that it may reduce image sizes while dealing with noise, border lines, and ruled paper for instance.

At each pixel, a measure of information is found within a 1×3 sized template for row calculations, and 3×1 template for column calculations. The empirically determined measures are shown in Table 1, along with the rationale behind the given information measure for each center pixel. Summations of these measures are made for rows and columns and those with low sums are delegates for removal.

Template	Information Weight	Reason
010	10	to maintain disconnectivity
101	10	same as above
011	5	to maintain an edge
110	5	same as above
111	2	to maintain shape
001	1	to maintain spacing
100	1	same as above
000	0	not important

TABLE 1. Information weights of pixel templates. The measure is made of the center pixel based on its two horizontal or vertical neighbors.

We have found this adaptive method to be useful only when applied fairly conservatively. That is, if the amount of attempted size reduction is too aggressive, the deletion of too many rows and columns through text lines will degrade the output text. However, even dense pages as found in IEEE Transaction articles can be reduced up to about 20% without significant text degradation using this method. Processing time to do this is about 40 seconds.

2.3 Page Layout Analysis

OCR is performed on the page images to yield text upon which to search, however the results of OCR only partially describe each page. To interpret the text fully it is important to have information on where it occurs within the layout of the page as well. Page layout analysis methods are used to perform this task, specifically to segment each page into logical entities and label them as title, authors, headings, subheadings, paragraphs, equations, words, etc.. For the RightPages system, segmentation is performed using the *document spectrum* or simply *docstrum*. For more than the brief description of the docstrum that is given here, see [4].

The docstrum describes page layout via groupings of its lowest-level primitives, nearest-neighbor pairings between character (and other) elements of the page. 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. Usually k nearest-neighbors are found for each page element, where $k \approx 5$. Therefore a character might make two or three pairings within a word and across word boundaries within the same line, and pairings also with characters on upper and lower lines. The inter-line pairings have larger distances than intra-line pairs, and angles approximately 90° apart. The docstrum is the plot of distance and angle for all k nearest-neighbor pairs on a page. It is a polar plot with origin at the center, radial distance from this the nearest-neighbor distance, and counter-clockwise angle from the horizontal the nearest-neighbor angle. The docstrum is so termed because of its similarity in appearance to the 2-D power spectrum and because of its analogous utility in globally describing an image — in this case a document image. The nearest-neighbor connections are shown for a portion of text from a table of contents page in Figures 2.

Page features can be determined by examining the docstrum plot. A symmetric pair of clusters will exist for both intra-word and inter-word character spacing, and at 90° to these for inter-line 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 that for an upright page (0°,180°), gives the skew angle.

From this docstrum information, text blocks are determined in a bottom-up manner. Characters are joined to words using skew angle and inter-character 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, blocks are merged from text lines on the basis of three features: proximity, parallelism, and overlap. The resulting blocks of text from this docstrum analysis are shown in Figure 3 for a table of contents page of the Journal of the ACM.

REFERENCES

1. G.A. Story, L. O'Gorman, D. Fox, L. Schaper, and H.V. Jagadish, "The RightPages image-based electronic library for alerting and browsing", IEEE Computer, 1992 (accepted for publication).
2. L. O'Gorman, "k x k Thinning", Computer Vision, Graphics, and Image Processing, Vol. 51, pp. 195-215, 1990.
3. L. O'Gorman, G.A. Story, "Subsampling text images", 1st Int. Conf. on Document Analysis and Recognition, St. Malo, France, pp. 219-227, 1991.
4. L. O'Gorman, "The document spectrum for page layout analysis", IAPR Workshop on Structural and Syntactic Pattern Recognition (SSPR), Bern, Switzerland, Aug., 1992. (or from the author)

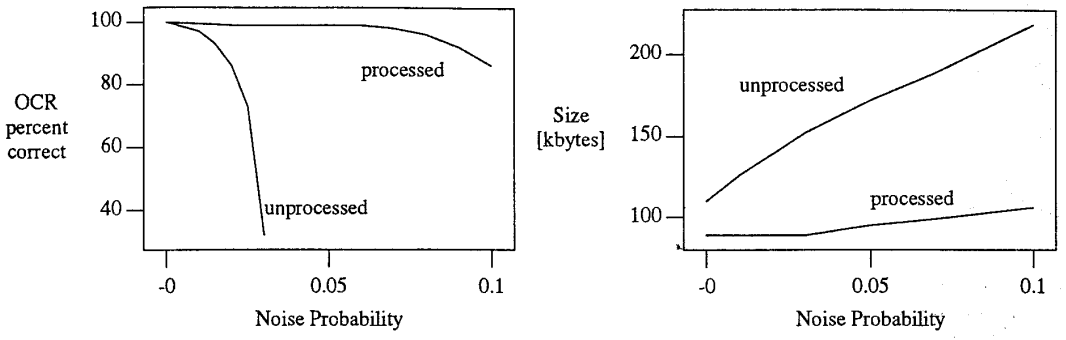

Figure 1: Effects of noise on OCR rate and compression size, with and without *kFill* processing.

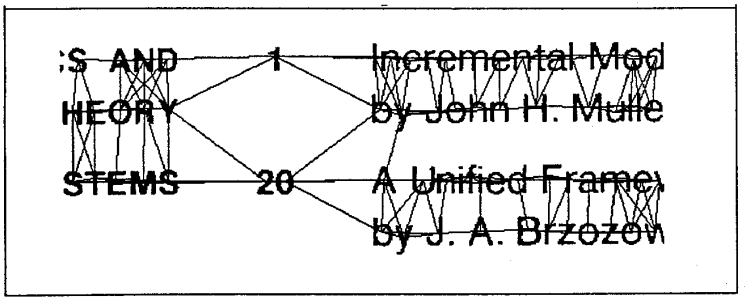

Figure 2: Intermediate results of docstrum analysis on a portion of text from a table of contents page shows nearest-neighbor connections.

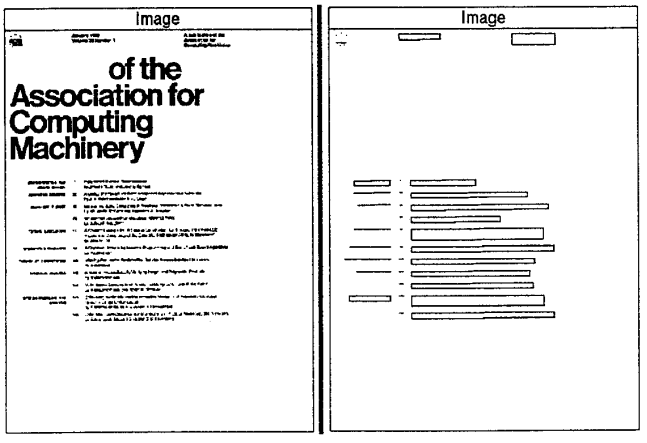

Figure 3: The original of a table of contents page (from the Journal of the ACM) is shown on the left, and its blocks found by docstrum analysis are shown on the right. (Note that the word "Journal" was lost during binarization. It is white on a light-colored background, and the rest of the text is black.)