The Document Spectrum for Page Layout Analysis

Lawrence O'Gorman

Abstract—**Page layout analysis is a document processing technique used to determine the format of a page. This paper describes the document spectrum, or *docstrum*, which is a method for structural page layout analysis based on bottom-up, nearest-neighbor clustering of page components. The method yields an accurate measure of skew, within-line, and between-line spacings and locates text lines and text blocks. It is advantageous over many other methods in three main ways: independence from skew angle, independence from different text spacings, and the ability to process local regions of different text orientations within the same image. Results of the method shown for several different page formats and for randomly oriented subpages on the same image illustrate the versatility of the method. We also discuss the differences, advantages, and disadvantages of the docstrum with respect to other lay-out methods.**

Index Terms— **Document image processing, geometric page layout analysis, page segmentation, skew estimation, structural page layout analysis.**

I. Introduction

PREDICTIONS of the "paperless office," which were made so frequently during the early 1980's, are made far less frequently now. Perhaps it is the sense of futility engendered by the *increase* in volume of paper documents with the prevalence of computers (much of it directly due to computer output) rather than a decrease in volume. Perhaps it is the realization that paper is a legitimate form of media that is well suited for many purposes. Instead of the quest to remove all paper, a more realistic objective is to integrate it into our computer world. The ideal would be that it be as easily machine-readable as magnetic tapes and optical discs. If this were the case, paper documents would have one advantage over these other traditional computer media because they could be read by both man and machine.

There are two analyses necessary for reading documents: one is optical character recognition (OCR), and the other is page lay-out analysis. Advances have been made in OCR technology that enable text to be machine read with rates in the high 90% range. However, although text is an important part of a document, it is also essential to know where the text resides in a page. Page format establishes meaning to regions of text. For instance, when searching for a paper by author name, only the author block of the title page needs to be be examined. Names in the body of the text and in the references have different connotations. Therefore, determining the location of text by page lay-out analysis is an essential complement to OCR.

Manuscript received June 3, 1992; revised November 30, 1992. Recommended for acceptance by Associate Editor R. Kasturi.

The author is with AT&T Bell Laboratories, Murray Hill, NJ, 07974.

IEEE Log Number 9212306.

In this paper, the *docstrum* method for page lay-out analysis is described. This is advantageous over many other methods in three main ways. One is that analysis is independent of page orientation or skew. There is no need to find the skew and correct it prior to docstrum analysis; however, a precise measure of orientation is a byproduct of the analysis. The second is that the method does not require *a priori* input of character size and line spacing. Instead, these parameters are determined in the course of the analysis. The third is that this technique can be applied to an image containing subregions of different document characteristics. For instance, the docstrum can be used to segment independently oriented smaller documents (receipts, checks, index cards, business cards, etc.) in a single image.

II. Background

A. Definitions

Document lay-out analysis is performed to determine **physical structure** of a document, that is, to determine document components. These **document components** can consist of single **connected components**—regions of black (1-valued) pixels that are adjacent to form single regions—such as noise specks, dots, dashes, lines, and the parts of a character that are touching; and groups of connected components, or **blocks**, such as a complete character consisting of one or more connected components (e.g., i, é, ü), a word, text line, or group of text lines. A **text line** is a group of characters, symbols, and words that are adjacent, "relatively close" to each other and through which a straight line can be drawn (usually with horizontal or vertical orientation). The dominant orientation of the text lines determines the **skew angle**. A document originally has zero skew, but when a page is manually scanned or photocopied, nonzero skew may be introduced. Text may also have a chosen orientation that is different from the page skew (for instance, diagonally aligned text in diagrams and charts). We refer to text **orientation** rather than skew when we imply this more general case.

When document layout analysis is performed in a **top-down** manner, a page may be split into one or more column blocks of text, each column is split into paragraph blocks, and each paragraph is split into text lines, etc. Alternatively, analysis may be performed in a **bottom-up** manner, where connected components are merged into characters, then words, then text lines, etc, or both top-down and bottom-up analyses may be combined. **Document lay-out understanding** consists of **functional labeling** of blocks determined by layout analysis. These **functional components** are usually distinguished in some way by their physical features (such as by font size)

and by the "meaning" of a block. Examples of functional components of a technical paper are title block, abstract, section titles, paragraphs, footnotes, etc.

B. Skew Estimation

Skew detection methods are described so that their differences can be contrasted to the docstrum method in Section V. We describe three categories of skew estimation techniques based on their approaches: 1) using the projection profile, 2) fitting baselines by the Hough transform, and 3) by nearest-neighbor clustering.

A commonly used method for skew detection employs the projection profile to iteratively compute skew angle [1]. The projection profile is a histogram of the number of black pixel values accumulated along parallel lines throughout the document. For a document whose text lines span horizontally, the horizontal projection profile will have peaks with widths equal to the character height, and the vertical projection profile will have wide peaks corresponding to each column of text. To determine the skew angle, the projection profile is computed at a number of angles close to the expected orientation, and for each angle, a measure is made of the total variation in the bin heights along the profile. The maximum variation corresponds to the best alignment with the text lines, and this determines the skew angle. Baird [2] proposes modifications to the projection profile method for very fast and accurate iterative convergence on the skew angle. Akiyama and Hagita [3] describe an approach that is very fast but less accurate. A document is divided into equal width vertical strips, horizontal projection profiles are calculated for each vertical strip, and skew is determined by measuring the average shift in zero crossings between strips. Pavlidis and Zhou [4] propose a method where alignment is measured of vertical projection profiles determined for horizontal strips of the page. A restriction of these methods is that they are limited to documents that have fairly small skews that are typically less that $\pm 10°$.

Srihari and Govindaraju [5] and Hinds *et al.* [6] propose similar skew detection methods based on the Hough transform. The Hough transform maps each point in the original (x, y) plane to all points in the (ρ, θ) Hough plane of possible lines through (x, y) with slope θ and distance from origin ρ. The dominant lines are found from peaks in the Hough space and thus the skew. One limitation of this approach is that if text is sparse, as for some forms, it may be difficult to correctly choose a peak in Hough space. The specified range of skew is limited to around $\pm 15°$.

A bottom-up technique for skew estimation based on nearest-neighbor clustering is described by Hashizume *et al.* [7]. In this work, the 1-nearest—neighbors of all connected components are found, the direction vectors for all nearest-neighbor pairs are accumulated in a histogram, and the histogram peak is found to obtain the dominant skew angle. This method has the advantage over the previous methods in that it is not limited to any range of skew. However, since only one nearest-neighbor connection is made for each component, connections with noise, subparts of characters (dot on"i"), and between-line connections can reduce the accuracy of this method. This is the most similar method to the docstrum method.

C. Lay-out Analysis

After skew detection, the image is usually rotated to zero skew, and lay-out analysis is performed. A major difference between the docstrum technique and the top-down techniques described here is that the former is not limited to Manhattan layouts, that is, layouts whose blocks are separable by vertical and horizontal cuts. We describe some top-down and bottom-up layout analysis approaches here so that their advantages and disadvantages can be contrasted with the docstrum in Section V.

Variants of the run-length smoothing algorithm (RLSA), which was first described by Wong *et al.* [8], are often used for lay-out analysis. The method merges characters into words, and words into text lines, and (sometimes) text lines into paragraphs by "smearing" the text to join characters into blobs. This is done by using smoothing filters whose sizes are based on intercomponent spacing. This method requires that the image is skew-corrected and that spacing is known and uniform within the image.

Nagy *et al.* [9], [10] use a top-down approach that combines structural segmentation and functional labeling. Horizontal and vertical projection profiles are used to split the document into successively smaller rectangular blocks. This segmentation is aided by performing functional labeling at the same time based on *a priori* knowledge of features of the expected blocks.

Baird [11] describes a top-down lay-out technique that first analyzes white space to isolate blocks and then uses projection profiles to find lines. As with all the top-down methods described above, the page must have a Manhattan layout.

Akiyama and Hagita [3] perform bottom-up lay-out analysis that employs both global and local text features as well as generic properties of documents in a similar fashion to Nagy's use of expected features, that is, the method yields both structurally and functionally labeled blocks; however, it is dependent on knowing features of the layout.

Fisher *et al.* [12] perform bottom-up segmentation that combines a number of the techniques described above. The skew is first found from the Hough transform, and then, between-line spacing is found as the peak of the 1-D Fourier transform of the horizontal projection profile. Within-line spacing is determined by finding the peak on a histogram of lengths of white spaces between black transitions along the skew angle. Merging of the text components is then done by a sequence of run-length smoothing operations in the directions of the skew and perpendicular to it. Esposito *et al.* [13] use a similar approach, except instead of operating with respect to pixels, they first determine character boxes and then accumulate projection profiles with respect to these.

III. The Docstrum: Formation and Analysis

A. Overview

In this paper, we describe the document spectrum, or *docstrum*. The docstrum is a representation of the document

page that describes global structural features of the page and can be used for page analysis. It is based on bottom-up, k-nearest-neighbor clustering of connected components of the page and has some of the characteristics of the family of bottom-up techniques described above.

The k-nearest-neighbor pairings of connected component elements on the document page are shown in Fig. 1(b). (In addition, see [14] for a description of k-nearest-neighbors.) The k-nearest-neighbors to each component i are the k closest components $0 < j < k$, where closeness is measured by Euclidean distance in the image. Each nearest-neighbor pair $\{i, j\}$ is described by a 2-tuple $D_{ij}(d, \phi)$ of the distance d and the angle ϕ between centroids of the two components. For example, two characters in the same word form a nearest-neighbor pair whose distance is relatively small and whose angle is close to zero if its resident text line has zero skew. For each page element k, nearest neighbors are found. For $k = 5$ (which is the value we usually use), a character might make two or three pairings within a word and across word boundaries within the same line, as well as pairings with characters on upper and lower lines. These between-line pairings have larger distances than within-line pairs and angles that are approximately 90° apart. The docstrum is the plot of $D_{ij}(d, \phi)$ for all nearest-neighbor pairs on a page (see Figs. 2 and 3.) It is a polar plot with origin at the center; radial distance from this is d, and the counterclockwise angle from the horizontal is ϕ. 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. Orientation and text line information can be determined directly from clusters on the docstrum plot. Text lines can be grouped to form blocks by the method to be described.

The resulting text lines and blocks from docstrum analysis are based on structural relationships of these entities. These results are used to perform functional labeling of document parts, perhaps entailing further merging of the structural blocks using application-specific information. We do not describe the functional labeling process here.

B. Preprocessing

Before the docstrum is determined, preprocessing is performed to get rid of noise and isolate document components. (We assume here that any diagrams or pictures have already been separated; therefore, this analysis is being applied to the textual parts of the page image only.) First, salt and pepper noise is reduced by applying the *kFill* filter [15] to the image to remove small regions of 0- or 1-valued pixels within text or background regions, respectively. This filter is designed for noise removal within binary text and removes spurs and smooths ragged edges on text and lines. The filter is designed specifically for binary text to remove noise while retaining text integrity, specifically to maintain corners of characters. Although the scale parameter of *kFill* can be set so that legitimate dots and periods are not removed if they are greater than a pixel in size, we usually do not require this. Instead, more noise is removed, including some dots and periods,

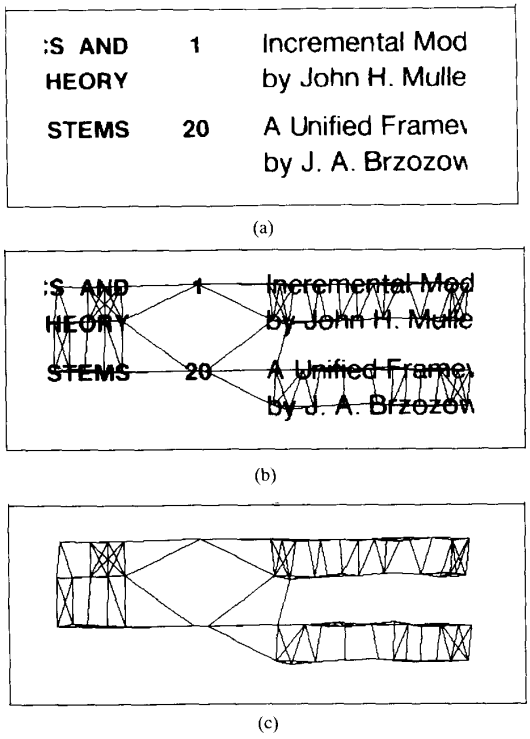


Fig. 1. Nearest-neighbor vectors for a portion of a table of contents image (the full page image is shown in Fig. 10) for k=5. The original image is in (a). In (b), overlays of the nearest-neighbor vectors are shown over the image. For each component, there are five vectors to the five components that are closest in Euclidean distance to the image plane. (Note that where less than five vectors can be seen, this is because two vectors overlay each other.) In (c), only the nearest-neighbor overlays are shown.

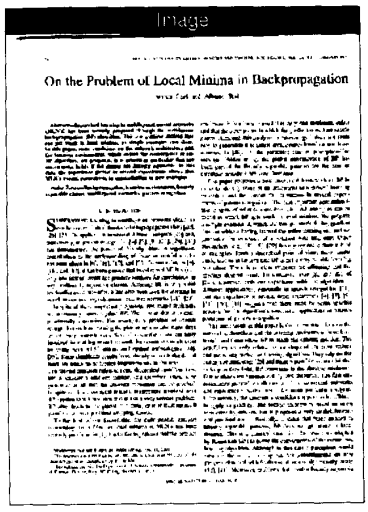


Fig. 2. Image of a title page (from the *IEEE Transactions on Pattern Analysis and Machine Intelligence)* is used for the examples shown in Figs. 3 to 6.

and the loss of these two entities does not adversely affect docstrum analysis.

After *kFill* noise reduction, connected components are found. We use a technique that is different from traditional

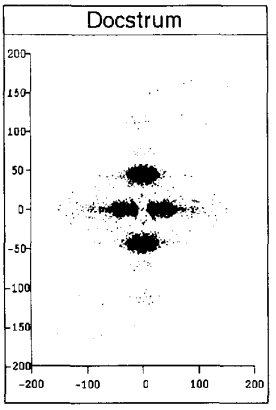


Fig. 3. Docstrum plot $D_{ij}(d, \phi)$ for the image in Fig. 2. Each point represents the distance and angle between a nearest-neighbor pair as the distance and angle from the origin at the center of the plot.

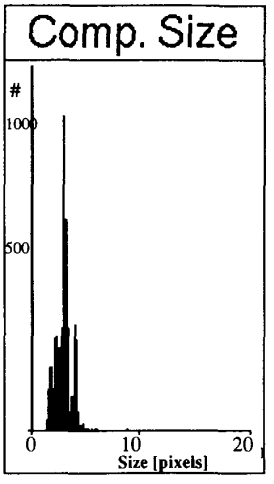


Fig. 4. Component-size histogram for the image in Fig. 2. The size is measured in pixel length as the square root of the bounding box area. The large characters of the title and small dots and noise have already been filtered out.

connected component analysis (which is described in more detail in [16] and [17]). Contours are first found within the image, and these are coded and represented by a line processing and analysis tool called thin line code (TLC). Features are calculated and stored for each TLC component, including bounding box size, contour length, moments, etc. Using these features, further noise reduction and segmentation are performed.

For pages containing text of different sizes, such as larger characters in the titles and smaller characters in the text body, the docstrum may be applied either to a subset of sizes or to each size separately. The reason for separately analyzing different ranges of component sizes before analysis is that the docstrum calculates average feature values, such as average within-line and between-line spacing, and then makes decisions based on these. The presence of a wide range of component sizes will increase the standard deviation of the averages, which may result in erroneous results. A histogram is made of component sizes, where size is that of the bounding box (Fig. 4). For a typical document, there is often a peak at small size for any remaining noise, a wider and higher peak for the distribution of characters of the predominant font size, and, often, a well-separated much smaller peak for title characters. From this data, peak detection can be done and the docstrum analysis confined to any or all of the separate component size groups. Note that this separation of different sizes need only be done for a large size range: the range of sizes between a capital letter, a small letter, and a subscript within an equation falls well within a single size range. (It should be noted that the bounding box measure is not independent of orientation; however, we have found this measure to be sufficient for our purposes because the bounding boxes of similarly shaped characters scale similarly for different orientations.)

Besides performing the separation of features in the preprocessing step, selective docstrum analysis is also performed later during docstrum analysis with respect to nearest-neighbor distances and angles. This is described in Section III-C, F, and I.

C. Nearest-Neighbor Clustering and Docstrum Plot

The first step in docstrum processing is to find the k-nearest-neighbors of each component as shown in Fig. 1. The value of k must first be chosen. Ideally, we would like to find neighbors to the right, left, above, and below each component (when those neighbors are present). This will give information on within-line and between-line spacing. We generally use a value of $k = 5$. For most components, this picks up those neighbors mentioned and an extra one for redundancy. The only disadvantage in choosing a larger value of k than the minimum is the extra computation time required. Other values of k may also be chosen for different purposes. For instance, if only text lines are ultimately desired, then between-line pairs are not needed, and $k = 2$ or 3 will be sufficient. Conversely, if text blocks are desired and between-line spacing is large compared with average within-line spacing, then larger k values are required. For example, if between-line spacing is greater than twice the average within-line spacing but less than three times, then a k value of 6 or 7 would be appropriate.

Because of the large number of components on a typical page (in the order of 1000 to 5000 for the page of a technical journal), the nearest-neighbor clustering step is the most time consuming of the nonpixel processing steps. To find the k-nearest-neighbors of each component in the most straightforward way requires $O(N^2)$ computation. We reduce this substantially by first sorting the components based on x position. Then, we find the closest k neighbors for expanding x distances until the x distance is greater than the kth closest neighbor already found.

Each nearest-neighbor pair i and j is related by distance d and the angle ϕ between components of the pair and is represented on the docstrum by $D_{ij}(d, \phi)$. We consider the vector connecting the pair of elements to be undirected; therefore, ϕ is quantized to $[0, 180°)$. For visual purposes in the docstrum plot, we replicate each point by its mirror point rotated 180°. Thus, the plot is symmetric with respect to the

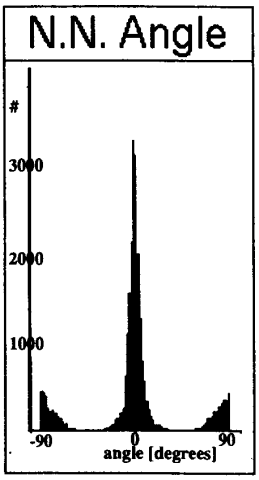


Fig. 5. Nearest-neighbor angle histogram for the image of Fig. 2. The angle range is from -90 to 90°.

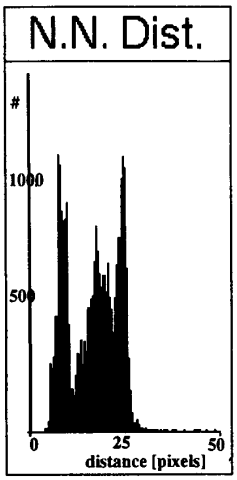


Fig. 6. Nearest-neighbor distance histogram for the image of Fig. 2. The distance is shown in pixels. Large distances are filtered out.

origin.[1] For typeset text, a number of features can be readily seen on the docstrum plot and related to the original page image. Typically, there are four to six clusters (depending on cluster separation) that form a cross pattern. For zero skew, there are two clusters each on the positive and negative x axes and one cluster each on the positive and negative y axes. Depending on the separation, the x axes clusters might appear instead as one cluster each. On the x axis, the clusters of lower distance and larger number indicates between-character spacing. The clusters of further distance and much smaller number are for between-word spacing. (We obtain the peak of one of these two clusters, usually intercharacter spacing, and refer to this distance as within-line spacing.) The y-axis clusters indicate between-line spacing. Although these features can be seen on the docstrum plot, they are not directly measured there. Instead, a particular sequence of analysis steps is performed for more reliable estimation. This is described next.

D. Spacing and Initial Orientation Estimation

Orientation, within-line spacing, and between-line spacing are estimated from the docstrum. Orientation and spacing are most easily seen when summation is carried out on the docstrum over the distance and angle variables, respectively. The histogram in Fig. 5 shows a distribution of nearest-neighbor angles for the docstrum plot of Fig. 3. The histogram in Fig. 6 shows a distribution of nearest-neighbor distances for the same docstrum plot.

Text orientation is first estimated from the nearest-neighbor angle histogram. The histogram is circularly smoothed, that is, the smoothing window is wrapped around at the ends since the angular data is circularly continuous. The peak is found from this smoothed data. To achieve a resolution of r_a°/bin, the histogram has $n_a = 180/r_a$ bins, corresponding to the angular range of $[0, 180)$°, and is smoothed by a rectangular smoothing window that is $W_a\%$ of the total range. For this initial estimate, we use a resolution of $r_a = 0.5$°/bin and a smoothing window of $W_a = 25\%$, giving a histogram length of $n_d = 360$ and a smoothing window length of 90. (This window shape and length are not critical because the histogram usually contains two well-separated, well-defined, and symmetric peaks. The window length should be less than 50% in order not to merge peaks and large enough to perform some smoothing in the sparsest expected histograms.) The location of the peak value is the estimate of average text line orientation or, simply, the orientation. This is an initial estimate; we find a more precise orientation measure later in the analysis.

Two spacing histograms are then compiled: one of nearest-neighbor distances for all angles within a specified angular range of the orientation estimate, which is called the within-line histogram, and the other for angles within the same angular range of the perpendicular to the orientation estimate, which is called the between-line histogram. To achieve a resolution of r_d pixels/bin, the histogram length is $n_d = (\max - \min)/r_d$, where the range of spacing values is [max, min]. The smoothing window is much more critical here than for the orientation estimate because peaks may be closely placed and are not usually symmetric. We define a tolerance that is smaller than the closest expected peak spacings t_s and make the window length a multiple of this $W_s = r_s(2t_s + 1)$. We use $r_s = 2$ pixels and $t_s = 2$ pixels. The peak found of the within-line histogram is an estimate of the character spacing; that from the between-line histogram is an estimate of the text line spacing.

E. Determination of Text Lines and Accurate Orientation Measurement

Although text line determination is synonymous with baseline determination in most of the literature, we find text

[1] It should be noted that angles are not inherently undirected. Just because component A has nearest-neighbor B does not necessarily imply the inverse; therefore, pair order is relevant, and it follows that ϕ quantization of $[0, 360°)$ is also relevant. It has been suggested that for sparse columns of data or perhaps other formats, directedness may be important; however, we do not pursue this further in this paper.

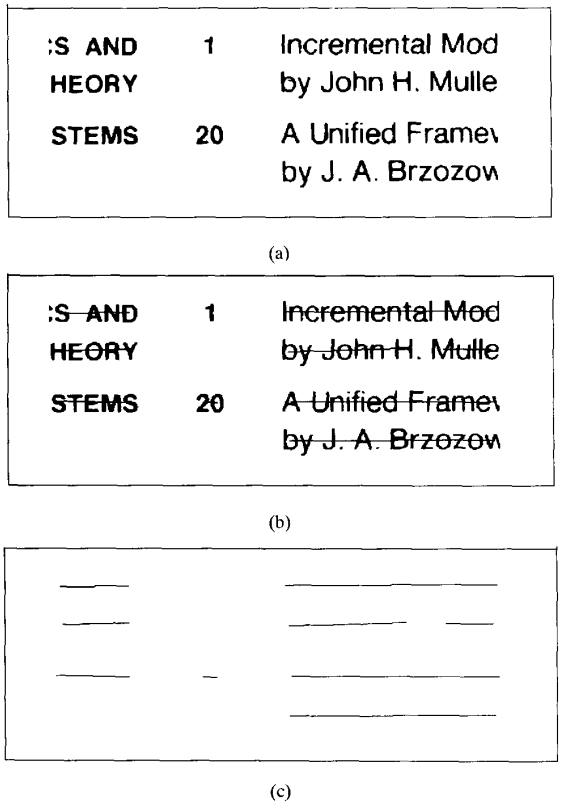


Fig. 7. Text lines for the same of a page shown in Fig. 1: (a) Original image; (b) overlays of the text lines; (c) only the overlays.

lines using connected component centroids instead. This is in keeping with our desired independence from any assumptions of page orientation, and this contrasts with the assumption in *base*line determination that the *bottom* of a character is known. If the baseline of each character could indeed be found at this point in the processing, this would yield better orientation estimates; however, the accuracy of baseline methods is reduced both by descenders and by skew. Of course, the accuracy of the docstrum orientation estimate via the centroids is also affected by descenders, as well as ascenders, but we show in Section IV that the method we describe is accurate and that its independence from orientation is advantageous.

We first perform a transitive closure on within-line nearest-neighbor pairings to obtain nearest-neighbor groups on the same text lines. A regression fit is then made to centroids of each group component to find text lines. This fits a straight line to the centroids in each group by minimizing the sum of squares of errors between the centroids and the line. Fig. 7 shows the text lines found for a portion of text.

From these text lines, we make our final estimation of orientation. This is more accurate than the initial estimate because the longer text lines substantially reduce the effects of descenders and noise.

F. Structural Block Determination

Structural blocks are groupings of one or more text lines grouped on the basis of spatial and geometric characteristics. To determine structural blocks, we extend a method first developed to find regions in contour line patterns [18], [19]. This method used three properties to determine if two lines are in the same pattern group: parallelness, perpendicular proximity, and overlap. For our application, the lines are text lines, and we extend this method by the addition of the property of parallel proximity. Therefore, if two text lines are approximately parallel, close in perpendicular distance, and they either overlap to some specified degree or are separated by only a small distance in parallel distance, then they are said to meet the criteria to belong to the same structural block.

The blocking method consists of examining pairs of text lines to determine if they should be in the same block, based on the criteria. If a pair of text lines meets the criteria, and if neither already belongs to a block, both are assigned to a new block. If one text line already belongs to a block and the other is unassigned, the unassigned text line is assigned to the block of the other. If both belong to different blocks, then these blocks are merged into one block. At the end of this process, each text line has been assigned to a block, and the blocks describe the structural layout of the page.

Although text lines, in general, are parallel to each other, our estimated text lines are only approximately parallel—to the degree of the length of the line and font size with respect to pixel resolution. For instance, a column of one- to three-digit page numbers in a table of contents page will have orientation precision that ranges from no precision for a single digit to better precision for two digits and, better yet, for three digits. Because our measured text lines are only approximately parallel, this leads to less straightforward calculation of structural blocking parameters.

Consider two text lines, which are shown as segments i and j in Fig. 8. These can be expressed as line segments with endpoints $\{(x_{Oi}, y_{Oi}), (x_{Fi}, y_{Fi})\}$ and $\{(x_{Oj}, y_{Oj}), (x_{Fj}, y_{Fj})\}$, respectively. The angular difference between the two segments is

$$\theta_{ij} = \left[\tan^{-1}\frac{\Delta y_j}{\Delta x_j} - \tan^{-1}\frac{\Delta y_i}{\Delta x_i}\right]_{180^\circ}$$

where $\Delta x = x_F - x_O$ and $\Delta y = y_F - y_O$, and $[\cdot]^\circ_{180}$ implies that this value is the smallest positive difference between angles such that $-90 \leq \theta \leq 90^\circ$.

The overlap p_j is an approximation of the length that segment i would overlap onto j if both were parallel and i were translated perpendicularly onto j. The translation of point (x_{Oi}, y_{Oi}) to (x_{Aj}, y_{Aj}) onto the line collinear with segment j is given as

$$x_{Aj} = \frac{x_{Oi}\Delta x_i \Delta x_j + x_{Oj}\Delta y_i \Delta y_j + \Delta x_j \Delta y_i (y_{Oi} - y_{Oj})}{\Delta y_i \Delta y_j + \Delta x_i \Delta x_j}$$

$$y_{Aj} = \frac{\Delta y_j}{\Delta x_j}(x_{Aj} - x_{Oj}) + y_{Oj}. \quad (2)$$

Equation (2) is used when $\Delta x_j \neq 0$. If $\Delta x_j = 0$, then y_{Aj} is calculated first, and x_{Aj} is calculated from that. The translation of (x_{Fi}, y_{Fi}) to (x_{Bj}, y_{Bj}) is calculated similarly. The middle two coordinates then define the overlap segment,

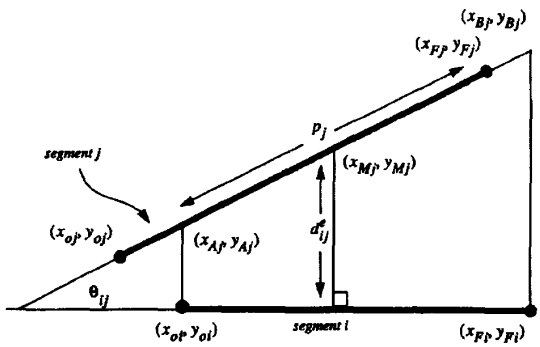


Fig. 8. Variables used in structural block determination. The two text lines, represented by segments i and j, are to be tested here to determine if they should be grouped into the same block. Their angular difference is θ_{ij}. The overlap length of segment i on segment j is p_j (and that is normalized to obtain the overlap parameter). The parallel distance between i and j is $d^a_{ij} = p_j$ in this case. The perpendicular distance betwen i and j is d^e_{ij}.

i.e., middle $\{(x,y)\} = \{(x_{Cj}, y_{Cj}), (x_{Dj}, y_{Dj})\}$. The middle coordinates are found

$$\text{middle}\{(x,y)\} = \{(x_{Cj}, y_{Cj}), (x_{Dj}, y_{Dj})\}$$
$$= \begin{cases} \{(x,y) | x \neq \min\{x\}, x \neq \max\{x\}\}, & \Delta x_j ! = 0 \\ \{(x,y) | y \neq \min\{y\}, y \neq \max\{y\}\}, & \Delta y_j \neq 0 \end{cases}$$

where $\{x\} = \{x_{Oj}, x_{Fj}, x_{Aj}, x_{Bj}\}$ and $\{y\} = \{y_{Oj}, y_{Fj}, y_{Aj}, y_{Bj}\}$. From these middle points, the length of overlap is found:

$$p_j = \left[(y_{Dj} - y_{Cj})^2 + (x_{Dj} - x_{Cj})^2\right] \frac{1}{2}.$$

These middle points are contained within both segments if they are overlapped, or they define a segment between them if they are not overlapped. We test for this and attribute positive overlap to the first case (overlapped) and negative overlap to the second case (nonoverlapped). This length is normalized by the length l_j of segment j between (x_{Oj}, y_{Oj}) and (x_{Fj}, y_{Fj}) to obtain the overlap parameter of segment i onto j

$$p_{ij} = \frac{p_j}{l_j} \text{or} - \frac{p_j}{l_j}.$$

The parallel distance d^a_{ij} between segments i and j describes the distance separating the closer opposite endpoints of each line, if they were collinear. This is the length of the overlap and is positive or negative depending on if there is or is not overlap, i.e.,

$$d_{ij}a = p_j; \text{ or } -p_j.$$

The perpendicular distance d^e_{ij} between segments i and j is the minimum, perpendicular, signed distance from segment i to the midpoint (x_{Mj}, y_{Mj}) of the overlap portion on segment j. The distance is calculated

$$d^e_{ij} = \begin{cases} \frac{(x_{Mj} - x_{Oi}) - (y_{Mj} - y_{Oi})\Delta x_i / \Delta y_i}{(\Delta x_i^2 / \Delta y_i^2 + 1)^{\frac{1}{2}}}, & \Delta x_i, \Delta y_i \neq 0 \\ x_Mj - x_Oi, & \Delta x_i = 0 \\ y_Mj - y_Oi, & \Delta y_i = 0 \end{cases}$$

where the midpoint of the overlap portion is found $x_{Mj} = (x_{Cj} + x_{Dj})/2$, $y_Mj = (y_Cj + y_Dj)\ /\ 2$.

G. Filtering

Filtering may be performed in docstrum analysis to eliminate components not removed during preprocessing or to iteratively apply the docstrum to separate component types of a single image. In practice, filtering is accomplished by flagging selected nearest-neighbor pairs on the basis of component size, nearest-neighbor distance, or nearest-neighbor angle whose value is out of the desired bounds so that docstrum processing is not performed on these. In this way, flags can be turned on and off for different feature values, and docstrum analysis can be performed separately on each range of values. For instance, if larger-size headline text were not eliminated during preprocessing, it could be analyzed separately on the basis that its average within-line spacing is larger than that for the body of the text. In addition, if there were a single angled line of text in a diagram, this could be analyzed separately from the body of text based on its different within-line angle.

H. Global and Local Lay-out Analysis

The analysis described thus far has pertained to global lay-out features, that is, the assumption has been that the analyzed image contains a single formatted page with one orientation. One of the unique features of the docstrum method is the ability to perform multiple local analyses of different layouts on a single image. There are two general cases where this is useful. One is when the image is of multiple, randomly oriented smaller documents, or "scraps," for instance, news clippings, credit card pages, business cards, store receipts, subparts of a larger document, etc. The other is when one or more regions of the same image are formatted at a different orientation. A common example of this is for mileage charts, where city names are written horizontally along the left column but rotated 90° to vertical along the top row. Diagrams also may contain angled or perpendicularly oriented text lines.

In contrast with the layout methods, where skew must be precomputed and lay-out features determined by global processing, the docstrum's bottom-up k-nearest-neighbor approach facilitates local lay-out analysis. The method is straightforward. After k nearest neighbors are found for all components, orientation is determined separately for each group of connected nearest neighbors. Components are said to be in the same group if a path can be found from one component to the other via the nearest-neighbor connections. Block determination is performed on all groups of components with the same approximate orientation. For instance, if an image consists of three scraps (two at the same orientation and one at a different orientation), then one or more groups would be found for each scrap. Blocks would be found for groups within the two scraps that are at the same orientation, and blocks would be found separately for the groups within the other scrap at a different orientation. (Continuing with this same application but following docstrum analysis, data-dependent information can be used (for the business card application, this is the business card dimensions) to group blocks within each scrap). An example of local analysis is shown in Section IV.

This method works well when groups within different scraps are well separated, that is, when the kth-nearest-neighbor does

TABLE I
RESULTS OF DOCSTRUM ANALYSIS FOR IMAGES SHOWN IN FIGS. 9 TO 13. ORIENTATIONS ARE GIVEN IN DEGREES. DISTANCES ARE GIVEN IN PIXELS (WHERE EACH PIXEL SPACING IS 1/300 IN). PROCESSING TIME IS GIVEN IN SECONDS

example	no. of components	no. of pairs	filtered pairs	prelim. orient	final orient.	within-line distance	between-line distance	no. of segments	no. of blocks	process. time
PAMI article	5021	25 105	13 147	0.6	0.1	18.2	49.2	426	18	65
JACM toc	1433	7165	4147	1.2	0.2	16.7	42.2	64	40	8
Spectrum toc	1435	7175	3780	2.4	-0.7	17.5	45.2	248	64	11
PAMI toc	1457	7285	4866	-0.6	1.5	18.1	49.4	93	49	10
cards	1505	6700	2718	-21,9	-22,16	13,25	29,67	80	6	9

not extend outside of a single scrap. When this is not the case, either the value of k should be reduced, or a filter should be applied to nearest-neighbor distance to form nearest neighbors only if the distance is less than some value that relates to the scrap size.

H. Parameters

In this section, we describe the parameter values that can be changed by the user in our implementation and their default values. The component size (area of bounding box in pixels) can have low- and high-pass thresholds. The default lowest size is 3, and the default highest size is 3 times the peak size found during analysis. The procedure can also be operated recursively for all size ranges specified. The default number of nearest neighbors is $k = 5$. If only text lines are important, that value can be reduced to 2 or 3. If between-line distances are large and blocking of text lines is desired, k should be increased such that both within-line and between-line nearest-neighbor connections are made. All processing is done with respect to the orientation angle that is estimated by analysis. There is a tolerance of ±30° around this angle, within which connections are said to be within line. This angular range can also be changed by the user. The nearest-neighbor distance can have low and high bounds as well. In the default mode, the lowest distance is zero, and the highest distance is the lesser of 3 times the within-line distance determined in the analysis or $\sqrt{2}$ times the between-line distance that is also determined at runtime. For most of our applications, the default values of these parameters are appropriate.

For the final stage of block formation—after text line determination—user parameters are more likely to be required due to the wide variety of page formats. The default maximum perpendicular distance between text lines for blocking is 1.3 times the between-line distance found by the analysis. The default maximum parallel distance between ends of text lines for blocking is 1.5 times the within-line distance, which is also found by the analysis. Both of these can be changed by the user either in relative mode (relative to spacings determined by the analysis) or in absolute mode by specifying distances in inches. An option that is available if only text lines are required or if the desired functional blocks cannot be separated structurally is to set the perpendicular distance blocking parameter value to 0 such that blocking does not occur between text lines. The overlap parameter value is set to zero (no overlap is required in the default mode), and the value of the parallelness parameter is set to the same angular range as for the orientation tolerance; the default is 30°. The final choice available to the user is to perform the analysis in global or local mode. Global mode is the default and is for the common case where the entire image has the same skew angle. Local mode is used for subimages of multiple skew angles.

IV. EXAMPLES

The docstrum has been used to segment hundreds of images (mainly of scanned journal pages for the RightPages electronic library system [20] and of business cards for a system built on the local mode docstrum to extract multiple scraps from a single image). In the RightPages system, there are issues from about 40 journals and 15 different publishers, covering over a period of about 2 yr; these represent a wide variety of page formats. The nature of images presented to the local mode docstrum also represent a wide variety of formats. These are the images on which the docstrum has been applied. In this section, we present typical examples of docstrum use and show results. In Section V, we discuss how well the docstrum has performed over these and other images analyzed and describe the conditions for which layout errors result. This information should be used to judge the appropriateness of the docstrum for formats both presented and not presented here.

For the following examples, results are tabulated in Table I. This first shows the number of components found on the page after some of the noise has been removed during preprocessing. The number of nearest-neighbor pairs is given for the chosen number of nearest-neighbors per component of $k = 5$ then given after filtering out those that were at angles outside ±30° of the estimated orientation and that were longer than three times the within-line spacing (all default values). The page orientation is given both for the preliminary estimated analysis and for the final, more accurate analysis. Within-line and between-line spacing values are shown. The number of regression segments and the final number of text blocks are given. Finally, the computation time for docstrum analysis on a Sun Microsystems SparcStation 2 is given. Note that when two numbers are given, this indicates a range of values for multiple local docstrum analyses.

The results are shown in Figs. 9 to 13. In each of these figures, subsampled versions of the original 300 dots/in images are shown at the left, and the blocks found by docstrum analysis are shown at the right. Although the text in these reduced-sized original images is not readable, the objective is to show the structural blocks on the page. For each of these examples, the smaller text of the body of the document comprises the components of interest, and the larger title text is first filtered out. The docstrum analysis could be applied in

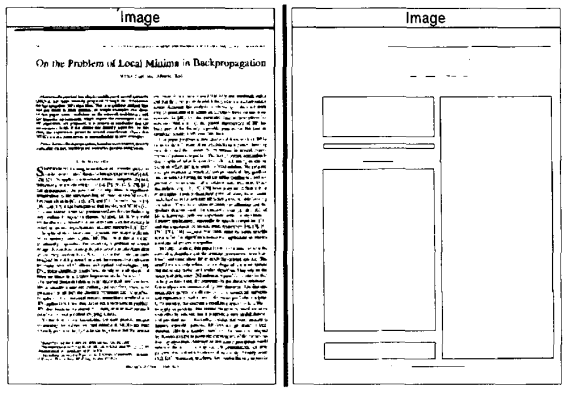


Fig. 9. Original of a journal title page (same as in Fig. 2) is shown on the left, and its blocks found by docstrum analysis are shown on the right.

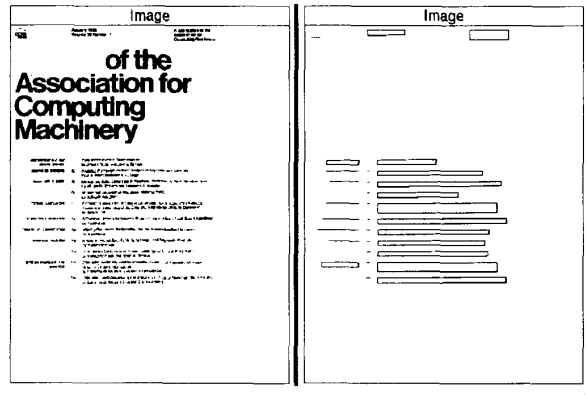


Fig. 10. Original of a table of contents page (from the *Journal of the ACM*) is shown on the left, and its blocks found by docstrum analysis are shown on the right. (Note that the word "Journal" was lost during binarization. It is white on a light-colored background, and the rest of the text is black.)

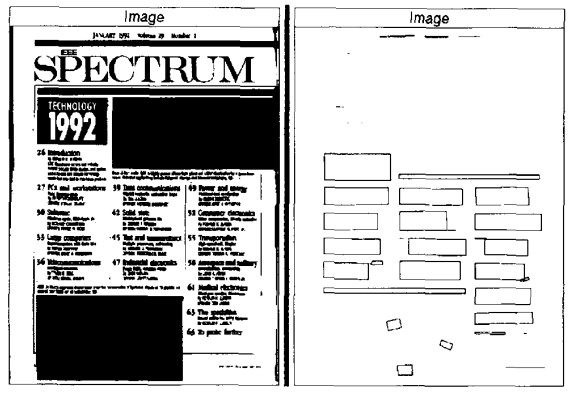


Fig. 11. Original of a three-column table of contents page (from *IEEE Spectrum*) is shown on the left, and its blocks found by docstrum analysis are shown on the right.

the same way to only the title text or to both sizes of text; however, the latter would have to be done in two iterations for the reason given in Section III-B.

The first example, which is shown in Fig. 9, is for the first page of a technical paper from the *IEEE Transactions on Pattern Analysis and Machine Intelligence*. The text of the document body constitutes the larger peak in the size histogram, and it is this size group for which analysis results are shown. In reading order, blocks are found for the following: page number, issue information, author names, abstract, keywords, section title, first column text block, footnote information, second column text block, and order number at the bottom. Although the bulk of the page has been segmented well, the author names have been segmented into four blocks. This is due to the slightly larger font size and word spacing for the authors names than the rest of the text. This would be corrected in the functional labeling stage. Note in Table I that the computation time is significantly larger than for the other examples due to the larger number of components in this article page.

Fig. 10 shows a table of contents image for the *Journal of the ACM*. Characters of larger font size in the journal title are filtered out in the preprocessing step. The three blocks at the top on the right image are for the ACM logo, the date and issue information, and the publisher information, respectively. The table of contents information is arranged with the subject heading (for one or more articles) in the left column, the page number in the middle column, and the title and author information in the blocks of the right column.

Fig. 11 illustrates a three-column table of contents for *IEEE Spectrum*. Each table of contents entry is found as a block that includes the page number, title, subtitle and author names. The two wide, short blocks are for figure captions. Note that this image has not been cleaned of noise, and neither have some of the others, and some extraneous small blocks are found. One reason for this is the presence of noise in the diagram regions, which were not presegmented. Another reason is that many of the characters are actually touching (the issue scanned here is January 1992). For the most part, blocks are correctly found here. For the few extraneous blocks, these can be recognized using journal-specific format information during functional labeling.

The table of contents page in Fig. 12 for the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE contains article entries with equidistant spacing both within and between them. Since structural analysis alone cannot separate entries due to this equidistant spacing, blocking is performed only to join breaks within text lines and not to group between text lines. This is done by setting the perpendicular distance blocking parameter to zero. (This is the only nondefault parameter value used in these examples.) In the figure, each entry has a title, followed by dots, the author names, and page number aligned to the right margin. The separated page number text lines can be used with journal-specific information to segment each entry. In addition, note in this figure that filtering has been performed to get rid of the large font size of the journal title and the small size of the dots within each entry.

The final example in Fig. 13 illustrates local docstrum analysis. Six business cards have been scanned at various orientations on the same image. The original image is shown at the top left. The top right and bottom left images are of the nearest-neighbor connections and text lines, respectively. The

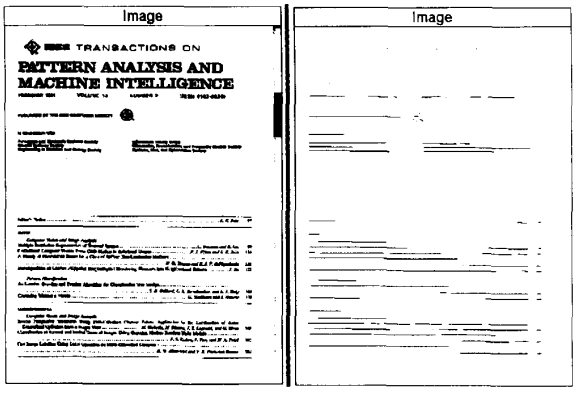


Fig. 12. Original of a table of contents page (from *IEEE Transactions on Pattern Analysis and Machine Intelligence*) is shown on the left, and its text lines found by docstrum analysis are shown on the right.

final image shows each block corresponding to the business cards. In Table I, the range of final orientations is given as -22 to 16° from the horizontal. After docstrum blocking, task-specific information of business card dimensions is used in this example for determining the card boundaries shown.

V. Discussion

In this section, we discuss the advantages and disadvantages of the docstrum method. We compare these with the other methods described in Section II.

One of the most important features of any algorithm is robustness with respect to input parameters. A feature of the docstrum is that spacing parameters are not required from the user. Instead, the docstrum automatically determines dominant spacing from peaks on the histogram for nearest-neighbor distances and then uses multiples of these for text line and block detection. This differs from methods that employ run-length smoothing via a fixed-sized filter window. (Note that the method described in [12] is an exception to this because the window size is a function of the character spacing that is automatically determined.) This also differs from the top-down methods where these global spacings are not automatically calculated during analysis.

Another important feature of the docstrum is independence from page orientation. One ramification of this is that skew correction need not be performed before docstrum analysis, unlike the top-down methods in Section II. Another ramification is that since most skew detection techniques are limited to a small range of skew around 0° and the docstrum is not, the docstrum is more versatile in this respect.

The method is relatively tolerant with respect to randomly distributed noise in the image. When some nearest-neighbor connections are made to noise, this usually does not affect measurement of global parameters, and if the number of nearest-neighbors is chosen large enough for some redundancy, then there is usually not a problem of gaps in text lines. An important case where noise is very destructive to the analysis is when it is not randomly distributed. For instance, we have encountered several examples where salt-and-pepper noise occurs in a line up the gutter of a book or journal being scanned. Since this noise is along a vertical line, the peak orientation can be erroneously found 90° to the correct orientation. Even though most material we scan for the RightPages project is clean, we prefer to err on the side of caution and preprocess all material as described in Section III-B.

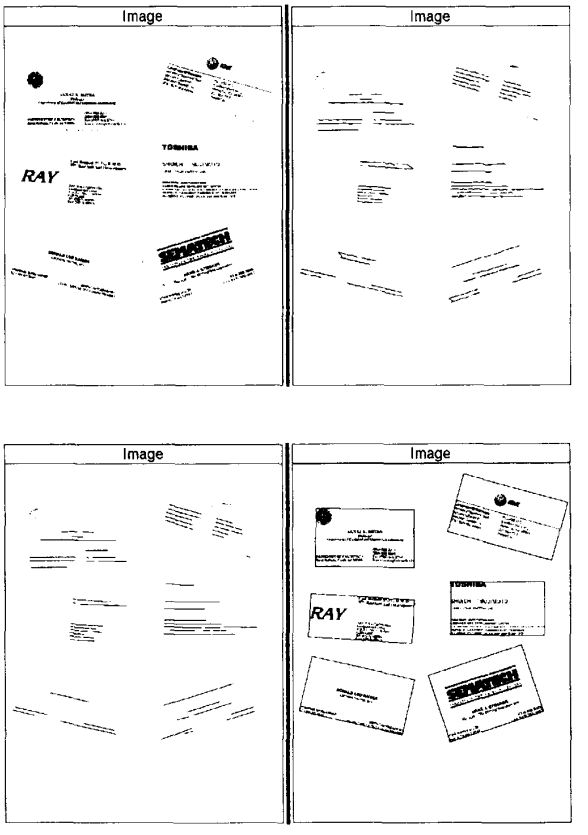


Fig. 13. Image containing six business cards, each with different formats and different orientations, is shown at the top left. The top right image shows the nearest-neighbor vectors. The bottom left image shows the text lines. The image at the lower right shows the blocks.

The docstrum can be applied to text containing tables and equations as well. For tables, the larger separating lines of the table are first filtered out with the size histogram. Then, the same nearest-neighbor analysis is performed. Note that if horizontal spacing is larger than vertical spacing, the text lines will be found to be vertical columns. For equations, the major differences from regular text are the presence of subscripts and superscripts and the differently sized characters. Depending on separating distances, separate short text lines may be found for the subscript and superscript expressions (such as the bounds above and below a summation sign). Since these have the same orientation as the rest of the text, they are merged into the same blocks in the structural block determination step. As far as the differently sized characters in an equation, they are not of great enough deviation from the rest of the text size to affect the results.

One disadvantage of bottom-up versus top-down methods is that they are generally more computationally expensive. It has been shown in Table I that the method is slower with more image components. Although this is true, comparison against other methods must take into account the following. One is that the docstrum requires no previous skew correction because orientation is found as part of the analysis, unlike top-down methods. Another is that many of the skew estimation methods are iterative, requiring more iterations with larger skew or greater precision. The docstrum does not iterate to determine skew. Because the docstrum analysis measures spacing as part of the analysis, this precludes the need for some other automatic process to do so. Furthermore, all docstrum processing is done on connected components rather than pixels. Many of the other methods do the same, but for the ones that do not, pixel access is more time consuming because the ratio of pixels to components is on the order of 10^3.

The docstrum operates on ranges of global spacing values to obtain the nearest-neighbor pairs. Because larger size title text also has longer inter-character distances than smaller text on the same page, the title text and body text must be analyzed separately. This is unlike some of the top-down methods based on projection profiles where these are handled on the same step. However, performing multiple analyses of the docstrum does not require much additional time since one of the steps (for the title text) usually deals with much sparser data.

Since the docstrum measures document features via nearest-neighbor pairs, it is essential that characters be separated. For the same reason, this method is also inappropriate for joined characters of handwriting. This is true of the other bottom-up methods but different from methods using run-length smoothing.

The use of the centroid of components for estimating orientation and spacing is different from the other methods use of the "bottom" point of the bounding box. The use of the centroid enables the docstrum to be independent of orientation. Because both ascenders and descenders affect the centroid location versus just descenders for the baseline methods, initial skew estimation via docstrum nearest-neighbor connections is slightly less precise than for baseline methods for pages with small skew. However, the regression fits to text lines improve precision. Furthermore, with larger skew, the baseline methods lose precision, whereas the docstrum centroids are independent of skew. We have not undertaken a comparison of skew precision for different methods in this paper, but that would be useful future work.

Although we have found docstrum analysis up to the stage of text line detection to be very robust, block segmentation is less so. For instance, some pages are formatted with spacing between functional blocks that shrinks and expands with the amount of text on the page. When the spacing shrinks to the same size as the between-line spacing within a block, structural layout analysis alone is insufficient to perform segmentation. For these cases, we use the docstrum to the text line stage and then employ page-specific format information to perform functional segmentation. Of course, there is also a problem when an expected format changes rules. This is handled not at this stage but in the following functional labeling stage, where results inconsistent to the expected format are flagged.

As far as the feature of the docstrum of being able to perform multiple analyses on blocks of different orientations, it may be argued that most pages contain rectangular regions of singular skew. This point is quite valid; however, we have mentioned some applications where this multiorientation capability is useful. A drawback of the local versus global mode is that measurement of skews and spacings are less precise than when information from the entire page is used. This is true, but we have found that precision is adequate for the cases tested. Although this mode of analysis is not the most common, it speaks well of the versatility of the algorithm that it can handle such cases. None of the methods described in Section II have been shown to handle these. One disadvantage of this local docstrum mode is that the user must be careful that components of different orientation groups are not "too close" such that nearest neighbors are formed across these groups. This means it should be assured that the maximum distance of the kth nearest neighbor should be within the same orientation group. As mentioned in Section III-H, k may be reduced, or a filter may be put on the maximum length of the nearest-neighbors found such that connections outside of an orientation group are not made. In practice, such as for the multiple business cards application of Fig. 13, the user is told to space the cards at least a half card distance away from adjacent cards.

VI. Summary

The docstrum has been presented as a method for page layout analysis. It proceeds in a bottom-up manner by forming k nearest-neighbor pairs between image components, automatically estimating text orientation and spacing parameters, and forming text lines and blocks. Three characteristics of the method have been described: independence from skew range, independence from different text spacings, and ability to process local regions of different text orientations within the same image. It has been shown by example how the docstrum is used on different page formats. Finally, there is a discussion of the differences, advantages, and disadvantages among several families of skew detection and layout analysis methods, pointing out how the docstrum compares with each.

References

[1] W. Postl, "Detection of linear oblique structures and skew scan in digitized documents," in *Proc. 8th Int. Conf. Patt. Recogn. (ICPR)* (Paris), Oct. 1986, pp. 687–689.

[2] H. S. Baird, "The skew angle of printed documents," in *Proc. Conf. Soc. Photog. Scien. Eng.* (Rochester, NY), May 1987, pp. 14–21.

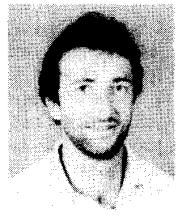
[3] T. Akiyama and N. Hagita, "Automated entry system for printed documents," *Patt. Recogn.*, vol. 23, no. 11, pp. 1141–1154, 1990.

[4] T. Pavlidis and J. Zhou, "Page segmentation by white streams," in *Proc. First Int. Conf. Document Anal. Recogn. (ICDAR)* (St. Malo, France), Sept. 1991, pp. 945–953.

[5] S. N. Srihari and V. Govindaraju "Analysis of textual images using the Hough transform," *Machine Vision Applications*, vol. 2, pp. 141–153, 1989.

[6] S. C. Hinds, J. L. Fisher, and D.P. D'Amato, "A document skew detection method using run-length encoding and the Hough transform," in *Proc. 10th Int. Conf. Patt. Recogn. (ICPR)* (Atlantic City, NJ), June 1990, pp. 464–468.

[7] A. Hashizume, P. -S. Yeh, and A. Rosenfeld, "A method of detecting the orientation of aligned components," *Patt. Recogn. Lett.*, vol. 4, pp. 125–132, 1986.
[8] K. Y. Wong, R. G. Casey, and F. M. Wahl, "Document analysis system," *IBM J. Res. Development*, vol. 6, pp. 642–656, Nov. 1982.
[9] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," in *Proc. 7th Int. Conf. Patt. Recogn. (ICPR)* (Montreal, Canada), 1984, pp. 347–349.
[10] G. Nagy, S. Seth, and M. Viswanathan, "A prototype document image analysis system for technical journals," *IEEE Comput.*, Special issue on Document Image Analysis Systems, pp. 10–22, July 1992.
[11] H. S. Baird, S. E. Jones, and S. J. Fortune, "Image segmentation using shape-directed covers," in *Proc. 10th Int. Conf. Patt. Recogn. (ICPR)* (Atlantic City, NJ), June 1990, pp. 820–825.
[12] J. L. Fisher, S. C. Hinds, and D. P. D'Amato, "A rule-based system for document image segmentation," in *Proc. 10th Int. Conf. Patt. Recogn. (ICPR)* (Atlantic City, NJ), June 1990, pp. 567–572.
[13] R. Esposito, D. Malerba, and G. Semeraro, "An experimental page layout recognition system for office document automatic classification: An integrated approach for inductive generalization," *Proc. 10th Int. Conf. Patt. Recogn. (ICPR)* (Atlantic City, NJ), June 1990, pp. 557–562.
[14] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973, pp. 103–105.
[15] L. O'Gorman, "Image and document processing techniques for the RightPages electronic library system," in *Proc. 11th Int. Conf. Patt. Recogn. (ICPR)* (The Hague, The Netherlands), Aug. 1992, pp. 260–263.
[16] L. O'Gorman, "Primitives chain code", in *Progress in Computer Vision and Image Processing* (A. Rosenfeld and L. G. Shapiro, Eds.). San Diego: Academic, 1992, pp. 167–183.
[17] H. V. Jagadish and L. O'Gorman, "An object model for image recognition," *IEEE Comput.*, vol. 22, no. 12, pp. 33–41, Dec 1989.
[18] L. O'Gorman and G. I. Weil, "An approach toward segmenting contour line regions," in *Proc. 8th Int. Conf. Patt. Recogn.* (Paris), Oct. 1986, pp. 254–258.
[19] M. Seul, L. R. Monar, L. O'Gorman, and R. Wolfe, "Morphology and local structure in labyrinthine stripe domain phases," *Sci.*, vol. 254, Dec. 13, 1991, pp. 1616–1618.
[20] G. A. Story, L. O'Gorman, D. Fox, L. Schaper, and H. V. Jagadish, "The RightPages: An electronic library for alerting and browsing," *IEEE Comput.*, pp. 17–26, 1992.



Lawrence O'Gorman received the B.A.Sc. degree form the University of Ottawa, Ontario, in 1978, the M.S. degree from the University of Washington, Seattle, in 1980, and the Ph.D. degree from Carnegie-Mellon University, Pittsburgh, PA, in 1983, all in electrical engineering.

From 1980 to 1981, he was with Computing Devices Company, Ottawa, Canada, where he worked on digital signal processing and filter design. He has been at AT&T Bell Laboratories, Murray Hill, NJ, in the Computing Systems Research Laboratory since 1984. His research interests include image processing, pattern recognition, document image analysis, and machine vision precision. Lately, he has been involved in the design of the RightPages electronic library project.