IEEE Computer Society Executive Briefings

Document Image Analysis

Lawrence O’Gorman

Rangachar Kasturi

ISBN 0-8186-7802-X

Library of Congress Number 97-17283

1997

Table of Contents

Preface

In the late 1980's, the prevalence of fast computers, large computer memory, and inexpensive scanners fostered an increasing interest in document image analysis. With many paper documents being sent and received via fax machines and being stored digitally in large document databases, the interest grew to do more with these images than simply view and print them. Just as humans extract information from these images, research was performed and commercial systems built to read text on a page, to find fields on a form, and to locate lines and symbols on a diagram. Today, the results of research work in document processing and optical character recognition (OCR) can be seen and felt every day. OCR is used by the post offices to automatically route mail. Engineering diagrams are extracted from paper for computer storage and modification. Handheld computers recognize symbols and handwriting for use in niche markets such as inventory control. In the future, applications such as these will be improved, and other document applications will be added. For instance, the millions of old paper volumes now in libraries will be replaced by computer files of page images that can be searched for content and accessed by many people at the same time — and will never be mis-shelved. Business people will carry their file cabinets in their portable computers, and paper copies of new product literature, receipts, or other random notes will be instantly filed and accessed in the computer. Signatures will be analyzed by the computer for verification and security access.

This book describes some of the technical methods and systems used for document processing of text and graphics images. The methods have grown out of the fields of digital signal processing, digital image processing, and pattern recognition. The objective is to give the reader an understanding of what approaches are used for application to documents and how these methods apply to different situations. Since the field of document processing is relatively new, it is also dynamic, so current methods have room for improvement, and innovations are still being made. In addition, there are rarely definitive techniques for all cases of a certain problem.

The intended audience is executives, managers, and other decision makers whose business requires some acquaintance or understanding of document processing. (We call this group "executives" in accordance with the *Executive Briefing* series.) Some rudimentary knowledge of computers and computer images will be helpful background for these readers. We begin at basic principles (such as, what is a pixel?), but do not belabor them. The reader is expected not so much as to have knowledge of picture processing as to have a level of comfort with the tasks that can be accomplished on a computer and the digital nature by which any computer technique operates. A grasp of the terminology goes a long way toward aiding the executive in discussing the problem. For this reason, each section begins with a list of keywords that also appears in the index. With knowledge of the terminology and whatever depth of method or system understanding that he or she decides to take from the text, the executive should be well-equipped to deal with document processing issues.

In each chapter, we attempt to identify major problem areas and to describe more than one method applied to each problem, along with advantages and disadvantages of each method. This gives an understanding of the problems and also the nature of trade-offs that so often must be made in choosing a method. With this understanding of the problem and a knowledge of the methodology options, an executive will have the technical background and context with which to ask questions, judge recommendations, weigh options, and make decisions.

We include both technology description as well as references to the technical papers that best give details on the techniques. The technology descriptions in the book are enough to understand the methods; if implementation is desired, the references will facilitate this. Popular and accepted methods are emphasized so that the executive can compare against the options offered against the accepted options. In many cases, these options are advanced methods not currently used in commercial products. But, depending on the level of need, advanced methods can be implemented by the programming staff to yield better results. We also describe many full systems entailing document processing. These are described from a high enough level as to be generally understandable and, in addition, to motivate understanding of some of the techniques.

The book is organized in the sequence that document images are usually processed. After document input by digital scanning, pixel processing is first performed. This level of processing includes operations that are applied to all image pixels. These include noise removal, image enhancement, and segmentation of image components into text and graphics (lines and symbols). Feature-level analysis treats groups of pixels as entities, and includes line and curve detection, and shape description. The last two chapters separate text and graphics analysis. Text analysis includes optical character recognition (OCR) and page format recognition. Graphics analysis includes recognition of components of engineering drawings, maps, and other diagrams.

Chapter 1

What is a Document Image and What Do We Do With It?

Traditionally, transmission and storage of information have been by paper documents. In the past few decades, documents increasingly originate on the computer, however, in spite of this, it is unclear whether the computer has decreased or increased the amount of paper. Documents are still printed out for reading, dissemination, and markup. The oft-repeated cry of the early 1980's for the “paper-less office” has now given way to a different objective, dealing with the flow of electronic and paper documents in an efficient and integrated way. The ultimate solution would be for computers to deal with paper documents as they deal with other forms of computer media. That is, paper would be as readable by the computer as magnetic and optical disks are now. If this were the case, then the major difference — and the major advantage — would be that, unlike current computer media, paper documents could be read by both the computer and people.

The objective of document image analysis is to recognize the text and graphics components in images, and to extract the intended information as a human would. Two categories of document image analysis can be defined (see Figure 1). Textual processing deals with the text components of a document image. Some tasks here are: recognizing the text by optical character recognition (OCR), determining the skew (any tilt at which the document may have been scanned into the computer), finding columns, paragraphs, text lines, and words. Graphics processing deals with the non-textual line and symbol components that make up line diagrams, delimiting straight lines between text sections, company logos, etc. Because many of these graphics components consist of lines, such processing includes line thinning, line fitting, and corner and curve detection. (It should be noted that the use of “line” in this book can mean straight, curved, or piecewise straight and/or curved lines. When straightness or curvedness is important, it will be specified.) Pictures are a third major component of documents, but except for recognizing their location on a page, further analysis of these is usually the task of other image processing and machine vision techniques, so we do not deal with picture processing in this book. After application of these text and graphics analysis techniques, the megabytes of initial data are culled to yield a much more concise semantic description of the document.

It is not difficult to find examples of the need for document analysis. Look around the workplace

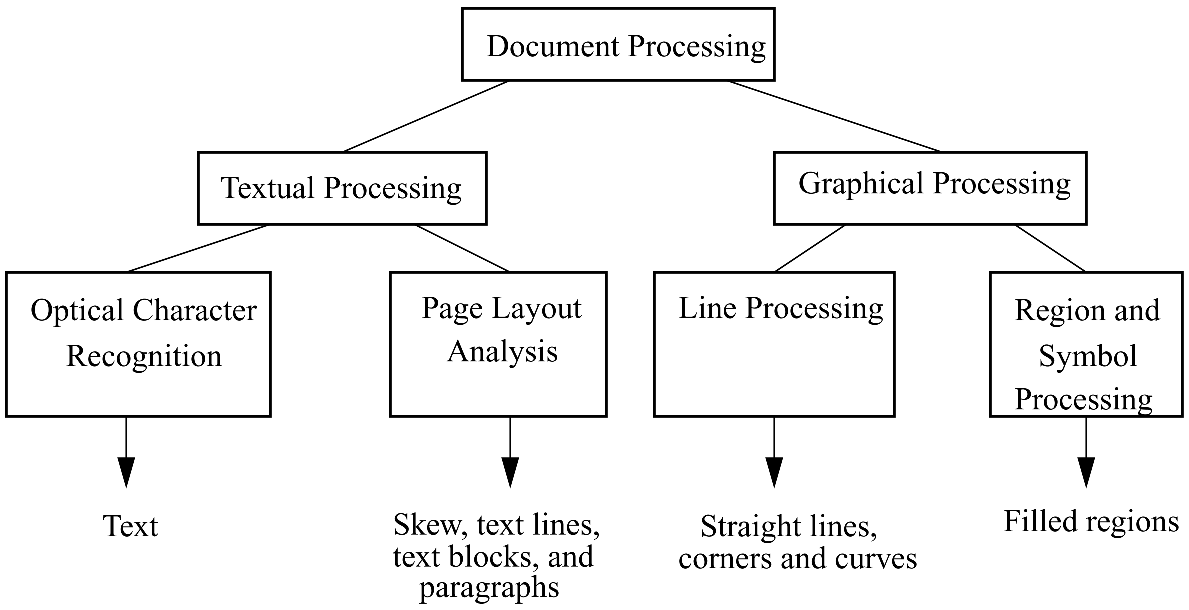

Document Processing
Textual Processing
Graphical Processing
Optical Character Recognition
Page Layout Analysis
Line Processing
Region and Symbol Processing
Text
Skew, text lines, text blocks, and paragraphs
Straight lines, corners and curves
Filled regions

Figure 1. A hierarchy of document processing subareas listing the types of document components dealt with in each subarea.

to see the stacks of paper. Some may be computer generated, but if so, inevitably by different computers and software such that even their electronic formats are incompatible. Some will include both formatted text and tables as well as handwritten entries. There are different sizes, from a 3.5x2" (8.89x5.08cm) business card to a 34x44" (86x111cm) engineering drawing. In many businesses today, imaging systems are being used to store images of pages to make storage and retrieval more efficient. Future document analysis systems will recognize types of documents, enable the extraction of their functional parts, and be able to translate from one computer generated format to another. There are many other examples of the use of and need for document systems. Glance behind the counter in a post office at the mounds of letters and packages. In some U.S. post offices, over a million pieces of mail must be handled each day. Machines to perform sorting and address recognition have been used for several decades, but there is the need to process more mail, more quickly, and more accurately. As a final example, examine the stacks of a library, where row after row of paper documents are stored. Loss of material, misfiling, limited numbers of each copy, and even degradation of materials are common problems, and may be improved by document analysis techniques. All of these examples serve as applications ripe for the potential solutions of document image analysis.

Though document image analysis has been in use for a couple of decades (especially in the banking business for computer reading of numeric check codes), it is just in the late 1980's and early 1990's that the area has grown much more rapidly. The predominant reason for this is greater speed and lower cost of hardware now available. Since fax machines have become ubiquitous, the cost of optical scanners for document input have dropped to the level that these are affordable to even small businesses and individuals. Though document images contain a relatively large amount of data, even personal computers now have adequate speed to process them. Computer main memory also is now adequate for large document images, but more importantly, optical memory is now available for mass storage of large amounts of data. This improvement in hardware, and the increasing use of computers for storing paper documents, has led to increasing interest in improving the technology of document processing and recognition. An essential complement to these hardware improvements are the advancements being made in document analysis software and algorithms. With OCR recognition rates now in the mid to high 90% range, and other document processing methods achieving similar improvements, these advances in research have also driven document image analysis forward.

As these improvements continue, document systems will become increasingly more evident in the form of every-day document systems. For instance, OCR systems will be more widely used to store, search, and excerpt from paper-based documents. Page layout analysis techniques will recognize a particular form, or page format and allow its duplication. Diagrams will be entered from pictures or by hand, and logically edited. Pen-based computers will translate handwritten entries into electronic documents. Archives of paper documents in libraries and engineering companies will be electronically converted for more efficient storage and instant delivery to a home or office computer. Though it will be increasingly the case that documents are produced and reside on a computer, the fact that there are very many different systems and protocols, and also the fact that paper is a very comfortable medium for us to deal with, ensures that paper documents will be with us to some degree for many decades to come. The difference will be that they will finally be integrated into our computerized world.

Hardware Advancements and the Evolution of Document Image Analysis

The field of document image analysis can be traced back through a computer lineage that includes digital signal processing and digital image processing. Digital signal processing, whose study and use was initially fostered by the introduction of fast computers and algorithms such as the fast Fourier transform in the mid 1960's, has as its objective the interpretation of one-dimensional signals such as speech and other audio. In the early 1970's, with larger computer memories and still faster processors, image processing methods and systems were developed for analysis of two-dimensional signals including digitized pictures. Special fields of image processing are associated with particular application — for example, biomedical image processing for medical images, machine vision for processing of pictures in industrial manufacturing, and computer vision for processing images of three-dimensional scenes used in robotics, for example.

In the mid- to late-1980's, document image analysis began to grow rapidly. Again, this was predominantly due to hardware advancements enabling processing to be performed at a reasonable cost and time. Whereas a speech signal is typically processed in frames of 256 samples long and a machine vision image size is 512x512 pixels, a document image is from 2550x3300 pixels for a business letter digitized at 300 dots per inch (dpi) (12 dots per millimeter) to 34000x44000 pixels for a 34x44" E-sized engineering diagram digitized at 1000 dpi.

Commercial document analysis systems are now available for storing business forms, performing OCR on typewritten text, and compressing engineering drawings. Document analysis research continues to pursue more intelligent handling of documents, better compression — especially through component recognition — and faster processing.

From Pixels to Paragraphs and Drawings

Figure 2 illustrates a common sequence of steps in document image analysis. This is also the organization of chapters in this book. After data capture, the image undergoes pixel level processing and feature analysis, then text and graphics are treated separately for recognition of each.

Data capture is performed on a paper document usually by optical scanning. The resulting data is stored in a file of picture elements, called pixels, that are sampled in a grid pattern throughout the document. These pixels may have values: OFF (0) or ON (1) for binary images, 0-255 for gray-scale images, and 3 channels of 0-255 color values for color images. At a typical sampling resolu-

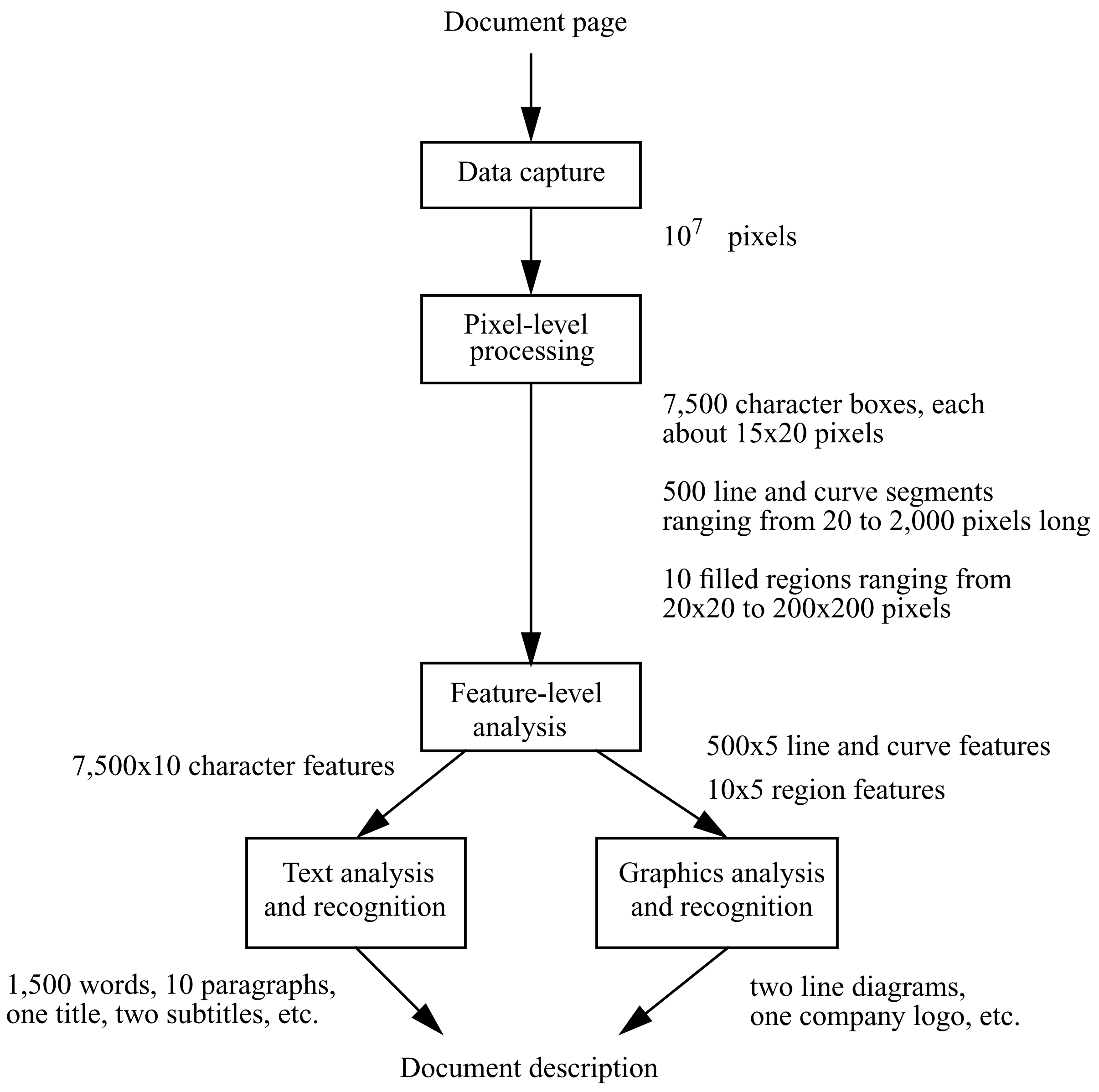

Document page
Data capture
10^7 pixels
Pixel-level processing
7,500 character boxes, each about 15x20 pixels
500 line and curve segments ranging from 20 to 2,000 pixels long
10 filled regions ranging from 20x20 to 200x200 pixels
Feature-level analysis
7,500x10 character features
500x5 line and curve features
10x5 region features
Text analysis and recognition
Graphics analysis and recognition
1,500 words, 10 paragraphs, one title, two subtitles, etc.
two line diagrams, one company logo, etc.
Document description

Figure 2. A typical sequence of steps for document analysis, along with examples of intermediate and final results and the data size.

tion of 300 dpi, a 8.5x11" page would yield an image of 2550x3300 pixels. It is important to understand that the image of the document contains only raw data that must be further analyzed to glean the information. For instance, Figure 3 shows the image of a letter “e”. This is a pixel array of ON or OFF values whose shape is known to humans as the letter “e” — however to a computer it is just a string of bits in computer memory.

Pixel-level processing (Chapter 2). This stage includes binarization, noise reduction, signal

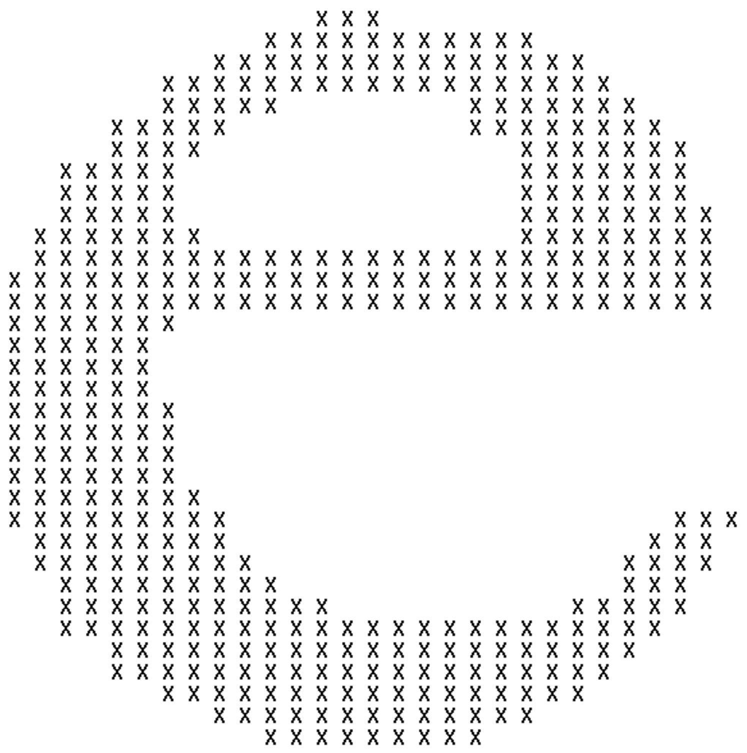


Figure 3. A binary image of the letter “e” is made up of ON and OFF pixels, where the ON pixels are shown here as “X”s.

enhancement, and segmentation. For gray-scale images with information that is inherently binary such as text or graphics, binarization is usually performed first. The objective in methods for binarization is to automatically choose a threshold that separates the foreground and background information. An example that has seen much research is the extraction of handwriting from a bank check — especially a bank check containing a scenic image as a background.

Document image noise occurs from image transmission, photocopying, or degradation due to aging. Salt-and-pepper noise (also called impulse noise, speckle noise, or just dirt) is a common form of noise on a binary image. It consists of randomly distributed black specks on white background and white specks on black background. It is reduced by performing filtering on the image where the background is “grown" into the noise specks, thus filling these holes. Signal enhancement is similar to noise reduction, but uses domain knowledge to reconstitute expected parts of the signal that have been lost. Signal enhancement often applied to graphics components of document images to fill gaps in lines that are known to be continuous otherwise.

Segmentation occurs on two levels. On the first level, if the document contains both text and graphics, these are separated for subsequent processing by different methods. On the second level, segmentation is performed on text by locating columns, paragraphs, words, titles, and captions; and on graphics, segmentation usually includes separating symbol and line components. For instance, in a page containing a flow chart with an accompanying caption, text and graphics are first separated. Then the text is separated into that of the caption and that of the chart. The graphics is separated into rectangles, circles, connecting lines, etc.

Feature-level analysis (Chapter 3). In a text image, the global features describe each page, and consist of skew (the tilt at which the page has been scanned), line lengths, line spacing, etc. There are also local features of individual characters, such as font size, number of loops in a character, number of crossings, accompanying dots, etc., which are used for OCR.

In a graphical image, global features describe the skew of the page, the line widths, range of curvature, minimum line lengths, etc. Local features describe each corner, curve, and straight line, as well as the rectangles, circles, and other geometric shapes.

Recognition of text and graphics (Chapters 4 and 5). The final step is the recognition and description step, where components are assigned a semantic label and the entire document is described as a whole. It is at this stage that domain knowledge is used most extensively. The result is a description of a document as a human would give it. For a text image, we refer not to pixel groups, or blobs of black on white, but to titles, subtitles, bodies of text, footnotes, etc. Depending on the arrangement of these text blocks, a page of text may be a title page of a paper, a table of contents of a journal, a business form, or the face of a mail piece. For a graphical image, an electrical circuit diagram for instance, we refer not to lines joining circles and triangles and other shapes, but to connections between AND gates, transistors and other electronic components. The components and their connections describe a particular circuit that has a purpose in the known domain. It is this semantic description that is most efficiently stored and most effectively used for common tasks such as indexing and modifying particular document components .

Chapter 2

Preparing the Document Image

2.1: Introduction

Data capture of documents by optical scanning or by digital video yields a file of picture elements, or pixels, that is the raw input to document analysis. These pixels are samples of intensity values taken in a grid pattern over the document page, where the intensity values may be: OFF (0) or ON (1) for binary images, 0-255 for gray-scale images, and 3 channels of 0-255 color values for color images. The first step in document analysis is to perform processing on this image to prepare it for further analysis. Such processing includes: thresholding to reduce a gray-scale or color image to a binary image, reduction of noise to reduce extraneous data, and thinning and region detection to enable easier subsequent detection of pertinent features and objects of interest. This pixel-level processing (also called preprocessing and low-level processing in other literature) is the subject of this chapter.

2.2: Thresholding

Keywords: thresholding, binarization, global thresholding, adaptive thresholding, intensity histogram

In this treatment of document processing, we deal with images containing text and graphics of binary information. That is, these images contain a single foreground level that is the text and graphics of interest, and a single background level upon which the foreground contrasts. We will also call the foreground: objects, regions of interest, or components. (Of course, documents may also contain true gray-scale (or color) information, such as in photographic figures; however, besides recognizing the presence of a gray-scale picture in a document, we leave the analysis of pictures to the more general fields of image analysis and machine vision.) Though the information is binary, the data — in the form of pixels with intensity values — are not likely to have only two levels, but instead a range of intensities. This may be due to nonuniform printing or nonuniform reflectance from the page, or a result of intensity transitions at the region edges that are located between foreground and background regions. The objective in binarization is to mark pixels that belong to true foreground regions with a single intensity (ON) and background regions with a different intensity (OFF). Figure 1 illustrates the results of binarizing a document image at different threshold values. The ON-values are shown in black in our figures, and the OFF-values are white.

For documents with a good contrast of components against a uniform background, binary scanners are available that combine digitization with thresholding to yield binary data. However, for the many documents that have a wide range of background and object intensities, this fixed threshold level often does not yield images with clear separation between the foreground components and background. For instance, when a document is printed on differently colored paper or when the foreground components are faded due to photocopying, or when different scanners have different light levels, the best threshold value will also be different. For these cases, there are two alternatives. One is to empirically determine the best binarization setting on the scanner (most binary scanners provide this adjustment), and to do this each time an image is poorly binarized. The other alternative is to start with gray-scale images (having a range of intensities, usually from 0 to 255) from the digitization stage, then use methods for automatic threshold determination to better perform binarization. While the latter alternative requires more input data and processing, the advantage is that a good threshold level can be found automatically, ensuring consistently

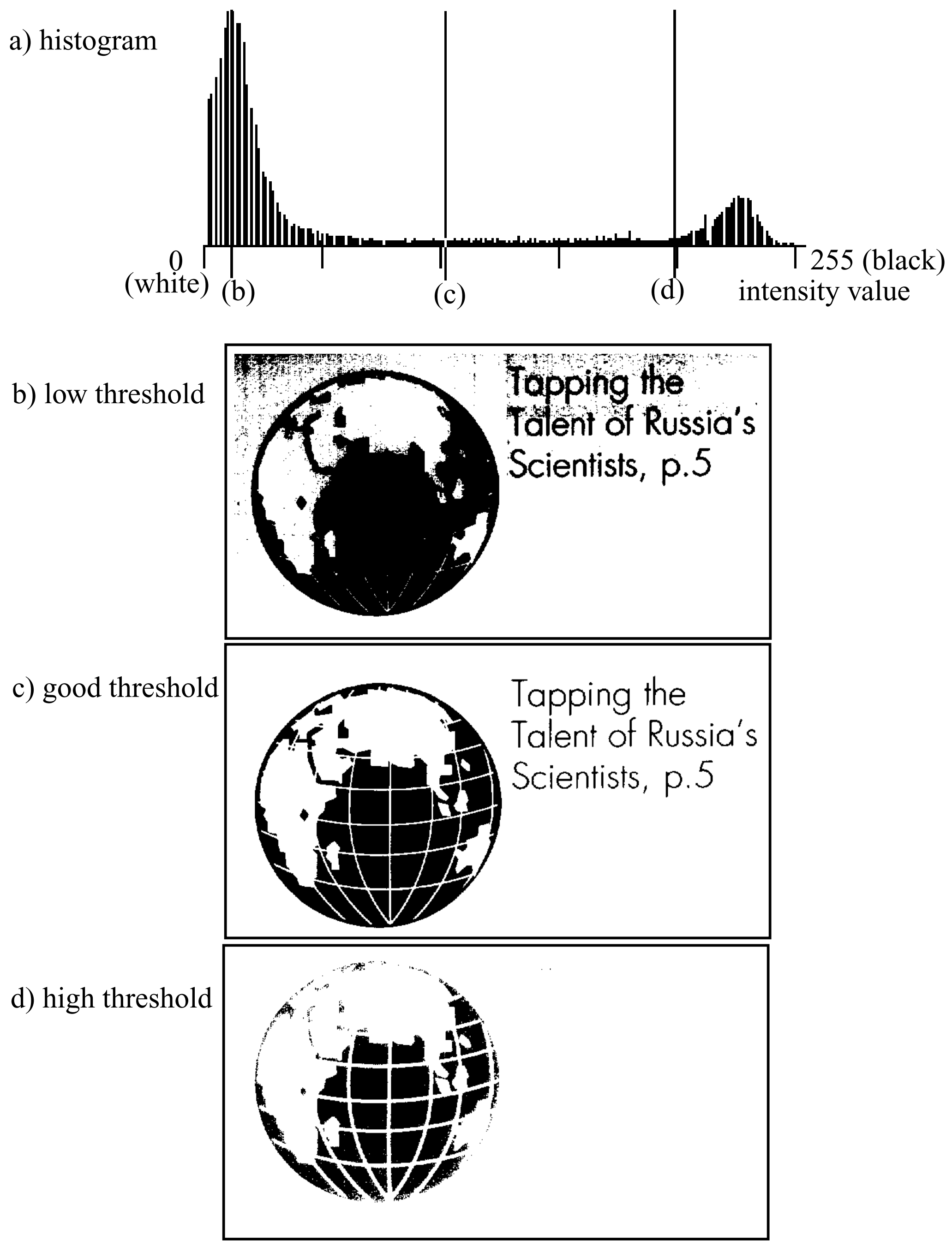


Figure 1. Image binarization. a) Histogram of original gray-scale image. Horizontal axis shows markings for threshold values of images below. The lower peak is for the white background pixels, and the upper peak is for the black foreground pixels. Image binarized with: (b) too low a threshold value, c) a good threshold value, and d) too high a threshold value.

good images, and precluding the need for time-consuming manual adjustment and repeated digitization. The following discussion presumes initial digitization to gray-scale images.

If the pixel values of the components and those of the background are fairly consistent in their respective values over the entire image, then a single threshold value can be found for the image. This use of a single threshold for all image pixels is called global thresholding. Processing methods are described below that automatically determine the best global threshold value for different images. For many documents, however, a single global threshold value cannot be used even for a single image due to nonuniformities within foreground and background regions. For example, for a document containing white background areas as well as highlighted areas of a different background color, the best thresholds will change by area. For this type of image, different threshold values are required for different local areas; this is adaptive thresholding, and is also described below.

2.2.1: Global Thresholding

The most straightforward way to automatically select a global threshold is by use of a histogram of the pixel intensities in the image. The intensity histogram plots the number of pixels with values at each intensity level. See Figure1 for a histogram of a document image. For an image with well-differentiated foreground and background intensities, the histogram will have two distinct peaks. The valley between these peaks can be found as the minimum between two maxima and the intensity value there is chosen as the threshold that best separates the two peaks.

There are a number of drawbacks to global threshold selection based on the shape of the intensity distribution. The first is that images do not always contain well-differentiated foreground and background intensities due to poor contrast and noise. A second is that, especially for an image of sparse foreground components, such as for most graphics images, the peak representing the foreground will be much smaller than the peak of the background intensities. This often makes it difficult to find the valley between the two peaks. In addition, reliable peak and valley detection are separate problems unto themselves. One way to improve this approach is to compile a histogram of pixel intensities that are weighted by the inverse of their edge strength values [Mason 1975]. Region pixels with low edge values will be weighted more highly than boundary and noise pixels with higher edge values, thus sharpening the histogram peaks due to these regions and facilitating threshold detection between them. An analogous technique is to highly weight intensities of pixels with high edge values, then choose the threshold at the peak of this histogram, corresponding to the transition between regions [Weszka 1979]. This requires peak detection of a single maximum, and this is often easier than valley detection between two peaks. This approach also reduces the problem of large size discrepancy between foreground and background region peaks because edge pixels are accumulated on the histogram instead of region pixels; the difference between a small and large size area is a linear quantity for edges versus a much larger squared quantity for regions. A third method uses a Laplacian weighting. The Laplacian is the second derivative operator, which highly weights transitions from regions into edges (the first derivative highly weights edges). This will highly weight the border pixels of both foreground regions and their surrounding backgrounds, and because of this the histogram will have two peaks of similar area. Though these histogram shape techniques offer the advantage that peak and valley detection are intuitive, still peak detection is susceptible to error due to noise and poorly separated regions. Furthermore, when the foreground or background region consists of many narrow regions, such as for text, edge and Laplacian measurement may be poor due to very abrupt transitions (narrow edges) between foreground and background.

A number of techniques determine classes by formal pattern recognition techniques that optimize some measure of separation. One approach is minimum error thresholding [Kittler 1986, Ye 1988]. (See Figure 2.) Here, the foreground and background intensity distributions are modeled as normal (Gaussian or bell-shaped) probability density functions. For each intensity value (from 0 to 255, or a smaller range if the threshold is known to be limited to it), the means and variances are calculated for the foreground and background classes, and the threshold is chosen such that the misclassification error between the two classes is minimized. This latter method is classified as a parametric technique because of the assumption that the gray-scale distribution can be modeled as a probability density function. This is a popular method for many computer vision applications, but some experiments indicate that documents do not adhere well to this model, and thus results with this method are poorer than non-parametric approaches [Abutaleb 1989]. One non-parametric approach is Otsu's method [Otsu 1979, Reddi 1984]. Calculations are first made of the ratio of between-class variance to within-class variance for each potential threshold value. The classes here are the foreground and background pixels and the purpose is to find the threshold that maximizes the variance of intensities between the two classes, and minimizes them within each class. This ratio is calculated for all potential threshold levels and the level at which the ratio is maximum is the chosen threshold. A similar approach to Otsu's employs an information theory measure, entropy, which is a measure of the information in the image expressed as the average number of bits required to represent the information [Kapur 1985, Abutaleb 1989]. Here, the entropy for the two classes is calculated for each potential threshold, and the threshold where the sum of the two entropies is largest is chosen as the best threshold. Another thresholding approach is by moment preservation [Tsai 1986]. This is less popular than the methods above, however, we have found it to be more effective in binarizing document images containing text. For this method, a threshold is chosen that best preserves moment statistics in the resulting binary image as compared with the initial gray-scale image. These moments are calculated from the intensity histogram — the first four moments are required for binarization.

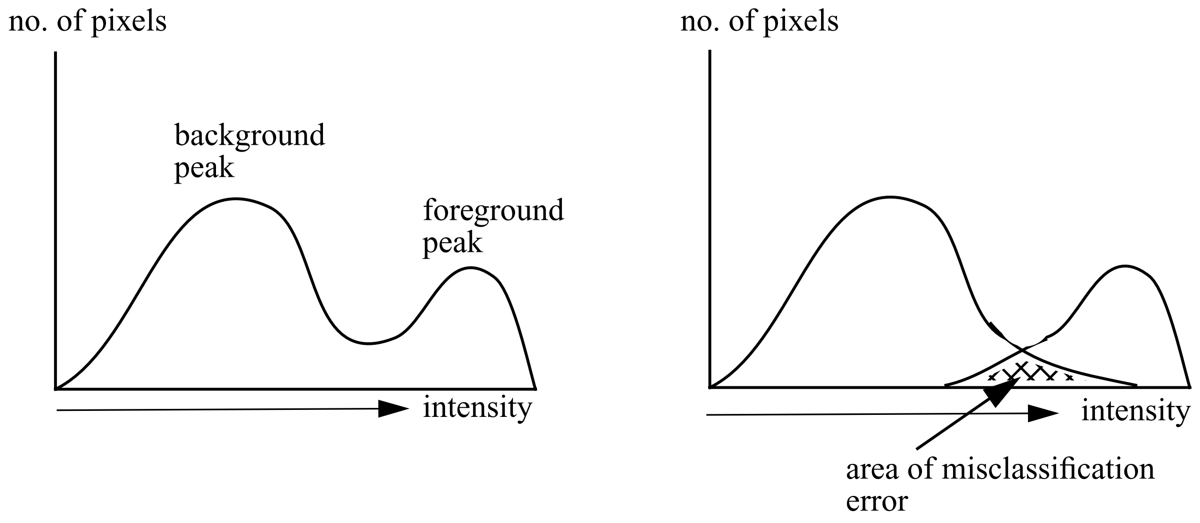


Figure 2. Illustration of misclassification error in thresholding. On left is intensity histogram showing foreground and background peaks. On right, the tails of the foreground and background populations have been extended to show the intensity overlap of the two populations. It is evident that this overlap makes it impossible to correctly classify all pixels by means of a single threshold. The minimum-error method of threshold selection minimizes the total misclassification error.

Many images have more than just two levels. For instance, magazines often employ boxes to highlight text where the background of the box has a different color than the white background of the page. In this case, the image has three levels: background, foreground text, and background of highlight box. To properly threshold an image of this type, multi-thresholding must be performed. There are many fewer multi-thresholding methods than binarization methods. Most (e.g. [Otsu 79]) require that the number of levels is known. For the cases where the number of levels is not known beforehand, one method [O'Gorman 94] will determine the number of levels automatically and perform appropriate thresholding. This added level of flexibility may sometimes lead to unexpected results. For instance, a magazine cover with three intensity levels may be thresholded to four levels instead due to the presence of an address label that is thresholded at a separate level.

2.2.2: Adaptive Thresholding

A common way to perform adaptive thresholding is by analyzing gray-level intensities within local windows across the image to determine local thresholds [Casey and Wong 90, Kamel 93]. White and Rohrer [White and Rohrer 83] describe an adaptive thresholding algorithm for separating characters from background. The threshold is continuously changed through the image by estimating the background level as a two-dimensional running-average of local pixel values taken for all pixels in the image. (See Figure 3.) Mitchell and Gillies [Mitchell and Gillies 89] describe a similar thresholding method where background white level normalization is first done by estimating the white level and subtracting this level from the raw image. Then, segmentation of characters is accomplished by applying a range of thresholds and selecting the resulting image with the least noise content. Noise content is measured as the sum of areas occupied by components that are smaller and thinner than empirically determined parameters. Looking back at the results of binarization for different thresholds in Figure 1, it can be seen that the best threshold selection yields the least visible noise. The main problem with any adaptive binarization technique is the choice of window size. The chosen window size should be large enough to guarantee that a large enough number of background pixels are included to obtain a good estimate of average value, but not so large as to average over nonuniform background intensities. However, often the features in the image vary in size such that there are problems with fixed window size. To remedy this, domain dependent information can be used to check that the results of binarization give the expected features (a large blob of an ON-valued region is not expected in a page of smaller symbols, for instance). If the result is unexpected, then the window size can be modified and binarization applied again.

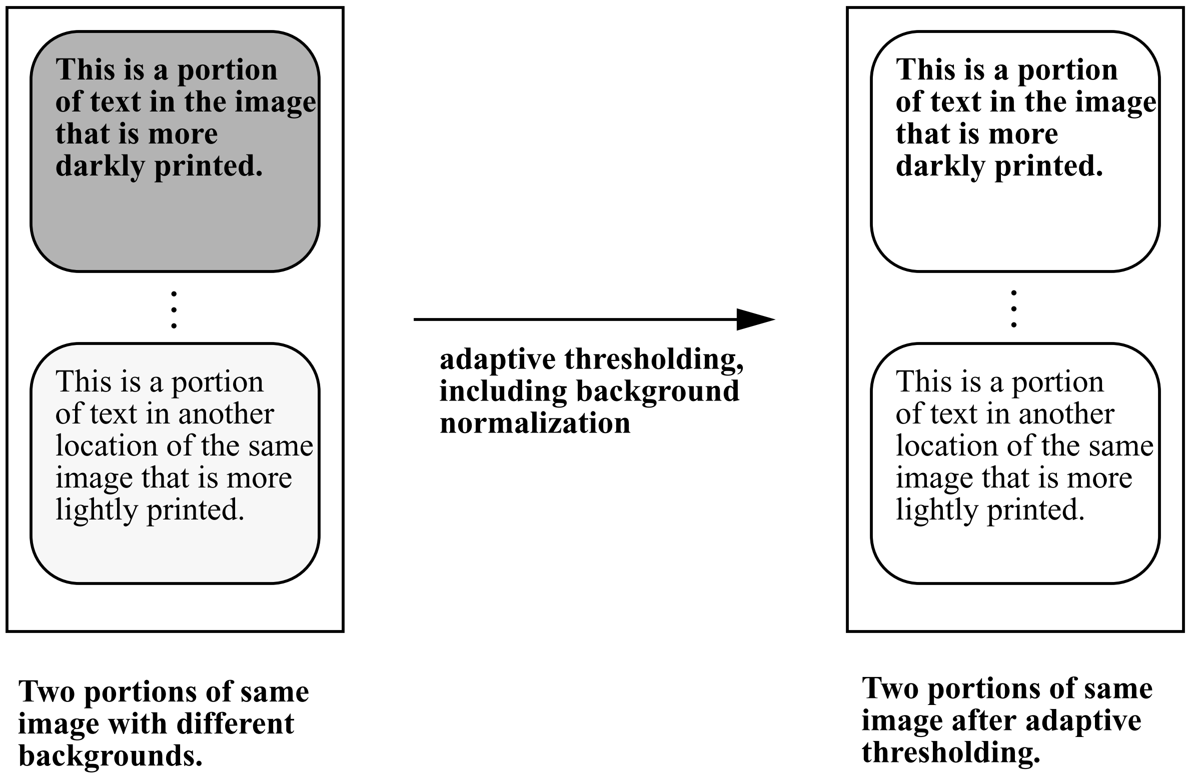

This is a portion of text in the image that is more darkly printed.
This is a portion of text in another location of the same image that is more lightly printed.
adaptive thresholding, including background normalization
This is a portion of text in the image that is more darkly printed.
This is a portion of text in another location of the same image that is more lightly printed.
Two portions of same image with different backgrounds.
Two portions of same image after adaptive thresholding.

Figure 3. Diagram illustrates adaptive thresholding by background normalization. Original image on left has portions of text with different average background values. The image on the right shows that the backgrounds have been eliminated leaving only ON on OFF.

2.2.3: Choosing a Thresholding Method

Whether global or adaptive thresholding methods are used for binarization, one can never expect perfect results. Depending on the quality of the original, there may be gaps in lines, ragged edges on region boundaries, and extraneous pixel regions of ON and OFF values. This fact that processing results will not be perfect is generally true with other document processing methods, and indeed image processing in general. The recommended procedure is to process as well as possible at each step of processing, but to defer decisions that don't have to be made until later steps to avoid making irreparable errors. In later steps there is more information as a result of processing to that point, and this provides greater context and higher level descriptions to aid in making correct decisions, and ultimately recognition. Deferment, when possible, is a principle appropriate for all stages of document analysis (except, of course, the last).

A number of different thresholding methods have been presented in this section. It is the case that no single method is best for all image types and applications. For simpler problems where the image characteristics do not vary much within the image or across different images, then the simpler methods will suffice. For more difficult problems of noise or varying image characteristics, more complex (and time-consuming) methods will usually be required. Commercial products vary in their thresholding capabilities. Today's scanners usually perform binarization with respect to a fixed threshold. More sophisticated document systems provide manual or automatic h9istogram-based techniques for global thresholding. The most common use of adaptive thresholding is in special purpose systems used by banks to image checks. The best way to choose a method at this time is first by narrowing the choices by the method descriptions, then just experimenting with the different methods and examining their results.

Because there is no "best" thresholding method, there is still room for future research here. One problem that requires more work is to identify thresholding methods or approaches that best work on documents with particular characteristics. Many of the methods described above were not formulated in particular for documents, and their performance on them is not well known. Documents have characteristics, such as very thin lines that will favor one method above another. Related to this is how best to quantify the results of thresholding. For text, one way is to perform optical character recognition on the binarized results and measure the recognition rate for different thresholds. Another problem that requires further work is that of multi-thresholding. Sometimes documents have not two, but three or more levels of intensities. For instance, many journals contain highlighting boxes within the text, where the text is against a background of a different gray level or color. While multi-thresholding capabilities have been claimed for some of the methods discussed above, not much dedicated work has been focused on this problem.

For other reviews and more complete comparisons of thresholding methods, see [Sahoo 1988, O'Gorman 1994] on global and multi-thresholding techniques, and [Trier and Taxt 1995] on adaptive techniques.

We suggest just manually setting a threshold when the documents are similar and testing is performed beforehand. For automatic, global threshold determination, we have found (in [O'Gorman 1994]) that the moment-preserving method [Tsai 1986] works well on documents. For adaptive thresholding, the method of [Kamel 1993] is a good choice. This paper also gives background and comparison on these adaptive methods. For multi-thresholding, the method [Reddi 1984] is appropriate if the number of thresholds is known, and the method [O'Gorman 1994] if not.

References

1. D. Mason, I.J. Lauder, D. Rutoritz, G. Spowart, "Measurement of C-bands in human chromosomes", Computers in Biology and Medicine, Vol. 5, 1975, 179-201.
2. J.S. Weszka, A. Rosenfeld, "Histogram modification for threshold selection", IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-9, No. 1, Jan. 1979, p. 38-52.
3. J. Kittler, J. Illingworth, "Minimum error thresholding", Pattern Recognition, Vol. 19, No. 1, 1986, pp. 41-47.
4. Q-Z. Ye, P-E Danielson, "On minimum error thresholding and its implementations", Pattern Recognition Letters, Vol. 7, 1988, pp. 201-206.
5. A.S. Abutaleb, "Automatic thresholding of gray-level pictures using two-dimensional entropy", Computer Vision, Graphics, and Image Processing, Vol. 47, 1989, pp. 22-32.
6. N. Otsu, "A threshold selection method from gray-level histograms", IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-9, No. 1, Jan. 1979, pp. 62- 66.
7. S.S. Reddi, S.F. Rudin, H.R. Keshavan, "An optimal multiple threshold scheme for image segmentation", IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-14, No. 4, July/Aug, 1984, pp. 661-665.
8. J.N. Kapur, P.K. Sahoo, A.K.C. Wong, "A new method for gray-level picture thresholding using the entropy of the histogram", Computer Vision, Graphics, and Image Processing, Vol. 29, 1985, pp. 273-285.
9. W-H. Tsai, "Moment-preserving thresholding: A new approach," Computer Vision, Grapics, and Image Processing, Vol. 29, 1985, pp. 377-393.
10. R.G. Casey, K.Y. Wong, "Document analysis systems and techniques", in *Image Analysis Applications*, R. Kasturi and M.M. Trivedi (eds), Marcel Dekker, 1990, pp. 1-36.
11. M. Kamel, A. Zhao, "Extraction of binary character/graphics images from grayscale docu-

ment images", CVGIP: Graphical Models and Image Processing, Vol. 55, No. 3, 1993, pp. 203-217.

12. J.M. White, G.D. Rohrer, "Image thresholding for optical character recognition and other applications requiring character image extraction", IBM J. Res. Development, Vol. 27, no. 4, July 1983, pp. 400- 411.
13. B.T. Mitchell, A.M. Gillies, "A model-based computer vision system for recognizing handwritten ZIP codes", Machine Vision and Applications, Vol. 2, 1989, pp. 231-243.
14. P.K. Sahoo, S. Soltani, A.K.C. Wong, Y.C. Chen, "A survey of thresholding techniques", Computer Vision, Graphics, and Image Processing, Vol. 41, 1988, pp. 233-260.
15. O.D. Trier and T. Taxt, "Evaluation of binarization methods for document images," IEEE Trans. PAMI, Vol. 17, No. 3, Mar. 95, pp. 312-315.
16. L. O'Gorman, "Binarization and multi-thresholding of document images using connectivity", CVGIP: Graphical Models and Image Processing, Vol. 56, No. 6, Nov. pp. 494-506, 1994.

2.3: Noise Reduction

Keywords: filtering, noise reduction, salt-and-pepper noise, filling, morphological processing, cellular processing

After binarization, document images are usually filtered to reduce noise. Salt-and-pepper noise (also called impulse and speckle noise, or just dirt) is a prevalent artifact in poorer quality document images (such as poorly thresholded faxes or poorly photocopied pages). This appears as isolated pixels or pixel regions of ON noise in OFF backgrounds or OFF noise (holes) within ON regions, and as rough edges on characters and graphics components. (See Figure 1.) The process of reducing this is called "filling". The most important reason to reduce noise is that extraneous features will otherwise cause subsequent errors in recognition. Another reason is that noise reduction reduces the size of the image file, and this in turn reduces the time required for subsequent processing and storage. The objective in the design of a filter to reduce noise is that it remove as much of the noise as possible while retaining all of the signal.

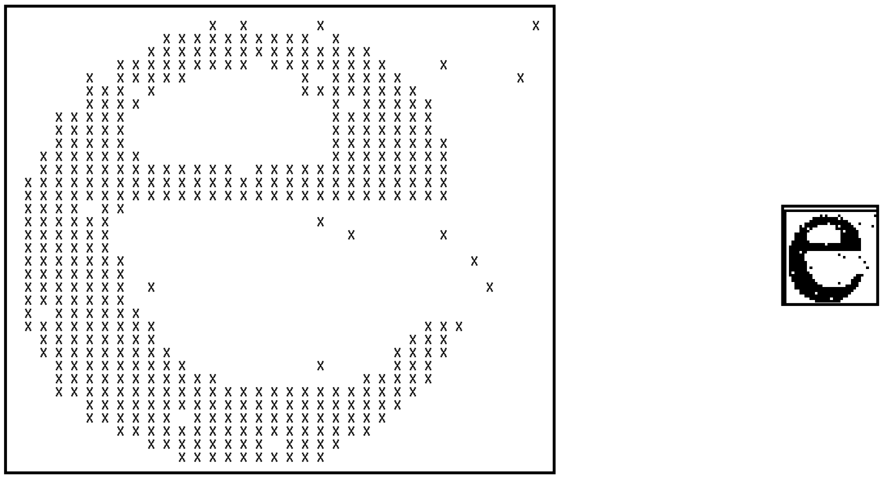


Figure 1. Illustration of letter "e" with salt-and-pepper noise. On the left, the letter is shown with its ON and OFF pixels as "X"s and blanks, and on the right the noisy character is shown.

2.3.1: Morphological and Cellular Processing

Morphological [Serra 1982, Haralick 1987, Haralick 1992] and cellular processing [Preston 1979] are two families of processing methods by which noise reduction can be performed. (These methods are much more general than for just the noise reduction application mentioned here, but we leave further description of the methods to the references.) The basic morphological or cellular operations are erosion and dilation. Erosion is the reduction in size of ON-regions. This is most simply accomplished by peeling off a single-pixel layer from the outer boundary of all ON regions on each erosion step. Dilation is the opposite process, where single-pixel, ON-valued layers are added to boundaries to increase their size. These operations are usually combined and applied iteratively to erode and dilate many layers. One of these combined operations is called opening, where one or more iterations of erosion are followed by the same number of iterations of dilation. The result of opening is that boundaries can be smoothed, narrow isthmuses broken, and small noise regions eliminated. The morphological dual of opening is closing. This combines one or more iterations of dilation followed by the same number of iterations of erosion. The result of closing is that boundaries can be smoothed, narrow gaps joined, and small noise holes filled. See Figure 2 for an illustration of morphological operations.

2.3.2: Text and Graphics Noise Filters

For documents, more specific filters can be designed to take advantage of the known characteristics of the text and graphics components. In particular, we desire to maintain sharpness in these document components —not to round corners and shorten lengths, as some noise reduction filters will do. For the simplest case of single-pixel islands, holes, and protrusions, these can be found by passing a 3×3 sized window over the image that matches these patterns [Shih and Kasturi 1988], then filled. For noise larger than one-pixel, the kFill filter can be used [O'Gorman 1992].

In lieu of including a paper on noise reduction, we describe the kFill noise reduction method in more detail. Filling operations are performed within a kxk 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 groups of ON-pixels in the neighborhood, and the r variable is the number of corner pixels that are ON (OFF). Filling occurs when the following

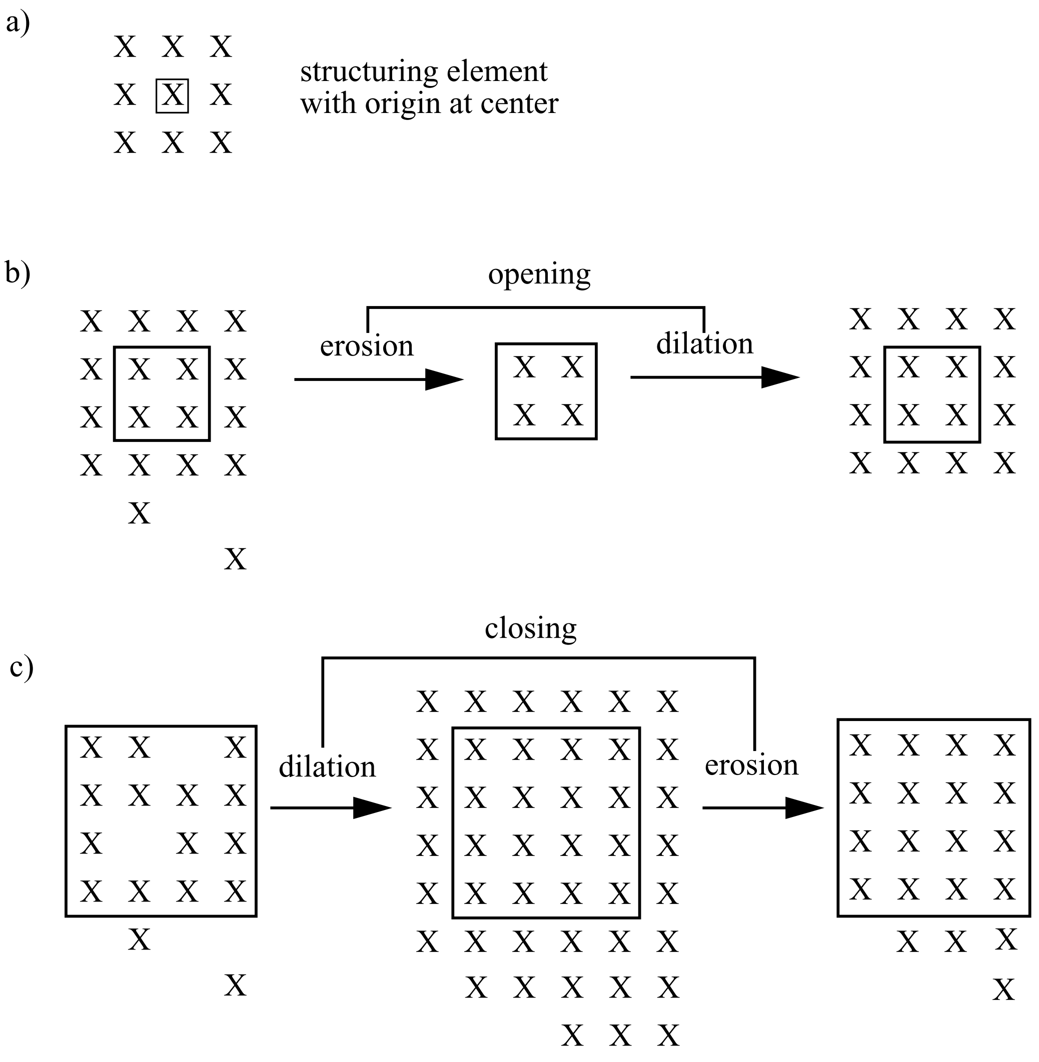


Figure 2. Morphological processing. The structuring element in (a) is centered on each pixel in the image and pixel values are changed as follows. For erosion, an ON-valued center pixel is turned OFF if the structuring element is over one or more OFF pixels in the image.For dilation, an OFF-valued center pixel is turned ON if the structuring element is over one or more ON pixels in the image. In (b), erosion is followed by dilation; that combination is called opening. One can see that the isolated pixel and the spur have been removed in the final result. In (c), dilation is followed by erosion; that combination is called closing. One can see that the hole is filled, the concavity on the border is filled, and the isolated pixel is joined into one region in the final result.

conditions are met:

$$(c = 1) AND [(n > 3k - 4) OR ((n = 3k - 4) AND (r = 2))]$$

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. An example is shown in Figure 3.

The kFill filter is designed specifically for text images 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^o 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. Since this filter is designed for fabricated symbols, text, and graphics, it is not appropriate for binarized pictures where less regularly shaped regions and dotted shading (halftone) are prevalent. A drawback of this filter — and of processes that iterate several times over the entire image in general — is that the processing time is expensive. Whether the expenditure of applying a filter such as this in the preprocessing step is justified depends on the input image quality and the tolerance for errors due to noise in subsequent steps.

Most document processing systems perform rudimentary noise reduction by passing 3x3 filter masks across the image to locate isolated ON and OFF pixels. For more extensive descriptions of these techniques in document systems, see [Modayur 1993] for use of morphology in a music reading system, and [Story 1992] for the use of kFill in an electronic library system.

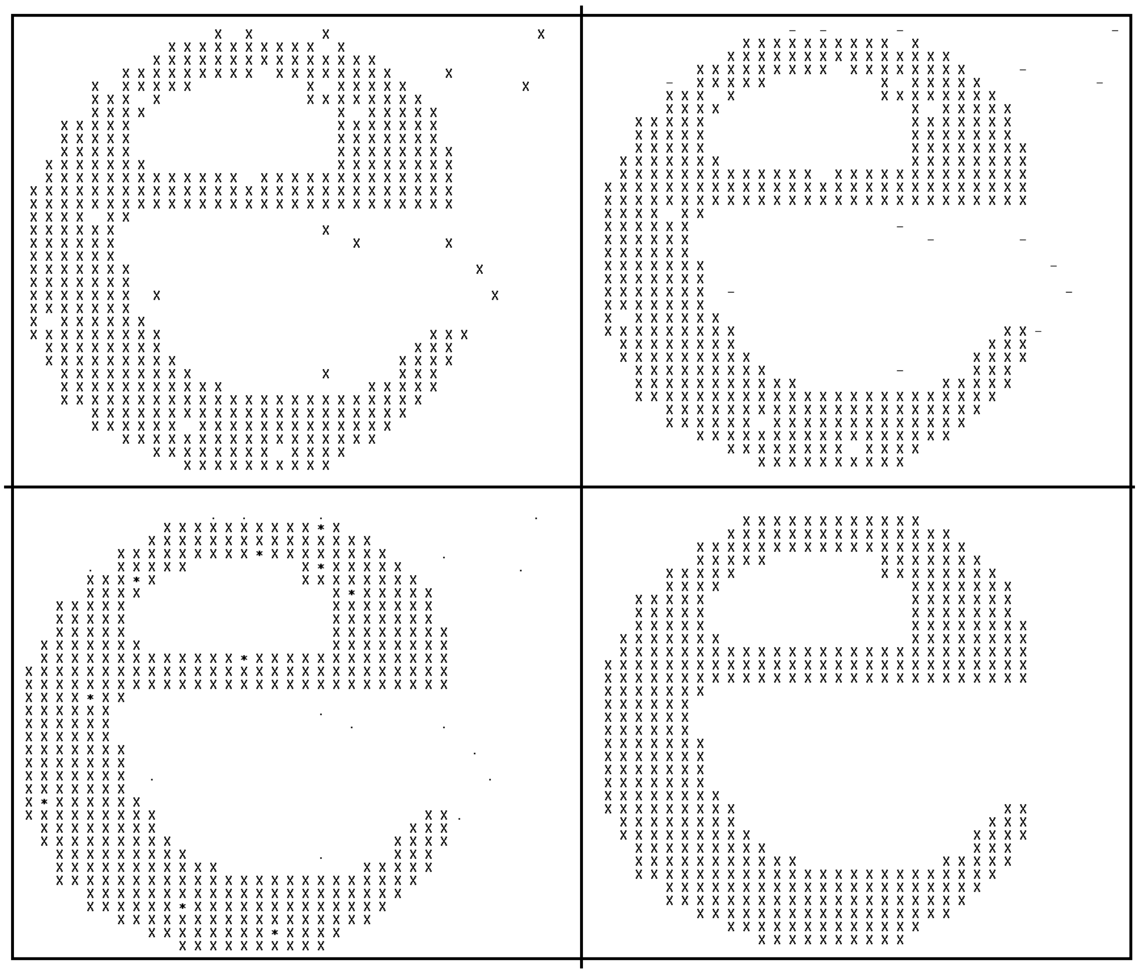


Figure 3. Results of kFill filter for salt-and-pepper noise reduction. The original noisy image is shown at the top left. The top right shows ON pixel removal, the bottom left shows OFF pixel filling, and the final image is shown in the bottom right.

References

1. J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, London, 1982.
2. R.M. Haralick, S.R. Sternberg, X. Zhuang, “Image analysis using mathematical morphology”, IEEE Trans. PAMI, Vol 9, July 1987, pp. 532-550.
3. R. M. Haralick, L. G. Shapiro, *Computer and Robot Vision,* Addison-Wesley, Reading, Mass., 1992.

4. K. Preston, Jr., M.J.B. Duff, S. Levialdi, P.E. Norgren, J-i. Toriwaki, “Basics of cellular logic with some applications in medical image processing”, Proc. IEEE, May, 1979, pp. 826-855.

5. C-C. Shih, R. Kasturi, “Generation of a line-description file for graphics recognition”', Proc. SPIE Conf. on Applications of Artificial Intelligence, 937:568-575, 1988.

6. L. O'Gorman, “Image and document processing techniques for the RightPages Electronic Library System”', Int. Conf. Pattern Recognition (ICPR), The Netherlands, Sept. 1992, pp. 260-263.

7. B. R. Modayur, V. Ramesh, R. M. Haralick, L. G. Shapiro, “MUSER - A prototype music score recognition system using mathematical morphology”, Machine Vision and Applications, Vol. 6, No. 2, 1993, pp.

8. G. Story, L. O'Gorman, D. Fox, L.L. Schaper, H.V. Jagadish, “The RightPages image-based electronic library for alerting and browsing”, IEEE Computer, Vol. 25, No. 9, Sept., 1992, pp. 17-26.

2.4: Thinning and Distance Transform

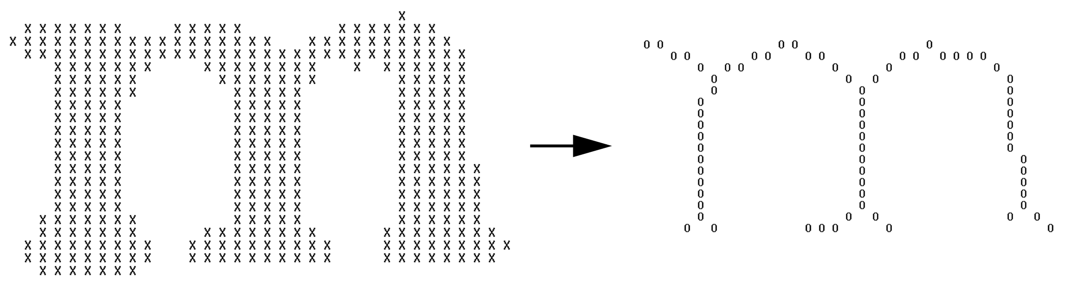
Keywords: thinning, skeletonizing, medial axis transform, distance transform

2.4.1: Thinning

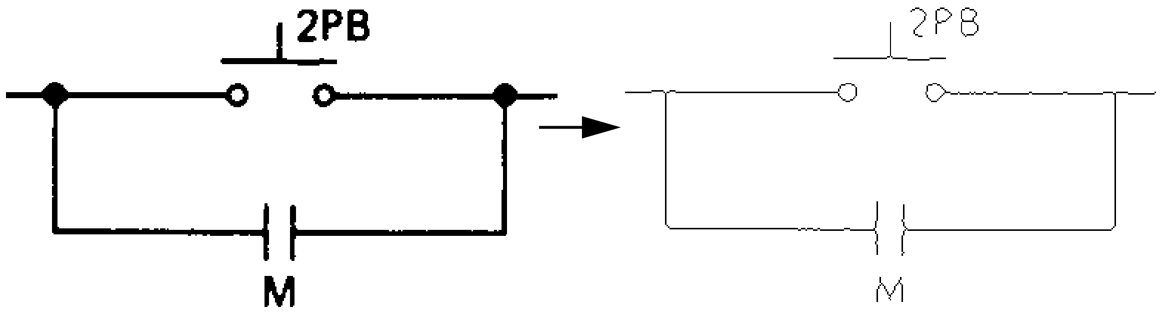
Thinning is an image processing operation in which binary valued image regions are reduced to lines that approximate the center lines, or skeletons, of the regions. The purpose of thinning is to reduce the image components to their essential information so that further analysis and recognition are facilitated. For instance, the same words can be handwritten with different pens giving different stroke thicknesses, but the literal information of the words is the same. For many recognition and analysis methods where line tracing is done, it is easier and faster to trace along one-pixel wide lines than along wider ones. Although the thinning operation can be applied to binary images containing regions of any shape, it is useful primarily for "elongated" shapes versus convex, or "blob-like" shapes. Thinning is commonly used in the preprocessing stage of such document analysis applications as diagram understanding and map processing. In Figure 1, some images are shown whose contents can be analyzed well due to thinning, and their thinning results are also shown here.

Note should be made that thinning is also referred to as "skeletonizing" and core-line detection in the literature. We will use the term "thinning" to describe the procedure, and thinned line, or skeleton, to describe the results. A related term is the "medial axis". This is the set of points of a region in which each point is equidistant to its two closest points on the boundary. The medial axis is often described as the ideal that thinning approaches. However, since the medial axis is defined only for continuous space, it can only be approximated by practical thinning techniques that operate on a sampled image in discrete space.

The thinning requirements are formally stated as follows: 1) connected image regions must thin to connected line structures, 2) the thinned result should be minimally eight-connected (explained below), 3) approximate endline locations should be maintained, 4) the thinning results should approximate the medial lines, and 5) extraneous spurs (short branches) caused by thinning should be minimized. That the results of thinning must maintain connectivity as specified by requirement 1 is essential. This guarantees an equal number of thinned connected line structures as the number of connected regions in the original image. By requirement 2, we stipulate that the resulting lines a)



b)



c)

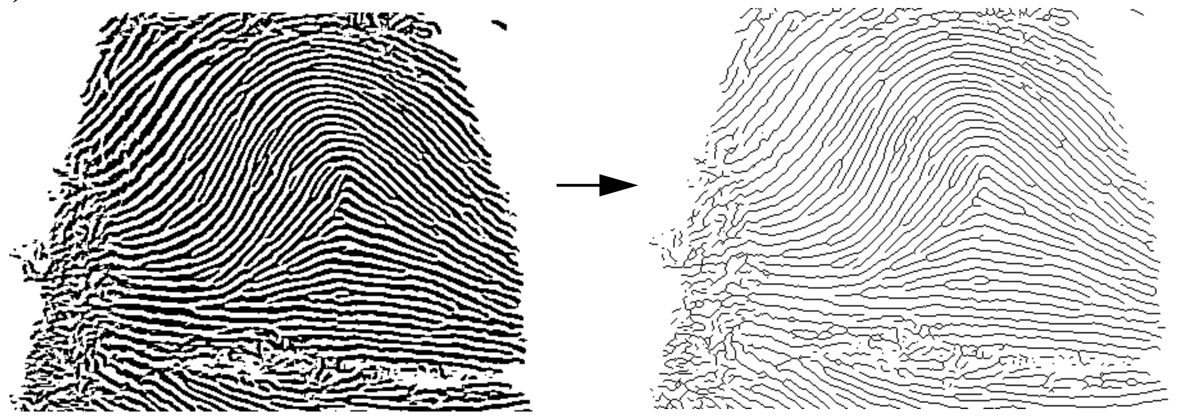


Figure 1. Original images on left and thinned image results on right. a) The letter “m”. b) A line diagram. c) A fingerprint image.

should always contain the minimal number of pixels that maintain eight-connectedness. (A pixel is considered eight-connected to another pixel if the second pixel is one of the eight closest neighbors to it.) Requirement 3 states that the locations of endlines should be maintained. Since thinning can be achieved by iteratively removing the outer boundary pixels, it is important not to also iteratively remove the last pixels of a line. This would shorten the line and not preserve its location. Requirement 4 states that the resultant thin lines should best approximate the medial lines of the original regions. Unfortunately, in digital space, the true medial lines can only be approximated. For instance, for a 2-pixel wide vertical or horizontal, the true medial line should run at the half-pixel spacing along the middle of the original. Since it is impossible to represent this in digital image space, the result will be a single line running at one side of the original. With respect to requirement 5, it is obvious that noise should be minimized, but it is often difficult to say what is noise and what isn't. We don't want spurs to result from every small bump on the original region, but we do want to recognize when a somewhat larger bump is a feature. Though some thinning algorithms have parameters to remove spurs, we believe that thinning and noise removal should be performed separately. Since one person's undesired spur may be another's desired short line, it is best to perform thinning first, then, in a separate process, remove any spurs whose length is less than a specified minimum.

The basic iterative thinning operation is to examine each pixel in an image within the context of its neighborhood region of at least *3 x 3* pixels and to "peel" the region boundaries, one pixel layer at a time, until the regions have been reduced to thin lines. (See [Hilditch 1969] for basic *3 x 3* thinning, and [O'Gorman 1990] for generalization of the method to *k x k* sized masks.) This process is performed iteratively — on each iteration every image pixel is inspected, and single-pixel wide boundaries that are not required to maintain connectivity or endlines are erased (set to OFF). In Figure 2, one can see how, on each iteration, the outside layer of 1-valued regions is peeled off in this manner, and when no changes are made on an iteration, the image is thinned.

Unwanted in the thinned image are isolated lines and spurs off longer lines that are artifacts due to the thinning process or noise in the image. Some thinning methods, e.g. [Arcelli and di Baja 85], require that the binary image is noise filtered before thinning because noise severely degrades the effectiveness and efficiency of this processing. However noise can never be totally removed, and, as mentioned, it is often difficult to distinguish noise from the signal in the earlier stages. An alternative approach is to thin after rudimentary noise reduction, then perform noise reduction with higher level information. Segments of image lines between endpoints and junctions are found, and descriptive parameters (length, type as classified by junctions or endlines at the ends of the segment, average curvature, absolute location, etc.) are associated with them. This descriptive and

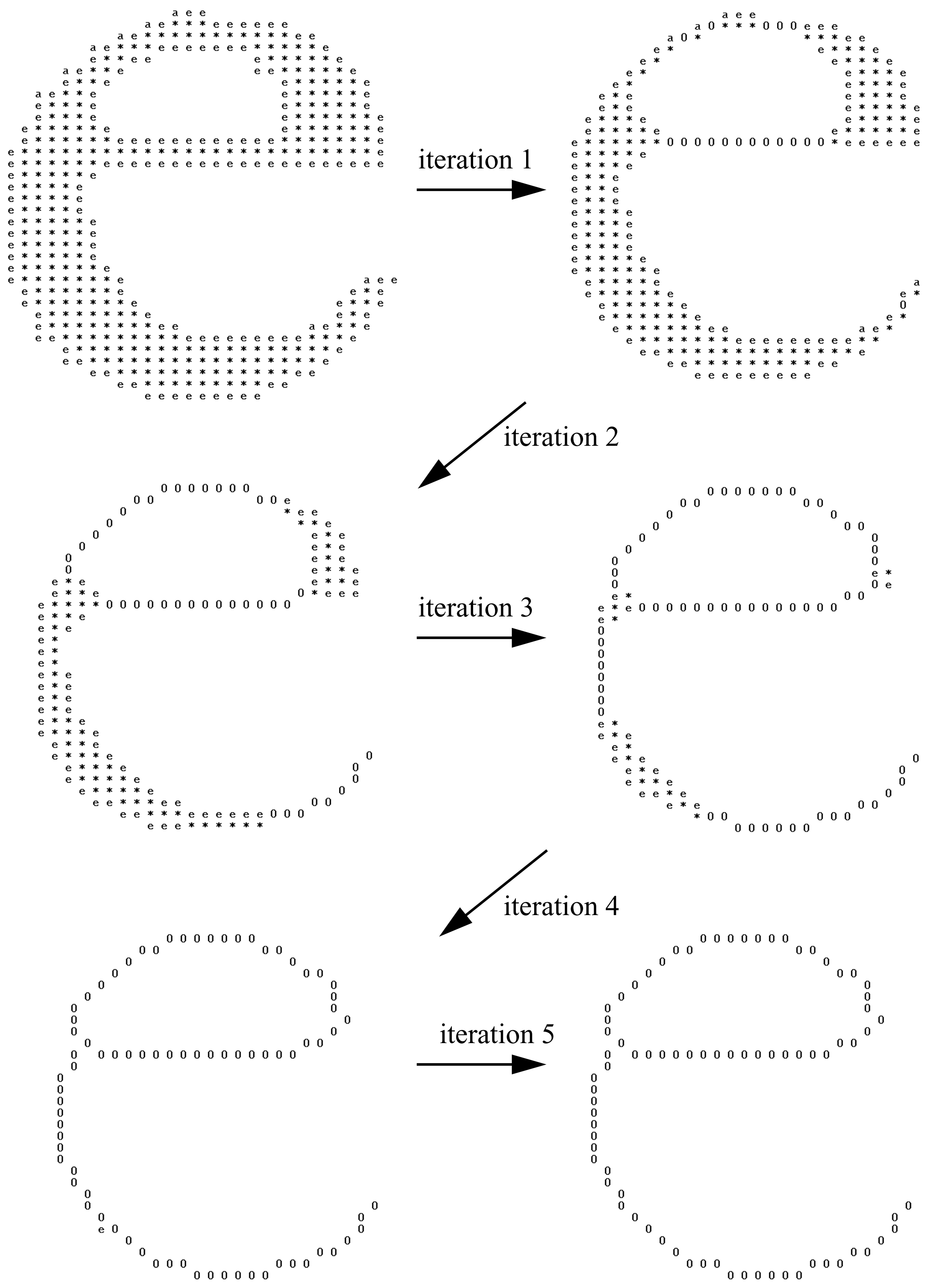


Figure 2. Sequence of five iterations of thinning on letter “e”. On each iteration a layer of the outer boundaries is peeled off. On iteration 5, the end is detected because no pixels change.

contextual information is then used to remove the line artifacts.

Instead of iterating through the image for a number of times that is proportional to the maximum line thickness, thinning methods have been developed to yield the result in a fixed number of steps [Arcelli and di Baja 85, Arcelli and di Baja 89, Sinha 87]. This is computationally advantageous when the image contains thick objects that would otherwise require many iterations. For these non-iterative methods, skeletal points are estimated from distance measurements with respect to opposite boundary points of the regions (see next section on distance transformation). Some of these methods require joining the line segments after thinning to restore connectivity, and also require a parameter estimating maximum thickness of the original image lines so that the search for pairs of opposite boundary points is spatially limited. In general, these non-iterative thinning methods are less regularly repetitive, not limited to local operations, and less able to be pipelined, compared to the iterative methods; and these factors makes their implementation in special-purpose hardware less appropriate.

Algorithms have also been developed for extracting thin lines directly from gray-level images of line drawings by tracking along gray-level ridges, that is without the need for binarization [Watson et al. 84]. These have the advantage of being able to track along ridges whose peak intensities vary throughout the image, such that binarization by global thresholding would not yield connected lines. However, a problem with tracking lines on gray-scale images is following false ridges (the gray-scale equivalent of spurs), which results in losing track of the main ridge or requires computationally expensive backtracking. Binarization and thinning are the methods most commonly used for document analysis applications because they are well understood and relatively simple to implement.

For recent overview papers on thinning, see [Lam 1992, Lam 1995]. For a reference on thinning applied specifically to documents, see [Eckhardt 1991].

2.4.2: Distance Transformation

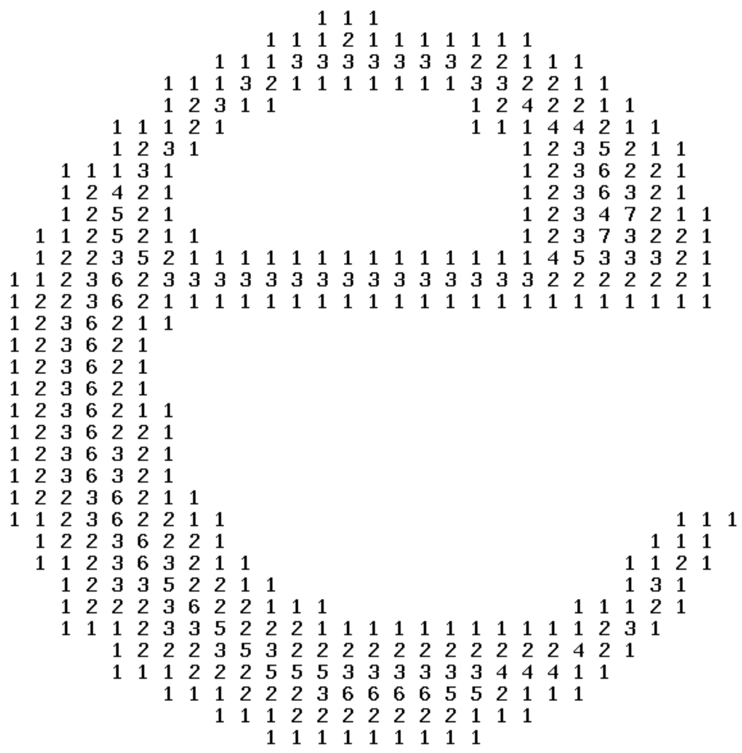
The distance transform is a binary image operation in which each pixel is labeled by the shortest distance from it to the boundary of the region within which it is contained. One way this can be used is to determine the shortest path from a given interior point to the boundary. It can also be used to obtain the thinned image by retaining only ridges of maximum local distance measures.

This thinned image, complete with distance values, has more information than simply the thinned image without distance information. It can be used as a concise and descriptive representation of the original image from which line widths can be obtained or the original image can be reconstructed. Results of the distance transform are shown in Figure 3.

There are two general approaches to obtaining the distance transformation. One is similar to the iterative thinning method described above. On each iteration, boundaries are peeled from regions. But, instead of setting each erased pixel value to OFF, they are set to the distance from the original boundary. Therefore, on the first iteration (examining *3 x 3* sized masks), erased boundaries are set to zero. On the second iteration, any erased core pixels will have a distance of 1 for vertical or horizontal distance to a boundary point, or $\sqrt{2}$ for diagonal distance to a boundary point. On the third and subsequent iterations, each pixel's distance value is calculated as the sum of the smallest distance value of a neighboring pixel plus its distance to that neighbor. When a core can no longer be thinned, it is labeled with its distance to the closest boundary.

The second approach requires a fixed number of passes through the image. To obtain the integer approximation of the Euclidean distance, two passes are necessary. The first pass proceeds in raster order, from the top row to the bottom row, left to right on each row. The distances are propagated in a manner similar to that above, but because the direction of the raster scan is from top left to bottom right, these first iteration values are only intermediate values— they only contain distance information from above and to the left. The second pass proceeds in reverse raster order, from bottom right to top left, where the final distance values are obtained now taking into account the distance information from below and to the right as well. For further treatments of distance transformations, see [Borgefors 1984, Borgefors 1986, Arcelli and di Baja1992].

The iterative method is a natural one to use if iterative thinning is desired. As mentioned above, thinning is only appropriate for elongated regions, and if the distance transform of an image containing thick lines or more convex regions is desired, the fixed-pass method is more appropriate. The fixed-pass method can also be used as a first step toward thinning. For all of these distance transformation methods, since distance is stored in the pixel, attention must be paid that the distance does not exceed the pixel word size, usually a byte. This problem is further exacerbated since floating point distance is approximated by integer numbers usually by scaling the floating point number up (e.g. 1.414 would become 14). This word size consideration is usually not a



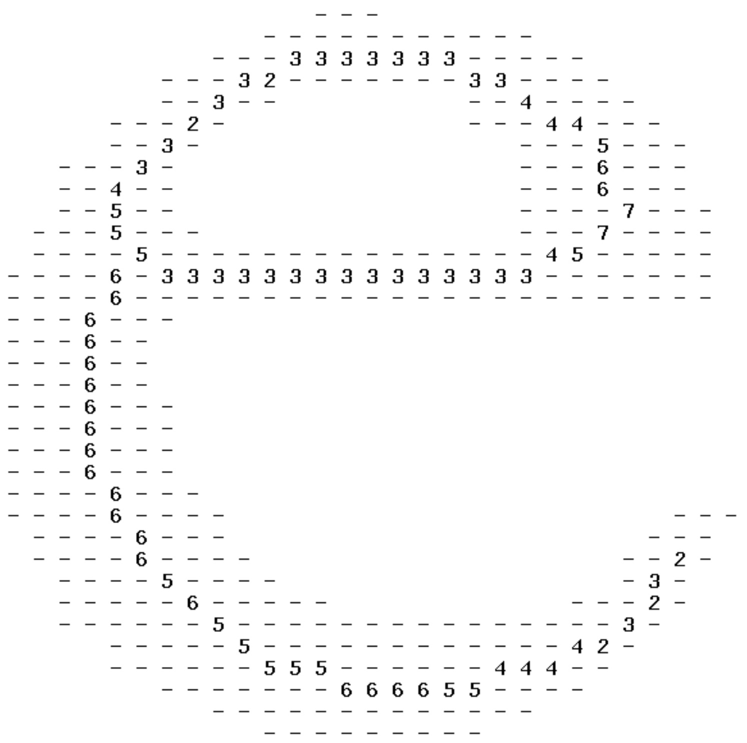


Figure 3. Distance transform. Top letter "e" shows pixel values equal to distance to closest border, except for midline pixels (shown in lower picture), which show total width at the midline.

problem for images of relatively thin, elongated regions, but may be a problem for larger regions.

Thinning is available on virtually all commercial graphics analysis systems. The particular method varies — there are many, many different thinning methods — but it is usually iterative and uses 3x3 or 3x4 sized masks. Most systems take advantage of a fast table look-up approach for the mask operations. These are implemented in software or in hardware for more specialized (faster and more expensive) machines.

References

1. C.J. Hilditch, "Linear skeletons from square cupboards", Machine Intelligence 4, 1969, pp. 403-420.
2. L. O'Gorman, "k x k Thinning", CVGIP, Vol. 51, pp. 195-215, 1990.
3. C. Arcelli, G. Sanniti di Baja, "A width-independent fast thinning algorithm", IEEE Trans. Pattern Recognition and Machine Intelligence, Vol. PAMI-7:4, 1985, pp. 463-474.
4. C. Arcelli, G. Sanniti di Baja, "A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform", IEEE Trans. Pattern Recognition and Machine Intelligence, Vol. 11:4, 1989, pp. 411-414.
5. R. M. K. Sinha, "A width-independent algorithm for character skeleton estimation", Computer Vision, Graphics, and Image Processing, Vol. 40, 1987, pp. 388-397.
6. L.T. Watson, K. Arvind, R.W. Ehrich, R.M. Haralick, "Extraction of lines and drawings from grey tone line drawing images", Pattern Recognition, 17(5):493-507, 1984.
7. L. Lam, S-W. Lee, C.Y. Suen, "Thinning methodologies — A comprehensive survey", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 14, No. 9, Sept. 1992, pp. 869-885.
8. L. Lam, C.Y. Suen, "An evaluation of parallel thinning algorithms for character recognition," IEEE Trans. Pattern Recognition and Machine Intelligence, Vol. 17, No. 9, Sept. 1995, pp. 914-919.
9. U. Eckhardt, G. Maderlechner, "Thinning for document processing," Int. Conf. Document Analysis and Recognition, France, 1991, pp. 490-498.

10. G. Borgefors, “Distance transformations in arbitrary dimensions”, Computer Vision, Graphics, and Image Processing, Vol. 27, 1984, pp 321-345.

11. G. Borgefors, “Distance transformations in digital images”, Computer Vision, Graphics, and Image Processing, Vol. 34, 1986, pp. 344-371.

12. C. Arcelli, G.S. diBaja, “Ridge points in Euclidian distance maps,” Pattern Recognition Letters, Vol. 13, 1992, pp. 237-243.

2.5: Chain Coding and Vectorization

Keywords: chain code, Freeman chain code, Primitives Chain Code (PCC), line and contour compression, topological feature detection, vectorization

2.5.1: Chain Coding

When objects are described by their skeletons or contours, they can be represented more efficiently than simply by ON and OFF valued pixels in a raster image. One common way to do this is by chain coding, where the ON pixels are represented as sequences of connected neighbors along lines and curves. Instead of storing the absolute location of each ON pixel, the direction from its previously coded neighbor is stored. A neighbor is any of the adjacent pixels in the 3 x 3 pixel neighborhood around that center pixel (see Figure 1). There are two advantages of coding by direction versus absolute coordinate location. One is in storage efficiency. For commonly sized images larger than 256x256, the coordinates of an ON-valued pixel are usually represented as two 16 bit words; in contrast, for chain coding with eight possible directions from a pixel, each ON-valued pixel can be stored in a byte, or even packed into three bits. A more important advantage in this context is that, since the chain coding contains information on connectedness within its code, this can facilitate further processing such as smoothing of continuous curves, and analysis such as feature detection of straight lines.

3	2	1
4	X	0
5	6	7

Figure 1. For a 3 x 3 pixel region with center pixel denoted as X, figure shows codes for chain directions from center pixel to each of eight neighbors: 0 (east), 1 (northeast), 2 (north), 3 (northwest), etc.

A further explanation is required regarding connectedness between pixels. The definition of connected neighbors that we will use here is called eight-connected. That is, a chain can connect from one pixel to any of its eight closest neighbors in directions 0 to 7 in Figure 1. Other definitions of connectedness are also used in the literature. The most common is four-connected, where a pixel can be connected to any of its four closest neighbors in the directions 0, 2, 4, or 6. An advantage of the four-connected chain is that each of its four chain directions has the same distance of one pixel spacing, versus eight-connected chains where there are two distances, 1 and $\sqrt{2}$ pixel spacings. The primary advantage of the eight-connected chain is that it more closely represents diagonal lines and portions of lines, that is, without the horizontal and vertical stair-steps to which the four-connected code is limited. Eight-connected codings also yield more concise representations.

The Freeman chain code is a widely used chain coding method [Freeman 1974]. Coding is accomplished in the following manner. A raster search is made from the top left of the image to the bottom right, examining each pixel. When an ON-valued pixel is found, the coordinate location of this pixel is stored, that pixel is set to OFF in the image, and chain coding is begun. The direction code is stored for each connection from the current pixel to a neighboring pixel, the current pixel is set to OFF, and this is done for all connected pixels until the end of a line or until a closed loop rejoins. If there is a branch in the line, one of the branching neighbors is chosen arbitrarily. The end of the chain is indicated by adding a code that is the inverse direction of the previous code (e.g. 0, 4; or 1, 5; etc.), and since this is otherwise impossible, this indicates the chain end. See Figure 2 for an example of a Freeman line coded line structure.

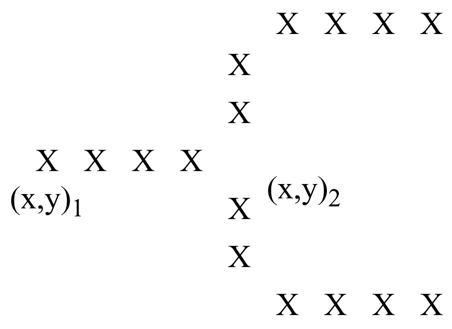


Figure 2. Pixels of a branching line structure.The Freeman chain code for this is: $\{(x,y)_1, 0, 0, 0, 1, 2, 1, 0, 0, 0, 4, (x,y)_2, 6, 7, 0, 0, 0, 4\}$, where the top branch is coded first, then the bottom beginning with the pixel at $(x,y)_2$.

Though the Freeman chain code is highly effective for compression of line images, it was designed for contours, without any provision for maintaining branching line structures (each branch is coded as a separate chain). This is fine for compression, but for image analysis it is important to retain the complete line structure with all its branches and to know the topology at each junction. In [Harris et al. 1982], the skeleton is coded with pixel values of 0, 1, 2, and 3 for background pixels, terminal (end of line) pixels, intermediate pixels, and junction pixels respectively. This, combined with chaining allows line segments and their interconnections to be easily determined.

Another method, the Primitives Chain Code (PCC) also preserves topological features [O'Gorman 1992]. PCC contains codes for the following features: ends of lines, bifurcation and cross junctions, and breaks indicating the beginning of coding within a closed contour. With these additional features, subsequent pattern recognition steps are facilitated. This code also usually results in higher compression than the other chain code techniques because it efficiently limits the number of codewords to the number of eight-connected possibilities, and packs them efficiently. The PCC has been applied to analysis of fingerprints, maps, and engineering diagrams, all of which contain branches as important features. PCC coding is accomplished in a similar manner to Freeman chain coding. That is, a raster scan for ON-valued pixels is first done, and any lines are followed and encoded. The difference is that features are also encoded, and because of the presence of codes for these features, different line topologies can be recognized from the code. For the example in Figure 2, the PCC code is: $(x,y)_1$, 41, **b**, 15, 41, **e**, 61, 41, **e**. The bold characters indicate PCC features and the non-bold numbers indicate portions of up to 3 chain directions between the features. Here, the features are seen easily as a branch junction (b) followed by two endlines (e). The other codes are table look-up values for the connections between features: 41 is equivalent to {0,0,0} in Freeman chain coding, 15 is equivalent to {1,2,1}, and 61 is equivalent to {6,7,0}.

2.5.2: Vectorization

An alternative to thinning and chain coding is to represent image lines by the straight line segments that can be drawn within the original thicker lines. This approach is called vectorization. In one vectorization approach [Ramachandran 1980], horizontal runs within lines are first found, adjacent runs on subsequent scan lines are grouped together, and piecewise-straight segments are fit along these connected regions. In [Ejiri et al. 1990], vectorization is performed at the initial digitizing level by hardware that locates and tracks long straight line segments (see Figure 3). One advantage of vectorization is that, since long lines are searched and found, there are fewer spurs than usually result by thinning and chain coding. It should be noted that these straight segments found by vectorization will not usually correspond to complete straight lines as intended in the original drawing. This is due in part to the stair-stepping that results from straight lines being digitized at slight angles. To determine complete line features, subsequent analysis is necessary. An example of post-processing of vectorized data to better match objects found against the object model is described in [Hori and Okazaki 1992].

Vectorization is intermediate in purpose between the thinning and chain coding procedures, and polygonalization that will be described in the next chapter. Thinning and chain coding attempt to exactly represent the line paths; polygonalization attempts to approximate line features; and vectorization yields something between the two. If the vectorization tolerance parameter is zero, the results are closest to those of thinning and chain coding. If the tolerance is such that a piecewise straight approximation of straight and curved lines results, then this is similar to polygonalization. In practice, the term “vectorization” is often used loosely for techniques that span this range. To avoid this confusion we use “chain coding” and “polygonalization” as terms to describe families of methods, and use vectorization only for the so-called hardware vectorizers mentioned here.

A final note should be made on automatic and human-assisted vectorization. Often the image quality is such that automatic line extraction is too error-prone to be useful. For instance, for conversion of engineering drawings to CAD format from older blueprints, the cost of a human operator correcting all the errors made by the computer may be higher than manually entering the entire diagram in the first place. For cases such as this, a computer can be used to facilitate manual conversion. One example of this is Fastrak [Fulford 1981], an interactive line following digitizer in which a human operator simply “leads” the computer along the path of the line using an input device such as a mouse. Most, or all, practical commercial systems for diagram entry employ a combination of image analysis and computer-aided manual correction.

Chain coding or vectorization is performed in virtually every commercially available system for storing and analyzing graphics documents. The Freeman code is most prevalent, though systems often incorporate proprietary vectorization techniques. We make reference [Hori and Okazaki 1992] to a paper using vectorization for map processing.

Other Figure Captions

Figure 3. The diagram shows the radial searching procedure to determine long, straight lines for vectorization [Ejiri et al. 90].

References

1. H. Freeman, “Computer processing of line drawing images”, Computing Surveys, Vol. 6, No. 1, 1974, pp. 57-98.
2. J.F. Harris, J. Kittler, B. Llewellyn, G. Preston, “A modular system for interpreting binary pixel representations of line-structured data”, in Pattern Recognition: Theory and Applications, J. Kittler, K.S. Fu, L.F. Pau (eds.), D. Reidel Publishing Co. pp. 311-351, 1982.
3. L. O'Gorman, “Primitives Chain Code”, in “Progress in Computer Vision and Image Processing”, edited by A. Rosenfeld and L. G. Shapiro, Academic Press, San Diego, 1992 pp. 167-183.
4. K. Ramachandran, “A coding method for vector representation of engineering drawings”, Proc. IEEE, 68:813-817, 1980.
5. M. Ejiri, S. Kakumoto, T. Miyatake, S. Shimada, K. Iwamura, “Automatic recognition of engineering drawings and maps”, in Image Analysis Applications, R. Kasturi and M.M. Trivedi (eds.), Marcel Dekker, 1990
6. O. Hori, A. Okazaki, “High quality vectorization based on a generic object model”, in Structured Document Image Analysis, H.S. Baird, H. Bunke, K. Yamamoto (eds.), Springer-Verlag, 1992, pp. 325-339.
7. N.C. Fulford, “The Fastrak automatic digitizing system”, Pattern Recognition, 14:65-74, 1981.

2.6: Binary Region Detection

Keywords: region extraction, binary segmentation, region growing, blob detection, blob coloring, connected component labeling, contour detection, line adjacency graph (LAG)

Before feature-level analysis can take place, segmentation must be performed to detect individual regions (also called objects or blobs) in the image. For gray-scale and color images, this is sometimes difficult because the objects may blend into the background or there may be overlap. However for binary images, it is a more straightforward procedure to find each group of neighboring ON pixels. Thus, the character, “a” comprises one region, and the character “i” two regions. A dashed line is comprised of as many regions. Once regions have been found, features can be determined from them, and recognition can be made of the region as one or part of a particular character or graphics component.

Region detection is analogous to chain coding preceded by thinning in that the objective of both procedures is to yield a single representation of a group of neighboring ON pixels. The difference is the applicability of each method. It has been mentioned that thinning and chain coding can be applied to any shape, but are most useful for elongated shapes. Region detection can also be applied to any shape, but it is useful especially for the rounder shapes where the results from thinning are less useful. For instance, a circular disk will thin to a single point (under perfect thinning conditions), and the size information will be lost in this thinned image; whereas region detection will retain this size information. Region detection is also useful for hole detection. For instance, thinning an image of Swiss cheese will result in many connected lines representing the cheese connections, however this may not be as useful as the result of contour detection that will have one contour for the boundary of the cheese and contours for each of the holes. As a counterexample, for the letter, “H”, thinning will yield the pertinent topological information that the symbol is made up of two vertical lines joined by a horizontal line; whereas region detection will yield only a single region with no topological information, and will require additional feature detection for character recognition. Since many document components are made from line strokes, thinning and chain coding are often used. However if the purpose is just to locate document components, or if the objects are truly blob-shaped, region detection is appropriate.

2.6.1: Contour Detection

Contour detection can be thought of as a reciprocal operation from thinning. Whereas thinning yields the inside skeletons, contour detection yields the outside boundaries, or contours. Since a single contour envelopes a single region, contour detection can be used for region detection. Contours are comprised of boundary, ON-valued pixels that border OFF-valued pixels. These contours can be found easily by examining each pixel within a *3 x 3* window, and if the center pixel is ON, and at least one of its neighborhood pixels is OFF, then the center pixel is a contour pixel and it is set to ON; all other pixels are set to OFF. Each resulting contour can then be chain coded to represent each region. Usually chain coding is done in one direction around ON regions (counter-clockwise) and in the opposite direction (clockwise) around holes. The result of contour detection is illustrated in Figure 1. (See [Cederberg 80, Pavlidis 82] for contour tracing algorithms.)

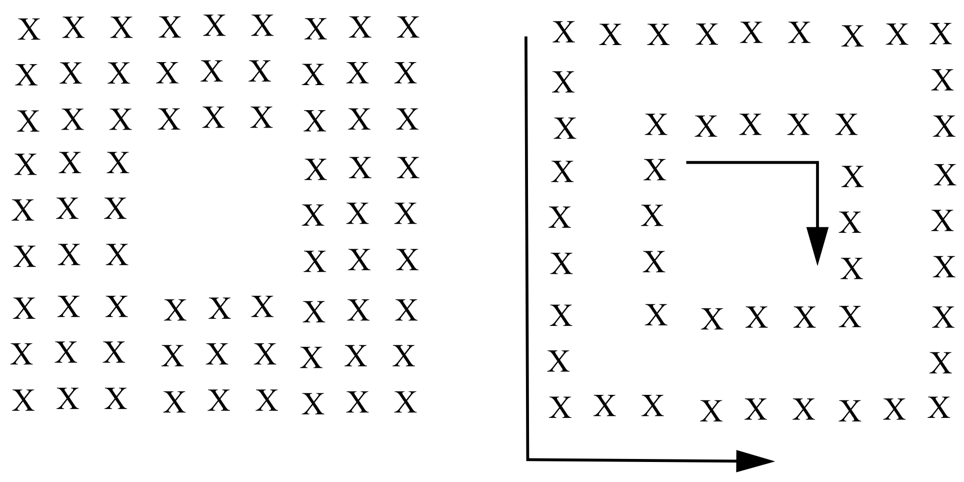


Figure 1. On the left is a region, where the ON-valued pixels are represented by "X"s. On the right is the contour of this region, where only the outer and inner boundaries remain ON. The arrows around the contour show the direction of coding.

The contour image can be used in a number of ways. For instance, the number of contours gives the number of regions (both ON-valued regions, and OFF-valued holes within ON-valued regions). The centroid of the boundary pixels gives a measure of the region location. The length of a contour indicates the enclosed region size. The length and enclosed area can be used to give a measure of how elongated or "fat" the region is. Curvature and corner features can be determined from the contour to determine the region shape. For other feature detection methods applicable to contours, see the following chapter.

2.6.2: Region Labeling

Regions can also be found by other techniques based on interior (versus contour) pixels. One way to locate regions is to perform connected component labeling, or region coloring, which results in each region of an image being assigned a different label, or color [Ronse and Divijver 84, Dillencourt et al. 92]. The method involves a two-pass process where pixels are individually labeled. The image is first scanned in raster order, and each pixel is examined. For each ON-valued pixel, the previously labeled pixels to the left and above are examined. If none is ON, then the pixel is set to a new label value. If one of these is ON, then the current pixel is given the same label as that pixel. If more than one are ON, then the pixel is set to one of the labels, and all labels are put in an equivalence class (that is, they are stored for later merging). At the end of this pass, the number of connected components is the number of equivalence classes, plus the number of labels not in an equivalence class. The second pass consists of first merging all labels in each equivalence class to be the same label, then reassigning all labeled pixels in the image to these final labels.

A method that relates to connected component labeling and to thinning is the line adjacency graph (LAG) [Pavlidis, 1982]. First, runs of ON pixels are found, where a run is a group of same-valued, adjacent pixels on the same row. Then for each run, if there is a run on an adjacent row in which one or more pixels are neighboring, these runs are said to be graph nodes and they are joined by graph branches. If these branches are visualized as segments drawn between the middle points of each pair of runs, the branches will form a structure similar to a skeleton (Figure 2). This skeleton will retain topological information of holes and will approximate large nodules in the shape, but because the method is dependent upon the orientation, it cannot be used to obtain a good approximation of the medial lines or of endlines.

Both contour detection and coloring operations are usually available on graphics analysis systems to be applied to the non-thin components.

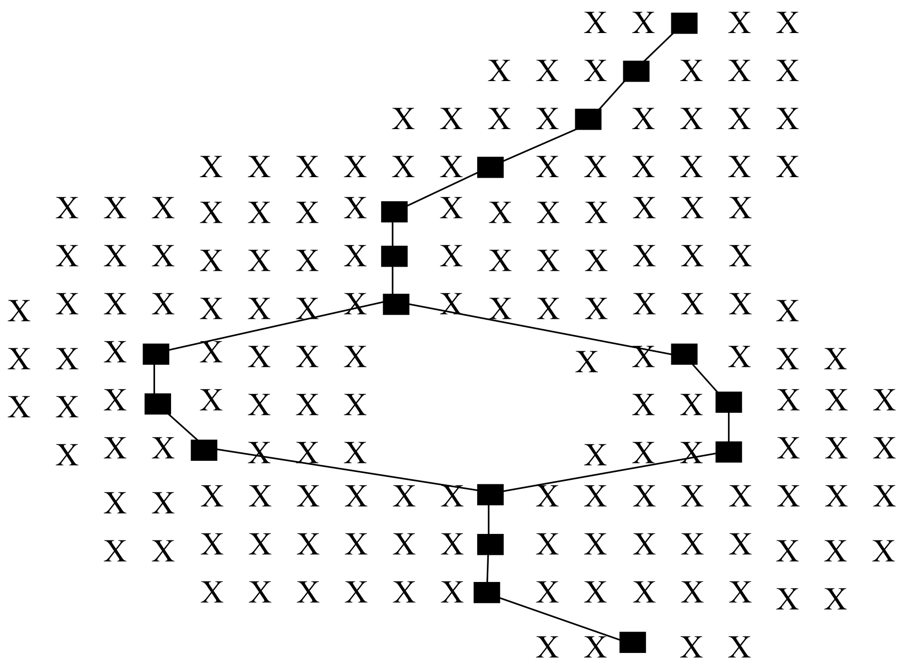


Figure 2. Line adjacency graph of region. The ON pixels are shown as "X"s. The middle of each row of ON pixels is shown as a filled box. The lines that join these midpoints of each ON row shows the line adja-

References

1. R. Cederberg, "On the Coding, Processing, and Display of Binary Images", Ph.D. Dissertation No. 57, Linkoping University, Dept. Electrical Eng., Linkoping, Sweden, 1980.
2. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Maryland, 1982, pp. 116-120 (LAG), 142-148 (contour detection).
3. C. Ronse, P.A. Divijver, *Connected Components in Binary Images: The Detection Problem*, Research Studies Press Ltd., Letchworth, England, 1984.
4. M.B. Dillencourt, H. Samet, M. Tamminen, "A general approach to connected-component labeling for arbitrary image representations", J. Association for Computing Machinery, Vol. 39, No. 2, April, 1992, pp. 253-280.

Chapter 3

Finding Appropriate Features

3.1: Introduction

After pixel-level processing has prepared the document image, intermediate features are found from the image to aid in the final step of recognition. An example of the difference between pixel data and features is as follows. A line diagram may be reduced to thinned lines and then stored as a chain code. Though this representation requires less storage size from the original image, still each codeword corresponds to a pixel, and there is no interpretation of pixels other than that they are ON or OFF. At the feature level, this thinned and chain-coded data is analyzed to detect straight lines and curves. This is a more informative representation that is also closer to how humans would describe the diagram — as lines and curves rather than as ON and OFF points. Besides features describing lines, we also describe in this chapter region features such as size and shape. These feature described in this chapter are used as intermediate descriptions that aid in the final recognition of characters for OCR, and lines and symbols for graphics analysis.

3.2: Polygonalization

Keywords: polygonalization, straight-line approximation, piecewise linear approximation

Polygonal approximation is one common approach to obtaining features from curves. The objective is to approximate a given curve with connected straight lines such that the result is close to the original, but that the description is more succinct. Polygonal approximation to the coast line of Britain is shown in Figure 1 [Wall 86]. The user can direct the degree of approximation by specifying some measure of maximum error from the original. In general, when the specified maximum error is smaller, a greater number of lines is required for the approximation. The effectiveness of one polygonalization method compared to another can be measured in the number of lines required to produce a comparable approximation, and also by the computation time required to obtain the result.

3.2.1: Iterative and Sequential Methods

The iterative endpoint fit algorithm [Ramer 72], is a popular polygonalization method whose error measure is the distance between the original curve and the polygonal approximation. The method begins by connecting a straight line segment between endpoints of the data. The perpendicular distances from the segment to each point on the curve are measured. If any distance is greater than a chosen threshold, the segment is replaced by two segments from the original segment endpoints to the curve point where distance to the segment is greatest. The processing repeats in this way for each portion of the curve between polygonal endpoints, and the iterations are stopped when all segments are within the error threshold of the curve. See Figure 2 for an illustration of this method.

The iterative endpoint fit algorithm is relatively time-consuming because error measures must be calculated for all curve points between new segment points on each iteration. Instead of this "overall" approach, segments can be fit starting at an endpoint, then fit sequentially for points along the line [Sklansky and Gonzalez 80, Williams 81]. That is, starting at an endpoint, a segment is drawn to its neighboring point, then the next, etc., and the error is measured for each. When the error is above a chosen threshold, then the segment is fit from the first point to the point previous to the above-threshold error. This process is continued starting at this point as before.

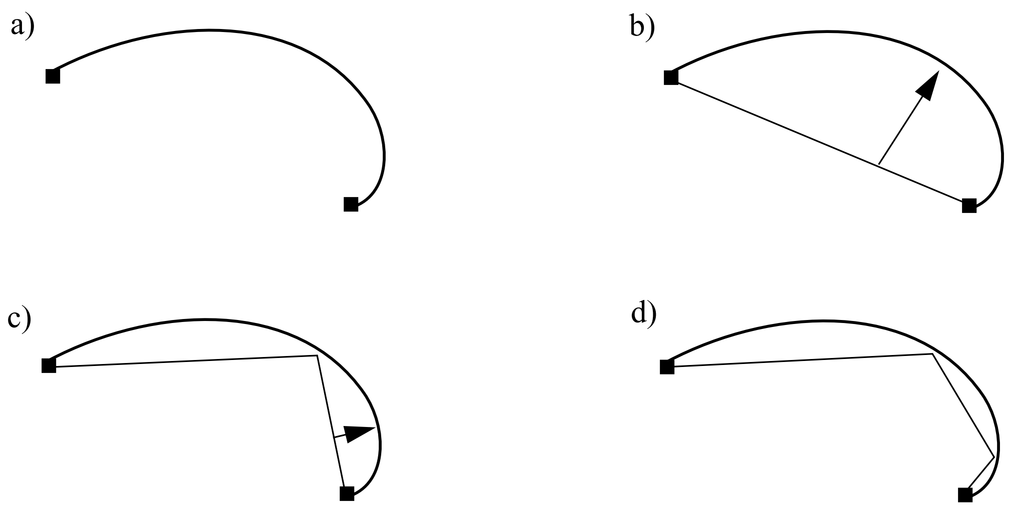


Figure 2. Illustration of the iterative endpoint algorithm. a) The original curve. b) A polygonal approximation is made between endpoints, and the furthest perpendicular distance from this line to the curve is found (line with arrowhead). c) Since this is greater than threshold, then two segments are fit to the curve at this point of greatest error, and the greatest distance between each new segment and the curve is found. d) Only one error is above threshold, so that segment is split, and the result is shown where all segments are within the threshold distance of the curve.

The error criterion used for this approach states that the straight line fit must be constrained to pass within a radius around each data point. See Figure 3 for an illustration of this method.

In another class of polygonalization techniques, area is used as a measure of goodness of fit [Wall and Danielsson 84]. (See Figure 4.) Instead of measuring the distance error, a scan-along technique is used where the area error between the original line and the approximation is accumulated. When this area measurement for a segment exceeds a preset value, the line segment is drawn to the previous curve point, and a new segment is begun. A similar area measure is used in [Imai and Iri 86], where a piece-wise linear approximation is made to a curve so as to cover a sequence of points by a minimum number of rectangles. The use of an area measure versus the worst-case perpendicular distance measure as described above is better from the standpoint of

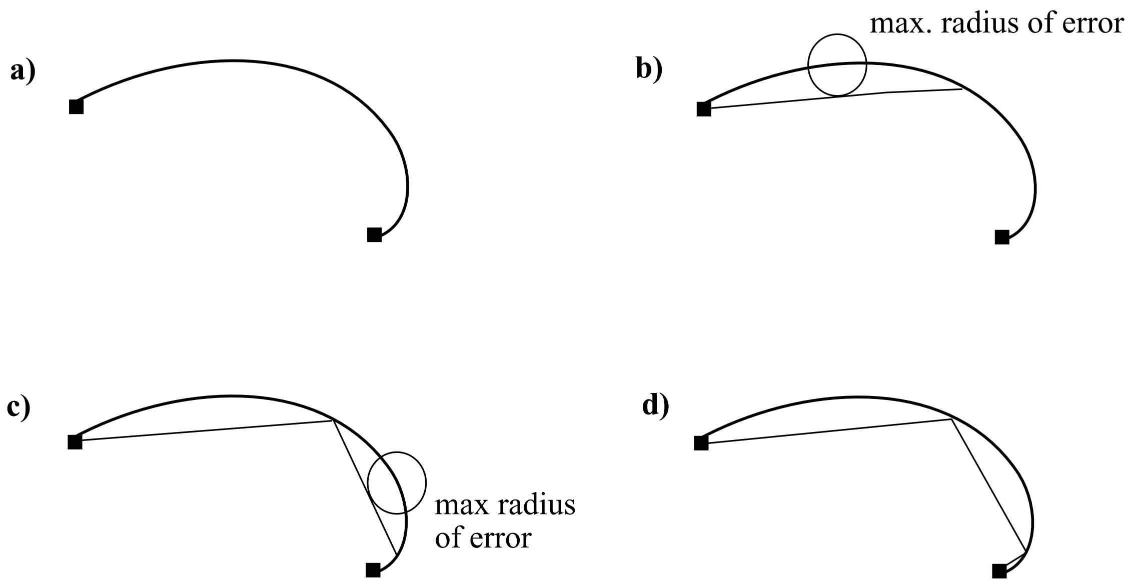


Figure 3. Illustration of sequential polygonal approximation employing distance as measure of error. a) The original curve. b) A polygonal approximation is made from the first endpoint to as far a point on the curve until the error (radius of circle between the curve and the approximating segment) is above threshold. c) A new segment is begun from the endpoint of the last, and it is drawn to as far a point until the error is again above threshold. d) A third segment goes from the previous to the endpoint of the original curve, and this is the final approximation.

giving a closer overall approximation to the original curve; in contrast, the use of a distance measure gives a closer approximation to sharp corners and curves, that is this gives better worst-case approximation.

3.2.2: Methods for Closer Approximation

The methods described above are usually adequate for the purpose of polygonalization. However, there are more sophisticated methods that can yield better approximations —at the expense of more computation, increased code complexity, or both. One of the drawbacks of most of the polygonal fit techniques is that the operations are not performed symmetrically with respect to

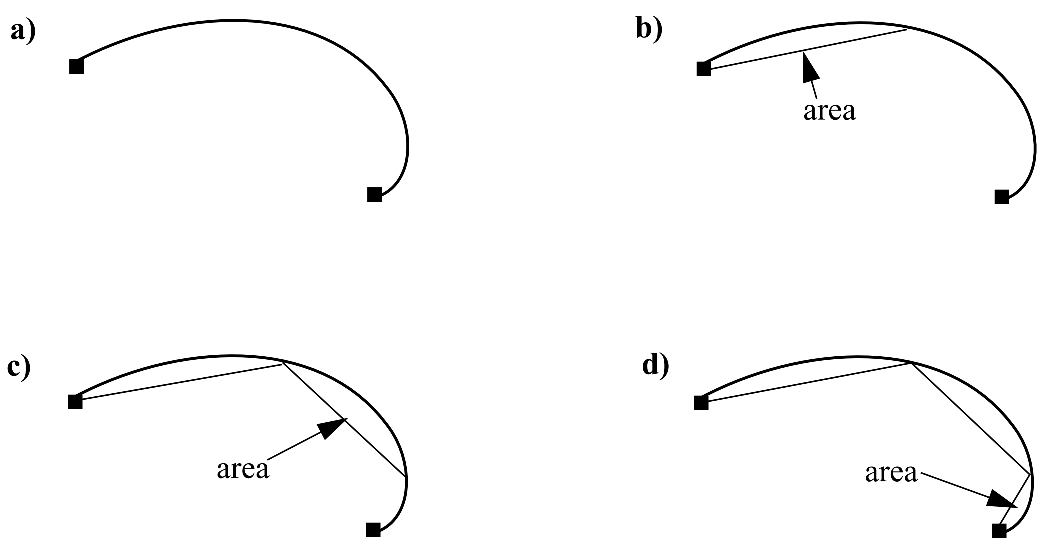


Figure 4. Illustration of sequential polygonal approximation employing area as measure of error. a) The original curve. b) A polygonal approximation is made from the first endpoint to as far a point on the curve until the error (area between the curve and the approximating segment) is above threshold. c) A new segment is begun from the endpoint of the last, and it is drawn to as far a point until the error is again above threshold. d) A third segment goes from the previous to the endpoint of the original curve, and this is the final approximation.

curve features such as corners and the centers of curves. The result is that the computed breakpoints between segments may be at different locations depending on starting and ending locations, or direction of operation of the line fit. Extensions can be made to some of the methods to produce fewer and more consistent breakpoint locations, but these procedures are usually iterative and can be computationally expensive. In choosing a particular method, the user must weigh the trade-off between the closeness of the fit and amount of computation.

If a closer polygonal approximation is desired, one method is to use a minimax measurement of error [Kurozumi and Davis 82]. Here, the line segment approximations are chosen to minimize the maximum distance between the data points and the approximating line segment. This method is especially useful for noisy curves such as curves fit to data with outlying (large linear error) data points. Though the minimax approach can give a better approximation, there is an added level of complexity for this method because segment endpoints must be adjusted to be within a gap tolerance of connecting segment endpoints.

One potential drawback of polygonal approximation is that the result is not usually unique, that is, an approximation of the same curve that begins at a different point on the curve will likely yield different results. The lack of a unique result will present a problem if the results of polygonalization are to be used for matching. One approach to reducing this problem is to find a starting point along the curve that will always yield similar results. In [Leu and Chen 88] the polygonal approximated is started simultaneously at positions along the curve where the arcs are closer to straight lines than their neighboring arcs. This is done by first calculating the maximum arc-to-chord deviation for each arc, and then finding the local minimum deviation by comparing each deviation with those of the neighboring arcs. Polygonal approximation is done by fitting lines between points whose maximum arc-to-chord distance is less than a given tolerance. Because an effort is made to position the segments with respect to true features in the original curve (i.e. straight lines and curves), this method straddles the boundary between polygonal approximation methods and the feature detection methods discussed in Section 3.4.

Besides polygonal approximation methods, there are higher order curve and spline fitting methods that achieve closer fits. These are computationally more expensive than most polygonalization methods, and can be more difficult to apply. We discuss these methods in detail in Section 3.3. It is important to stress that a polygonal fit just approximates the curve to some close error measure. If it is feature detection of true corners and curves that is desired, then the feature detection methods of Section 3.4 should be used rather than the methods in this section.

Commercial systems for engineering drawing and geographical map storage will usually provide the polygonalization operation. Sometimes polygonalization is used in drawing systems, often unbeknownst to the user, to store curves more efficiently. These systems use a very small value of the maximum error of approximation so that viewers percieve the curves as smooth rather than as straight line approximations.

One good reference describes a fast, sequential polygonalization using an area measure of error [Wall and Danielsson 1984]. This is easily implemented and will provide good approximation.

For smaller error (but longer computation), one of the minimax methods in the references can be used, for instance [Kurozumi, 1982].

Other Figure Captions

Figure 1. On the left is an outline of the coastline of Britain, and on the right is its polygonal approximation.

References

1. Wall, K., "Curve fitting based on polygonal approximation," Proc. 8th ICPR, pp. 1273-1275, Paris, 1986
2. Ramer, U.E., "An iterative procedure for the polygonal approximation of plane curves", Computer Graphics and Image Processing, 1:244-256, 1972
3. Sklansky, J. and V. Gonzalez, "Fast polygonal approximation of digitized curves," Pattern Recognition, 12:327-331, 1980
4. Williams, C.M., "Bounded straight-line approximation of digitized planar curves and lines," Computer Graphics and Image Processing, 16:370-381, 1981
5. Wall, K. and P.E. Danielsson, "A fast sequential method for polygonal approximation of digitized curves," Computer Graphics and Image Processing, 28:220-227, 1984
6. Imai, H. and M. Iri, "Computational-geometric methods for polygonal approximations of a curve," Computer Graphics and Image Processing, 36:31-41, 1986
7. Kurozumi, Y. and W.A. Davis, "Polygonal approximation by minimax method," Computer Graphics and Image processing, 19, 248-264, 1982
8. Leu, J.G. and L. Chen, "Polygonal approximation of 2-D shapes through boundary merging," Pattern Recognition Letters, 7:231-238, 1988

3.3: Critical Point Detection

Keywords: critical points, dominant points, curvature extrema, corners, curves, curvature, difference of slopes (DOS), *k*-curvature, Gaussian smoothing

The concept of “critical points” or “dominant points” derives from the observation that humans recognize shape in large part by curvature maxima in the shape outline [Attneuve 1954]. The objective in image analysis is to locate these critical points and represent the shape more succinctly by piecewise linear segments between critical points, or to perform shape recognition on the basis of the critical points. Critical point detection is especially important for graphics analysis of man-made drawings. Since corners and curves have intended locations, it is important that any analysis precisely locates these. It is the objective of critical point detection to do this — as opposed to polygonal approximation, where the objective is only a visually close approximation. In this section, we will describe methods for locating critical points. In addition to corner features, we will include circular curves as a type of critical feature; the critical points for curves are the transition locations along the line from straight lines into the curves.

3.3.1: Curvature Estimation Approaches

One approach for critical point detection begins with curvature estimation. A popular family of methods for curvature estimation is called the *k*-curvatures approach (also the difference of slopes, or DOS, approach) [Freeman and Davis 77, O'Gorman 88]. For these methods, curvature is measured as the angular difference between the slopes of two line segments fit to the data around each curve point. (See Figure 1.) Curvature is measured for all points along a line, and plotted on a curvature plot. For straight portions of the curve, the curvature will be low. For corners, there will be a peak of high curvature that is proportional to the corner angle. For curves, there will be a curvature plateau whose height is proportional to the sharpness of the curve (that is, the curvature is inversely proportional to the radius of curvature), and the length of the plateau is proportional to the length of the curve. To locate these features, the curvature plot is thresholded to find all curvature points above a chosen threshold value; these points correspond to features. Then, the corner is parameterized by its location and the curve by its radius of curvature and bounds, the beginning and end transition points from straight lines around the curve into the curve. The user must choose one method parameter, the length of the line segments to be fit along the data to determine the curvature. There is a tradeoff in the choice of this length. It should be as long as possible to smooth out effects due to noise, but not so long so as to also average out features. That is, the length should be chosen as the minimum arclength between critical points.

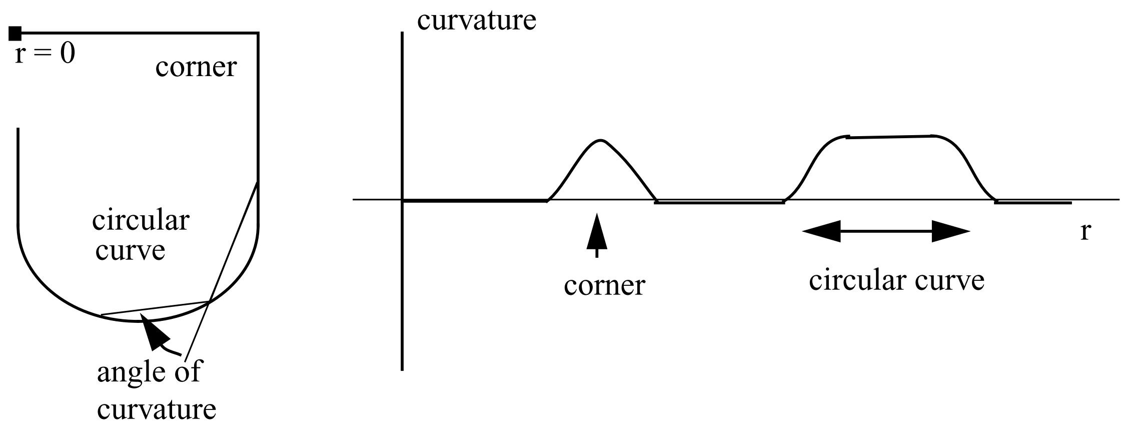


Figure 1. On the left is a curve with a corner and a circular curve feature. On the right is the curvature plot derived from this by the DOS method. The angle of curvature is shown on the curve for one point. On the curvature plot, the first portion has zero curvature for the horizontal straight line from r = 0. The corner is indicated by a peak on the curvature plot. The straight line from corner to the curve also has zero curvature. The rise and fall of the curvature plot around the plateau shows the transition point locations along the arc length into a curve. Finally, there is another zero-curvature section after the circular curve.

Another approach to critical point detection, we will call Gaussian smoothing [Asada 86, Pridmore 87]. Instead of regulating the amount of data smoothing by choice of the line segment length fit to the data as above, the difference of curvatures is first measured on only a few points along the curve (typically three to five); then these local curvature measures are plotted on a curvature plot and a Gaussian-shaped filter is applied to smooth noise and retain features. Critical points are found from this curvature plot by thresholding as before. The tradeoff between noise reduction and feature resolution is made by the user's choice of the width of the Gaussian smoothing filter. Similarly to the length of the fit for the *k*-curvatures method above, the Gaussian filter width is chosen as wide as possible to smooth out noise, but not so wide as to merge adjacent features. The user must empirically determine a width that yields a good tradeoff between noise reduction and feature maintenance.

An advantage of both the DOS and Gaussian smoothing families of methods over many other methods is that their results are symmetric with respect to curve features. That is, a feature will be located at the same point along the curve independently of the direction of curve traversal. However a drawback of these methods is that the user must have an idea of the noise and feature characteristics to set the length of the line fit for the DOS approach or the width of the Gaussian smoothing filter for that method. This is called the region of support because it relates to the arclength on the curve that affects the curvature measurement at any single point. The tradeoff on this region of support is the following. The support should be as large as possible to tend to average out noise, but it should not be so large as to also reduce true features. This requires the user to specify what constitutes a true feature versus noise, for instance the minimum corner angle or curve length. A problem can be encountered here if the noise or feature characteristics change for different images in an application, because this requires the user to adjust the method parameters for different images. Worse yet is if the feature characteristics have a large range in a single image, for instance if there is a very large, low-curvature circular curve and a very tight one. In this case, it is very difficult to accurately determine all critical points by the methods described.

3.3.2: Adaptive and Multi-Scale Methods

For the methods above, the choice of the method parameters determines the minimum curvature feature that can be resolved, and small-curvature features can be lost if the parameter value is chosen too large, or noise points detected as features if the parameter value is too small. Both of these methods can be performed at multiple scales (see below), but another approach is to adaptively estimate curvature when features of different size and separation are present along the curve. The adaptive approach requires no user parameter of expected region of support; instead this is determined using local feature data [Phillips and Rosenfeld 1987, Teh and Chin 1989]. In [Teh and Chin 1989], the region of support is adaptively determined by extending segments from a data point to points along the curve at both sides of each data point. The segments are extended from point to point, and the maximum perpendicular distance from the segment to the data is measured (similarly to the measure of error for Ramer’s polygonalization method in Section 3.1) for each extended segment. This extension is halted when a function of the error from segment to curve and the length of the segment indicates that the region of support has reached the next feature; and this final extent on the curve is the region of support. Because this region of support adapts to the data, it is as large as possible to reduce noise, but not too large as to reduce true signals. Critical points are detected by thresholding high values as above. This is an advantage over the *k*-curvatures and Gaussian approaches where the line segment length or Gaussian smoothing width must be chosen with some knowledge of the minimum region of support of a feature. However one drawback is that the output generated by this algorithm is sensitive to the direction of travel along the curve, and this results in critical points not necessarily being located in the center symmetrical features. This is in contrast to the *k*-curvatures and Gaussian smoothing methods, as mentioned above. Figure 2 shows an example of the results of this adaptive method, and compares it to other methods, including the *k*-curvature method.

Another way to detect different size features in the presence of noise is to effectively process with many regions of support, that is at multiple scales. This approach detects features across the range of scales examined. At a large scale, the region of support will extend a long length along the data curve, thus noise will be averaged out and large features detected; however, small features may be lost. At a small scale, the arc-length of support will be small, thus small features will be found, but noise may be erroneously detected as features. A uniform sequence of scales is used between the largest and smallest to determine features that will be detected best at specific scales throughout this range. One approach to multiple scales is an extension of the Gaussian smoothing method mentioned above [Asada 86, Mokhtarian 86]. The method employs Gaussian smoothing of the curvature plot for a range of sizes of Gaussian filter widths to cover the range of scales desired. Another approach is an extension of the *k*-curvature method [Deguchi 88]. In this case, *k*, the length of support on the arc, is the scale parameter. See the curvature plot for multiple scales of support upon the corner shown in Figure 3. The peaks on the curvature plot indicate the corner feature, and these peaks are smaller and wider as the region of support is larger.

Instead of calculating all scales of a feature, an adaptive method can be used to find the most appropriate scale of filtering that reduces noise but keeps the desired features. In [Saint-Marc et al. 89] the approach is taken to first assign continuity values to points on the curve, such that smooth sections have high continuity values. Then, small filter masks are applied iteratively to the data, weighted by these coefficients such that discontinuities are retained and noise within otherwise smooth portions is reduced. An example is shown in Figure 4. The curve is shown with its curvature plot before and after adaptive critical point detection, and the resulting critical points are also shown. In the first curvature plot without adaptive smoothing, there are peaks that correspond to corner and curve features, but there are also smaller peaks due to noise. The curvature plot is shown after 75 iterations of adaptive smoothing. Note that the peaks corresponding to features are retained, but those due to noise are reduced, These features can be more easily located on this plot and their correspondence can be made to features in the original curve.

The paper [Wu 1993] is a good reference on critical point detection. The approach in this paper begins with polygonal approximation, then refines the corners to better coincide with true corners. This paper also uses k-curvature in the preprocessing stage, so it provides a good description of this method and its use as well.

Other Figure Captions

Figure 2. Critical point detection for a number of different methods. (from [Teh and Chin, 1989]).

Figure 3. Multi-scale *k*-curvature method. On the left is a curve containing a corner. On the right are curvature plots for multiple scales of *k*-curvatures (i.e. multiple lengths of *k*). As k increases, the widths of the peaks increase, and the peaks decrease in height. (from [Deguchi, 1988]).

Figure 4. The data curve is shown in the top left, with a number of corner and curve features. The curvature plot shown in the top right is calculated from a small region of support and much noise is evident because of this. The adaptively smoothed curvature plot is shown in the lower left, where discontinuities are retained and noise reduced. The final diagram shows the locations of critical points from this curve, where the Xs indicate the bounds of circular curves, and the circles indicate corners.

REFERENCES

1. F. Attneuve, “Some informational aspects of visual perception”, Psychological Review, Vol. 61:183-193, 1954.
2. A. Rosenfeld, E. Johnston, “Angle detection on digital curves”, IEEE Trans. Computers, 22: 875-878, Sept. 1973.
3. A. Rosenfeld, J.S. Weszka, “An improved method of angle detection on digital curves”, IEEE Trans. Computer, Vol. C-24, 1975, pp. 940-941.
4. H. Freeman, L. Davis, “A corner-finding algorithm for chain-coded curves”, IEEE Trans. Computer, Vol. C-26, 1977, pp. 297-303.
5. L. O'Gorman, “Curvilinear feature detection from curvature estimation”, 9th Int. Conference on Pattern Recognition, Rome, Italy Nov., 1988, pp. 1116-1119.
6. H. Asada, M. Brady, “The curvature primal sketch”, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. PAMI-8, No. 1, Jan., 1986, pp. 2-14.
7. A.P. Pridmore, J. Porrill, J.E.W. Mayhew, “Segmentation and description of binocularly viewed contours”, Image and Vision Computing, Vol. 5, No. 2, May 1987, pp. 132-138.
8. T.Y. Phillips, A. Rosenfeld, “A method of curve partitioning using arc-chord distance”, Pattern Recognition Letters, 7: 201-206, 1972.
9. C.H Teh, R.T. Chin, “On the detection of dominant points on digital curves”, IEEE Trans. Pattern Analysis and Machine Intelligence, 11(8):859-872, 1989.
10. F. Mokhtarian, A. Mackworth, “Scale-based description and recognition of planar curves and two-dimensional shapes”, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(1):34-43, 1986.
11. K. Deguchi, “Multi-scale curvatures for contour feature extraction”, Proc. 9th Int. Conf. Pattern Recognition, Rome, Nov. 1988, pp. 1113-1115.
12. P. Saint-Marc, J.S. Chen, G. Medioni, “Adaptive smoothing: A general tool for early vision”, IEEE Trans. PAMI, Vol. 13, No. 6, June, 1991, pp. 514-529.
13. W-Y. Wu, M-J. J. Wang, “Detecting the dominant points by the curvature-based polygonal

approximation", CVGIP: Graphical Models and Image Processing, Vol. 55, No. 2, March, 1993, pp. 79-88.

3.4: Line and Curve Fitting

Keywords: straight line fit, corner detection, polynomial fit, spline fit, B-splines, Hough transform

Lines and curves, or portions of lines and curves, can be represented by mathematical parameters of spatial approximations, or fits, made to them. These parameters can be used in turn to identify features in the objects for subsequent recognition or matching. This representation by parameters instead of by the original pixels is useful because the parameters can provide a useful description of the objects as, for example: an object of four straight lines joined at 90^o corners (a rectangle), or one with a specific type of curve that matches a curve of a particular object of interest. To aid the fitting process, we can use information from analyses already described to guide the fit, such as the polygonal approximations (section 3.1) or locations of critical points (Section 3.2). It is also helpful to have some *a priori* idea of the type of fit and the limited shapes of objects in an application to reduce computation and ensure better fits.

3.4.1: Straight Line and Circular Curve Fitting

A simple way to fit a straight line to a curve is to just to specify the endpoints of the curve as the straight line endpoints. (See Figure 1.) However, this may result in some portions of the curve having large error with respect to the fit. A popular way to achieve the lowest average error for all points on the curve is to perform a least-squares fit of a line to the points on the curve. For a line fit of equation, $y = mx + b$, the objective is to determine m and b from the data points, (x_i, y_i). The least square fit is made by minimizing the sum of errors, $\Sigma(y - y_i)^2$ over all data points, $i = 1, \ldots n$. The solution is found by solving the simultaneous equations, $\Sigma y_i = m\Sigma x_i + bn$, and $\Sigma x_i y_i = m\Sigma x_i^2 + b\Sigma x_i$, for all data points, i, to obtain m and b.

There can be a problem in blindly applying the least-squares method just described to document analysis applications. That is, the method is not independent of orientation, and as the true slope of the line approaches vertical, this method of minimizing y-error becomes inappropriate. One approach is to minimize y-error only for slopes expected to be ± 45 degrees around the horizontal and to minimize x-error for slopes expected to be ± 45 degrees around the vertical. A more general approach is to minimize error not with respect to x or y, but with respect to the perpendicular distance to the line in any orientation. This can be done by a line-fitting method called principal axis

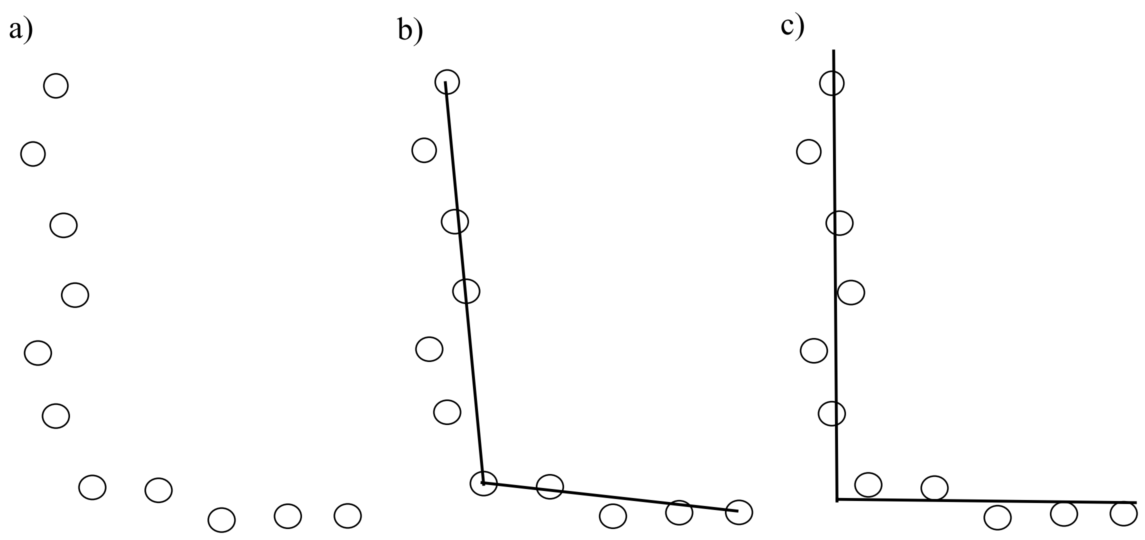


Figure 1. ON-valued pixels are shown in (a) and two corner fits are shown in (b) and (c). In (b), the fit is made exactly on data points. Because of this, the endpoints are sure to connect, but the corner angle is not as sharp as intended in the original. In (c), a least squares fit was performed to two sets of points, and the intersection of them was taken as the corner location. This yields a sharper corner.

determination or eigenvector line fitting. This is further detailed in [Duda and Hart 1973, Ballard and Brown 1982].

A second note with respect to line fitting methods is that they do not guarantee that endpoints fall on data points. For piecewise linear data each line fit will have to be adjusted to connect the lines. This is usually done by extrapolating to the intersection of the adjacent line fit.

Especially for machine-drawn documents such as engineering drawings, circular curve features are prevalent. These features are described by the radius of the curve, the center location of the curve, and the two transition locations where the curve either ends or smoothly makes the transition to one or two straight lines around it [O'Gorman 88, Rosin and West 89]. (See Figure 2) One sequence of procedures in circular curve detection begins by finding the transition points of the curve, that is the locations along the line where a straight line makes a transition into the curve, and then becomes a straight line again. Next, a decision is made on whether the feature between the straight lines is actually a corner or a curve. This can be done on the basis of an arclength threshold, that is if the arclength between transition points is longer than this threshold, then the feature is a curve; otherwise it is a corner. Finally, the center of the curve can be determined with the information that, since each transition point is defined to be at a tangent to the curve, two perpendicular lines through these transition points will intersect at the curve center.

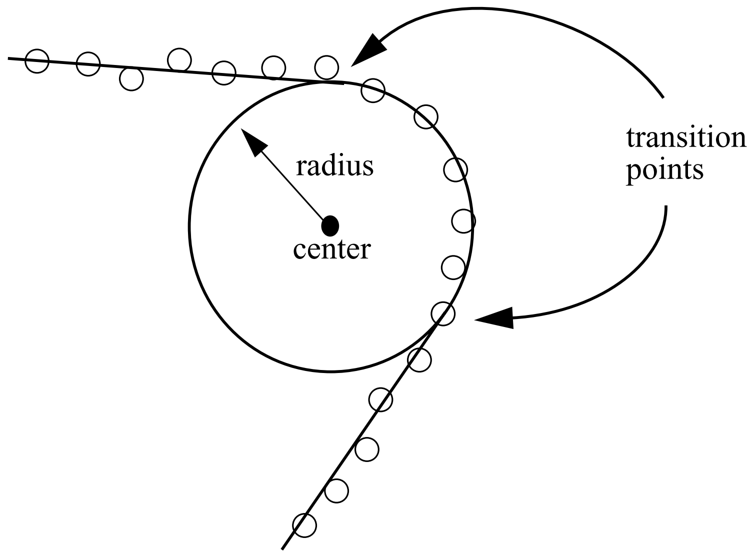


Figure 2. Circular curve fit to data points. Diagram shows parameters of this fit: two transition points from straight lines into the circular curve, and the radius and center location of the circle describing the curve.

3.4.2: Other Line Fitting Methods Including Splines and the Hough Transform

If a higher order fit is desired, splines can be used to perform piecewise polynomial interpolations among data points [Pavlidis 82, Medioni and Yasumoto 87]. B-splines are piecewise polynomial curves that are specified by a guiding polygon — such as is generated from polygonalization. Given two endpoints and a curve represented by its polygonal fit, a B-spline can be determined that performs a close but smooth fit to the polygonal approximation (Figure 3). The B-spline has several properties that make it a popular spline choice. One is that, besides a smooth interpolation between endpoints, it is also smoothly continuous (up to the second derivative) at the endpoints. Another property is that the B-spline is guaranteed to vary less than the guiding polygon of the sample points. This prevents problems that can occur with other types of fits, especially higher order fits, of wildly erroneous mis-fits at singularities. Though spline fits are useful to produce smoothly pleasing curves, it should be noted that first and second order features (straight lines are circular curves) are the most common features used for matching in document analysis. This is because it is difficult to obtain a higher order fit with matching parameters for two curves due to noise and discretization error. It is also because first and second order features are most common in document applications.

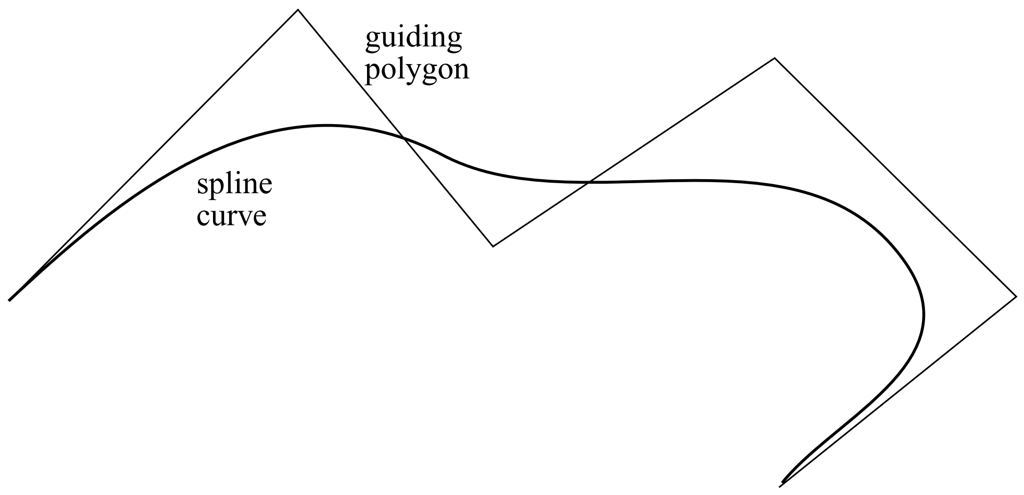


Figure 3. A spline curve is shown with its guiding polygon. The guiding polygon is from a straight-line polygonal fit to the image data. Notice that the spline lies within the corners of the polygon, thus yielding a smooth fit to the data.

A different approach for line and curve fitting is by the Hough transform [Illingworth and Kittler 88]. This approach is useful when the objective is to find lines or curves that fit groups of individual points on the image plane. The method involves a transformation from the image coordinate plane to parameter space. Consider the case where lines are the objects of interest. The equation of a line can be expressed as $r = x\cos\theta + y\sin\theta$, so there are two line parameters, the distance,r, and the angle, θ that define the transform space. Each *(x,y)* location of an ON pixel in the image plane is mapped to the locations in the transform plane for all possible straight lines through the point, that is for all possible values of r, and θ. When multiple points are colinear, their transformations will intersect at the same point on the transform plane. Therefore, the (r, θ) locations having the greatest accumulation of mapped points indicate lines with those parameters. (See example in Figure 4.). In practice, due to discretization error and noise, points will not be exactly colinear, and thus will not map to exactly the same location on the transform plane. Therefore, after transformation, the detection step of the Hough method requires two-dimensional peak detection. The method can be extended to other parametric curves. For instance, for a circle of equation, $(x-a)^2+(y-b)^2 = r$, the parameter space is three-dimensional with axes *a*, *b*, and *r*. For the circle and higher order curves, there is an increased memory and computational expense as the dimensionality increases.

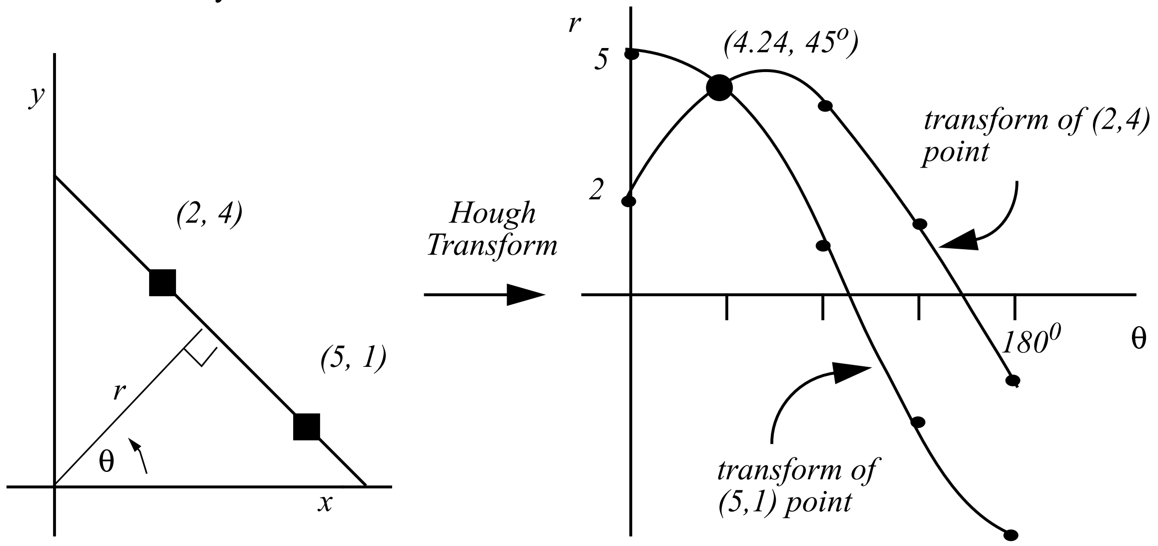


Figure 4. On the left is the image plane with two points through which a line can be drawn. On the right is the Hough parameter space for a line. Each point transforms to a sinusoid in Hough space. The point of intersection between the two sinusoids indicates the parameters of the line passing through both points. Note that, in practice the Hough plane is discrete; transform points (not curves) are plotted and accumulated, and the peak indicates the colinear point(s) in the image.

While the Hough transform is useful for fitting unconnected points, it is costly in terms of computation. In document analysis, we usually have more information than simply isolated points, and this can be used to reduce computation. In particular, points are usually connected in continuous lines (though there may be small gaps as in dashed lines or due to noise). For connected lines or portions of lines, computation can be reduced greatly by mapping not to all (r, θ) possibilities, but just to the one orientation (one point) indicated by the angle information of the line. Besides the computational constraints, the Hough transform is limited in that it does not yield the coordinates of the line endpoints and long lines are favored over shorter lines (in the peak-picking stage). For these reasons, there is often a spatial domain method that is faster and more effective in solving the problem. For instance, for dashed line detection in an engineering drawing, it is important to locate the line ends and the lengths of the dashes (when there are more than one type of dashed line). While the Hough transform could be used to perform a portion of this problem, a dashed line detection technique designed specifically for this can be used to solve the whole problem [Kasturi et al. 90, Lai and Kasturi 91]. However, the Hough transform is a general and widely useful method, and it will be described in Chapter 4 how it is used to determine text lines.

Two papers are suggested for advanced reading. One concisely describes straight line and circular curve fitting to true line features [O'Gorman 88]. The paper describes corner detection and B-spline fitting [Medioni 87].

References

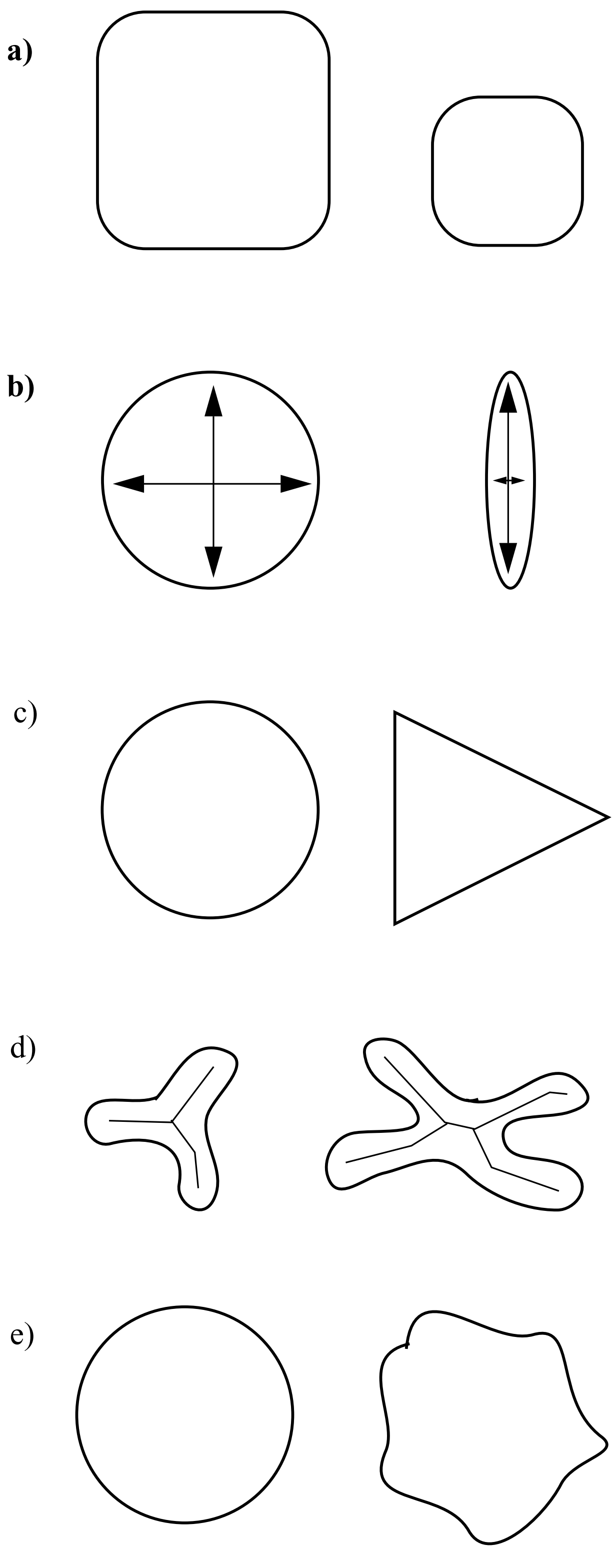
1. R. O. Duda, P. E. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York, 1973, pp. 332-335.
2. D. H. Ballard, C. M. Brown, *Computer Vision*, Prentice-Hall, New Jersey, 1982, pp. 485-489.
3. L. O'Gorman, ``Curvilinear feature detection from curvature estimation", 9th Int. Conference on Pattern Recognition, Rome, Italy Nov., 1988, pp. 1116-1119.
4. P.L. Rosin, G.A.W. West, ``Segmentation of edges into lines and arcs", Image and Vision Computing, Vol. 7, No. 2, May 1989, pp. 109-114.
5. Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, Maryland, 1982
6. G. Medioni, Y. Yasumoto, ``Corner detection and curve representation using cubic B-splines", Computer Vision, Graphics, and Image Processing, Vol. 29, 1987, pp. 267-278.
7. J. Illingworth, J. Kittler, ``A survey of the Hough transform", Computer Graphics and Image Processing, 44:87-116, 1988.
8. R. Kasturi, S. T. Bow, W. El-Masri, J. Shah, J. Gattiker, U. Mokate, "A system for interpretation of line drawings", IEEE Trans. PAMI, Vol. 12, No. 10, Oct. 1990, pp. 978-992.
9. C.P. Lai, R. Kasturi, ``Detection of dashed lines in engineering drawings and maps", Proc. first Int. Conf. on Document Analysis and Recognition, St. Malo, France, 1991, pp. 507-515.

3.5: Shape Description and Recognition

Keywords: shape description, topology, shape metrics, convex hull, moments, projections

When matching one object to another, the objective is to have a precise and concise description of the object so that incorrect matches can be quickly rejected and the proper matches can be made without ambiguity. The nature of the description will depend on the application, in particular the shapes of the objects and the number of objects to be matched against. For instance if the objective is to recognize text and graphics, where all the graphics consist of long lines, then discriminating descriptors would be region size or contour length. To recognize the difference between squares, triangles, and circles, the corner and curve detection methods of the previous section would apply. In this section, a sampling of shape descriptors are described, especially those that may apply to document analysis. The purpose here is not to give a comprehensive listing of descriptors, because there can be many and can be devised by the user for shapes of a particular application. Instead, many examples are shown to convey the approach in choosing shape descriptors. The approach is that they be appropriate to describe essential, inherent features of the shapes involved, and that they act as good discriminators to differ between these shapes. Figure 1 shows a number of shapes and some appropriate shape metrics that are discussed in this section..

One of the simplest ways to describe shapes is by shape metrics. For instance, the area measurements (number of ON-valued pixels) of connected components of a document can be used to separate large font size characters from smaller characters, or text from larger line graphics. Instead of counting up all the pixels in a connected component, a faster determination is the length of the contour, especially when the shape has already been stored as the chain code of its contour. This length measurement can be used to separate small regions from large ones. Of course, contour length and region area are not directly related — they also depend on other aspects of the shape. If the fatness versus thinness of a shape is an important discriminator, then compactness can be measured; that is the ratio of the square of the contour length over the area. This is small for a circular contour, and large for a long shape or a shape with many concavities and convexities. Another way to describe a shape is by its moments. The first moment of a region is the average of *(x, y)* locations for all pixels in the region. These x and y moments are, $m_x^{(1)} = (1/n)\Sigma x_i$ and $m_y^{(1)} = (1/n)\Sigma y_i$, where $i = 1, \ldots n$ is the number of ON pixels. This first moment describes



Size — number of ON pixels in region, length of contour, or area of bounding box

Compactness — ratio of area to square of contour length, ratio of major to minor axes, or second central moment

Asymmetry — third central moment The circle will have a value of zero indicating perfect symmetry, and the triangle will have a non-zero value.

Topology — number of branches in thinned versions of shapes (see also syntactic description below)

Contour Smoothness — Fourier descriptor Both shapes will have similar low harmonics, indicating similar shapes. The right shape will have power in the high harmonics indicating an unsmooth contour.

Figure 1. Shapes and suggested discriminating shape measures.

the average location of an object. The second moment is the average of the squares of x and y locations, etc. The central moments normalize the measurement with respect to the first moment to make it independent of location. The second central moments in x and y are: $m_x^{(2)} = (1/n)\Sigma(x_i - m_x^{(1)})^2$ and $m_y^{(2)} = (1/n)\Sigma(y_i - m_y^{(1)})^2$. These can be used to indicate the non-roundness, eccentricity, or elongation of the shape; for instance, if the second moments in x and y are similar in value, then the object is more round than otherwise. The third central moment gives a measure of the lack of symmetry of this eccentricity. Besides these central moments, moments of different orders can also be combined into functions that are invariant to other geometric transformations such as rotation and scaling; these are called moment invariants. (See reference [Hu 77] for details on moment invariants.) Moments are less useful to describe more complicated shapes with many concavities and holes; in these cases, specific features such as number of holes or thinned topology are more applicable.

Shapes can be described by topological features, that is the number of holes and branches. For instance, the letter, “B", has two holes; and the letter, “P", has one hole and one branch. For these two examples, connected-component labeling can be used to determine the holes. However, for this example, thinning is probably more appropriate for finding both the holes (these are loops on the thinned characters) and branches. If a shape has many convexities, its skeleton will have many branches. For example, the thinned results of the symbols, “*” and “X” will have six and four branches respectively. The number of branches, the number of loops, the number of endlines, and the directions, lengths, etc. of each are all descriptive of the original shape. Note should be made with regards to the use of thinning, that the thinned results will rarely match pixel-for-pixel against the same shape. This is due to digitization noise and noise in the boundary of the shape. This is why shapes are better matched on the basis of some description based on features such as number of branches, holes, etc.

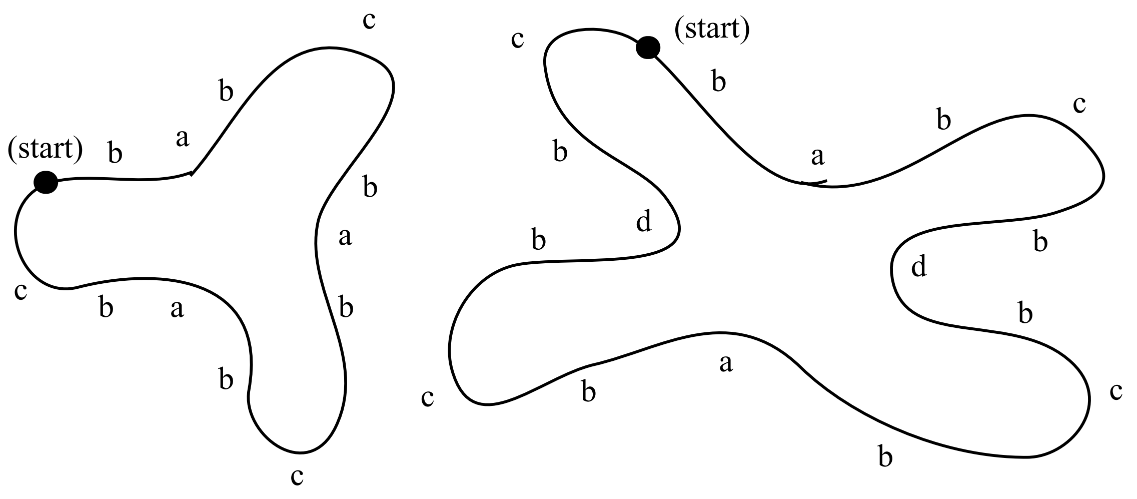
The contour of a shape can be used for shape description. A global description of the contour can be determined by calculating the Fourier transform of the contour along its arc length. This gives a frequency domain description of the boundary in terms of Fourier coefficients, called Fourier descriptors. Lower order terms contain information of the large scale shape, that is its low-frequency curvature characteristics; and higher order terms contain information on sharper features, that is high frequency curvature characteristics. Therefore shapes can be recognized and matched on the basis of having, for instance, lower high-order coefficient values for a smooth shape such as an “O", versus larger high-order coefficient values for a shape with sharper corners such as a “W". A drawback of using Fourier descriptors is that they provide only global information, therefore if two shapes have the same frequency information, for example, a “P" and a “d", then they cannot be discriminated using Fourier descriptors alone. Figure 2 shows letters and their Fourier approximations. For further description of Fourier descriptors, see, for instance, [Jain 89].

Non-global features along contours can also be used for shape description. Critical points, or the lines, corners, and curves, between them can be determined in the contour as described in Section 3.2, then they can be matched against these features on other shapes. For complicated shapes, it is common to code a sequence of features found around a contour into a syntactic description that describes a “legal" sequence as may be found for all shapes of interest. Examples of legal sequences are: (line, corner, line, corner, line, corner, line, corner) for a rectangle, or (curve) for a circular object, etc. Then, syntactic pattern recognition techniques that locate these legal sequences can be used to assist in matching. In Figure 3, two contours are shown with their syntactic descriptions.

A final example of a shape descriptor is the projection profile, or signature, of a shape. This is the accumulation of the number of ON pixels along all rows or columns into a one-dimensional histogram whose shape is descriptive of the original two-dimensional shape. To use the example of the letters, “P" and “d", again, a horizontal projection profile (accumulation along all rows through the letters) will indicate more ON pixels above and fewer below for “P" than “d", and vice versa. This shape measure is very dependent upon the orientation of the shape with respect to the axis of accumulation. In Chapter 4, it will be described how the projection profile can be used to describe the “shape" (orientation and text lines) of a document page.

In commercial systems, shape descrriptors are used in most aspects of document analysis. For OCR, systems use shape descriptors (in part) to distinguish characters. Lines are distinguished from characters by their shapes, the former to be analyzed by graphics analysis tools, the latter by OCR. In graphics analysis, long, thin objects are distinguished from rounder objects for appropriate processing: thinning-based processing for the former and region-based for the latter.

For more comprehensive reviews of shape description techniques, see [Pavlidis 1978, Marshall 1989]. For an extensive collection of papers on shape analysis, see [Arcelli 1992].



Y-shape: babcbabcbabc X-Shape: babcbdbcbabcbdbc

Figure 3. Y and X shapes with coded contours. The codes describe the type of curve segments by their shapes and amounts of curvature: a— low curvature, concave curve, b —low curvature curve or line, c — high curvature, convex curve, d — high curvature, concave curve. The two syntactic descriptions are the sequences of codes around the contour. These are different, indicating different shapes.

Other Figure Captions

Figure 2. Reconstructions of shapes from Fourier descriptors: a) original, b) using only five harmonics, c) using ten harmonics, d) using fifteen harmonics. As more of the complete Fourier description is used, the shapes appear more like the originals [Brill 68].

REFERENCES

1. M. K. Hu, "Visual pattern recognition by moment invariants", in *Computer Methods in Image Analysis*, J. K. Aggarwal, R. O. Duda, A. Rosenfeld (eds.), IEEE Computer Society, Los Angeles, 1977.
2. A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, New Jersey, 1989, pp.

377-381.

3. E. L. Brill, “Character recognition via Fourier descriptors”, WESCON, Paper 25/3, 1968.
4. T. Pavlidis, “A review of algorithms for shape analysis", Computer Graphics and image processing", vol. 7, 1978, pp. 243-258.
5. S. Marshall, “Review of shape coding techniques", Image and Vision Computing, Vol. 7, No. 4, Nov. 1989, pp. 281-294.
6. C. Arcelli, L. P. Cordella, G. Sanniti di Baja (eds.), *Visual Form: Analysis and Recognition*, (Proceedings of the International Workshop on Visual Form, May, 1991, Capri, Italy), Plenum Press, New York, 1992.

Chapter 4

Recognizing Components of Text Documents

4.1: Introduction

There are two main types of analysis that are applied to text in documents. One is optical character recognition (OCR) to derive the meaning of the characters and words from their bit-mapped images, and the other is page layout analysis to discover formatting of the text, and from that to derive meaning associated with the positional and functional blocks in which the text is located. These may be performed separately, or the results from one analysis may be used to aid or correct the other. OCR methods are usually distinguished as being applicable for either machine-printed or handwritten character recognition. Layout analysis techniques are applied to formatted, machine-printed pages, and a type of layout analysis, forms recognition, is applied to machine-printed or handwritten text occurring within delineated blocks on a printed form. Both OCR and format analyses are covered in this chapter, and the last section contains descriptions of document systems incorporating both.

We recommend two papers for advanced reading on a topic that straddles the bounds between text analysis of this chapter and graphics analysis of Chapter 5. One is a paper on separation of text from graphics components [Jain 1992], where text is the desired result. The other is a paper also on separation, but is from the point of view of yielding graphics elements [Fletcher and Kasturi 1988].

Another recommended paper describes a document database [Phillips et al. 1993] that is intended for researchers to test against a common and known set of data. The University of Washington document database contains images of machine-printed documents, with and without noise, and each with ground truth information that describes the contents, including the text and its location. This is a readily obtainable (by CD-ROM) source of documents for testing and comparing document processing techniques.

References

1. A. K. Jain and S. Bhattacharjee, "Text segmentation using Gabor filters for automatic docu-

ment processing,”, Machine Vision and Applications, Vol. 5, pp. 169-184, 1992.

2. Fletcher, Kasturi, “A robust algorithm for text string separation from mixed text/graphics images”, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 10, No. 6, Nov. 1988, pp. 910-919.
3. I.T. Phillips, S. Chen, and R.M. Haralick, “CD-ROM Document Database Standard”, Int. Conf. on Document Analysis and Recognition, Tsukuba Science City, Japan, October 1993, pp. 478-483.

4.2: Skew Estimation

Keywords: skew estimation, page orientation, projection profile, Hough transform, nearest-neighbor clustering, text lines

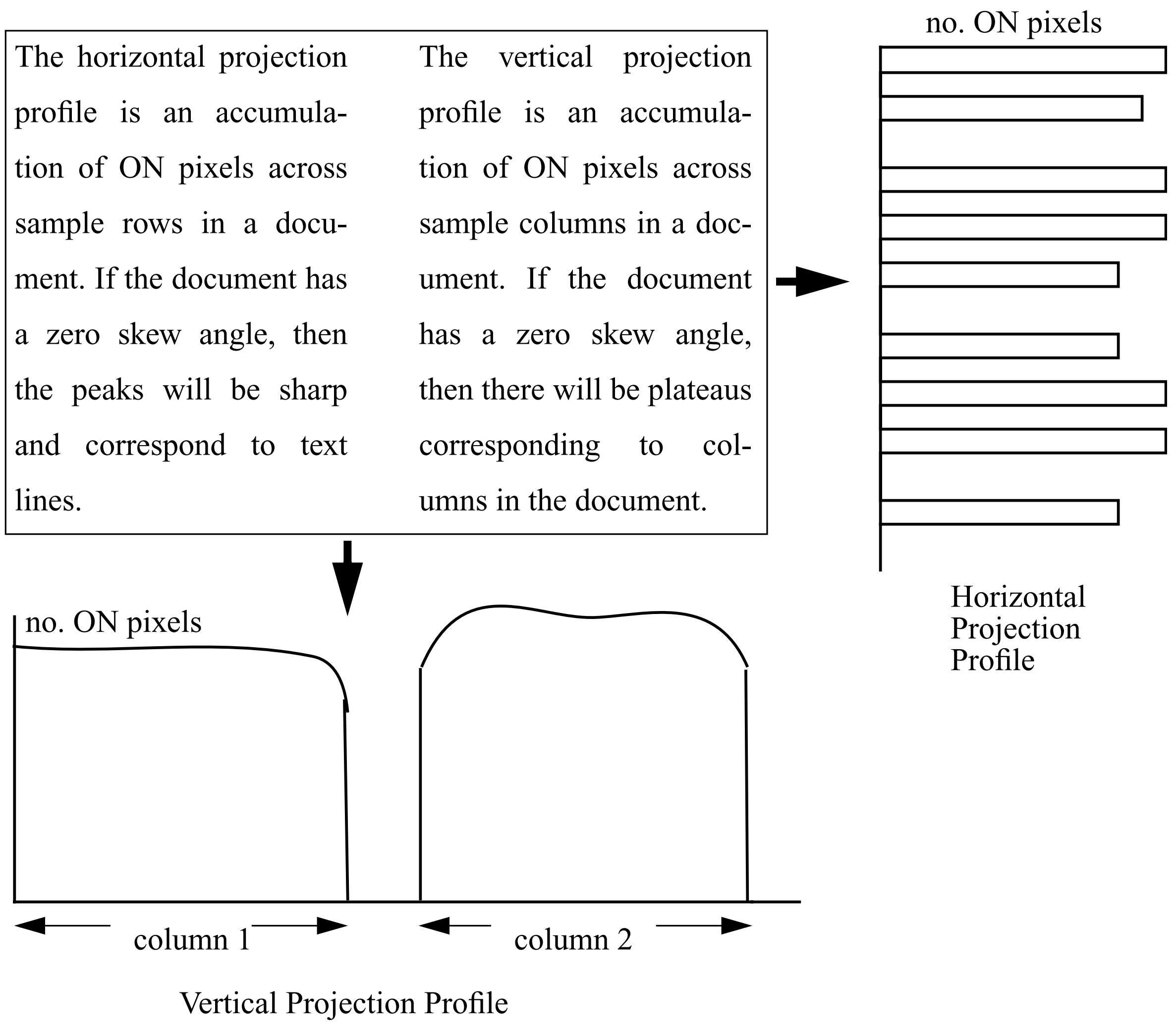
A text line is a group of characters, symbols, and words that are adjacent, relatively close to each other, and through which a straight line can be drawn (usually with horizontal or vertical orientation). The dominant orientation of the text lines in a document page determines the skew angle of that page. A document originally has zero skew, where horizontally or vertically printed text lines are parallel to the respective edges of the paper, however when a page is manually scanned or photocopied, non-zero skew may be introduced. Since such analysis steps as OCR and page layout analysis most often depend on an input page with zero skew, it is important to perform skew estimation and correction before these steps. Also, since a reader expects a page displayed on a computer screen to be upright in normal reading orientation, skew correction is normally done before displaying scanned pages.

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.

4.2.1: Projection Profile Methods

A popular method for skew detection employs the projection profile. A projection profile is a histogram of the number of ON pixel values accumulated along parallel sample lines taken through the document. (See Figure 1.) The profile may be at any angle, but often it is taken horizontally along rows or vertically along columns, and these are called the horizontal and vertical projection profiles respectively. For a document whose text lines span horizontally, the horizontal projection profile will have peaks whose widths are equal to the character height and valleys whose widths are equal to the between-line spacing. For multi-column documents, the vertical projection profile will have a plateau for each column, separated by valleys for the between-column and margin spacing.

The most straightforward use of the projection profile for skew detection is to compute it at a number of angles close to the expected orientation [Postl 1986]. For each angle, a measure is made of the variation in the bin heights along the profile, and the one with the maximum variation



The horizontal projection profile is an accumulation of ON pixels across sample rows in a document. If the document has a zero skew angle, then the peaks will be sharp and correspond to text lines.

The vertical projection profile is an accumulation of ON pixels across sample columns in a document. If the document has a zero skew angle, then there will be plateaus corresponding to columns in the document.

Figure 1. A portion of two columns of text are shown at the top left. The horizontal and vertical projection profiles from this text are also shown. Notice in the horizontal projection profile that each bin corresponds to a text line, and in the vertical projection profile, each plateau corresponds to a column.

gives the skew angle. At the correct skew angle, since scan lines are aligned to text lines, the projection profile has maximum height peaks for text and valleys for between-line spacing. Modifications and improvements can be made to this general technique to more quickly iterate to the correct skew angle and to more accurately determine it.

One modification of the projection profile method above is proposed by Baird [Baird 1987] to improve speed and accuracy of skew detection. Connected components are first found, and represented by points at the bottom-center of their bounding boxes. A measure of the total variation (such as the peak to valley difference in height) is determined for various projection angles. The resulting value for each angle is a measure of colinearity of the baseline points along each angle — the higher the variation, the closer is the skew to being zero. The peak of these measures gives the true skew angle. The accuracy of this method is typically ± 0.5 degrees with respect to the correct orientation. Because detection is done using the "bottom-center" point of each component, there is an assumption that the page is roughly upright when scanned, and, partly due to this assumption, the method has its highest accuracy within ± 10 degrees of skew.

A faster, though less accurate, method to approximate the skew angle measures shifts in projection profiles [Akiyama 1990]. For text lines that are approximately horizontal, the document is divided into equal width vertical strips that each cover the height of the page image. Horizontal projection profiles are calculated for each vertical strip, and the skew is the angle of the shift between neighboring strips where the peaks correspond. The chosen width of the vertical strips is related to the skew — it must be small enough such that sharp peaks are obtained in the profile. For larger skew, the width must be smaller, but there is a lower limit when not enough of the text line is accumulated to obtain good peaks. The authors' experiments show this technique to work well if skew is less than ± 10 degrees.

Another fast method based on the measurement of shifts between strips is done using vertical projection profiles determined for horizontal strips of the page [Pavlidis and Zhou 1991] (in contrast to horizontal profiles on vertical strips as for [Akiyama 1990] described above). The locations of edges of columns for each strip are found from the locations of minima on each profile. Lines fit along these edge locations for all intervals are perpendicular to the skew. This method is faster than most others because iterations are not required and fewer points need to be fit; however, for the same reasons, it is not as accurate. Also, the accuracy is a function of the initially unknown skew angle and the chosen interval length. The method is said to work up to a skew angle of ± 25 degrees.

4.2.2: Hough Transform Methods

In Section 3.3, we mentioned that the Hough transform was useful for straight line detection. It can be used for similar purpose to find skew from text line components. [Srihari 1989]. As described in greater detail in Section 3.3, the Hough transform maps each point in the original *(x,y)* plane to all points in the (r, θ) Hough plane of possible lines through *(x,y)* with slope θ and distance from origin, *r.* We mentioned that performing the Hough transform on individual points was expensive, but there were speedups, such as using slope of line segments. For documents, one speedup is to compute “burst” images to reduce the number of pixel transformations to the Hough space. These horizontal and vertical bursts are a run of continuous ON pixels along rows or columns respectively. These are coded by their ON-length at the location of the end of the run. Thus, the burst image has high values at the right and bottom edges of characters (for a page with small skew angle), and the total number of pixels to transform is reduced. This burst image is then transformed to Hough space. Here, each burst value is accumulated in bins at all values of (r, θ) that parameterize straight lines running through its (x,y) location on the burst image. In Hough space, the peak θ bin gives the angle at which the largest number of straight lines can be fit through the original pixels, and this is the skew angle. One limitation is that when vertical and horizontal bursts are taken, an assumption has been made that the skew is limited in range — in this case the authors assume ± 15 degrees. Also, if text is sparse, as for some forms, it may be difficult to correctly choose a peak in Hough space. Even with using the bursts to improve efficiency, the Hough transform approach is typically slower than the non-iterative projection profile methods described above; however the accuracy is typically high.

4.2.3: Nearest-Neighbors Methods

All the above methods have some limitation in the maximum amount of skew that they could handle. One approach, using nearest-neighbor clustering, does not have this limitation [Hashizume 1986]. In this approach, connected components are first determined, the nearest neighbor of each component is found (that is, the component that is closest in Euclidean distance), and the angle between centroids of nearest neighbor components are calculated. (See Figure 2.) Since intra-character spacing is smaller within words and between characters of words on the same text lines, these nearest neighbors will be predominantly of adjacent characters on the same text lines. All the direction vectors for the nearest-neighbor connections are accumulated in a histogram, and the histogram peak indicates the dominant direction — that is, the skew angle. The cost of being able to obtain any skew angle is that this method has relatively higher computational cost than most, especially in comparison to the fast projection profile methods. The accuracy of the method depends on the number of components; however, since only one nearest-neighbor connection is made for each component, connections with noise, subparts of characters (dot on “i”), and between text lines can reduce the accuracy for relatively sparse pages.

An extension of this nearest-neighbor approach is to obtain not just one neighbor for each component, but k neighbors, where k is typically 4 or 5 [O'Gorman 1993]. As for the one-nearest-neighbor approach just described, the nearest-neighbor directions between centroids of connected components are first accumulated on a histogram. However, because there are $k \cong 5$ directions for each component, and connections may be made both within and across text lines, this histogram peak is used only as an initial estimate of the skew angle. This estimate is used to eliminate connections whose directions are outside of a range of directions around the estimate, that is, they are probably inter-line connections. Then, a least squares fit is made to the centroids of components that are still grouped by nearest-neighbor connections. These least-squared fits are assumed to be fits to full text lines, and this refined measurement is used as the more accurate estimate of skew angle. Because individual text lines are found here, this method encroaches on page layout analysis, in which text lines, columns, and blocks are found. Indeed, this method, called the docstrum, comprises both skew estimation and layout analysis, and will be further explained in the next section).

The skew detection methods used in commercial systems primarily employ the projection profile. Because of this, they are limited in the amount of skew they can handle, usually less than 10 degrees.

References

1. W. Postl, “Detection of linear oblique structures and skew scan in digitized documents”, Proceedings of the 8th International Conference on Pattern Recognition (ICPR), Paris, France, October 1986, pp. 687-689.

2. H.S. Baird, “The skew angle of printed documents”, Proceedings of the Conference of the Society of Photographic Scientists and Engineers, Rochester on Hybrid Imaging Systems, NY, May, 1987, pp. 14-21.

3. T. Akiyama, N. Hagita, “Automated entry system for printed documents”, Pattern Recognition, Vol. 23, No. 11, 1990, pp. 1141-1154.

4. T. Pavlidis, J. Zhou, “Page segmentation by white streams,”, Proceedings of the 1st Int. Con-

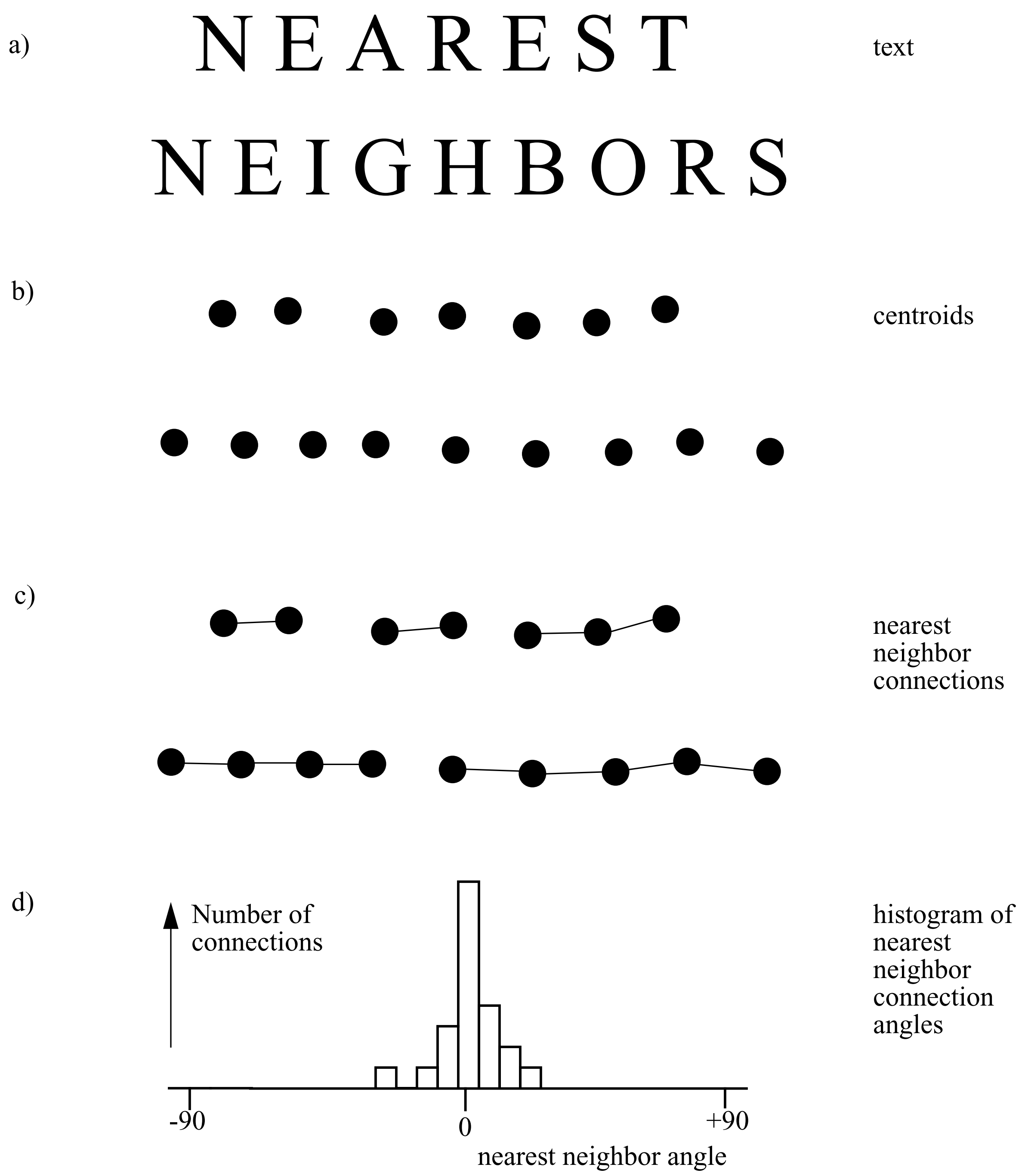


Figure 2. Diagram illustrates nearest-neighbor method. a) Text. b) Centroids of characters in (a). c) Connections between closest neighbors (1 neighbor each). d) Plot of connection angles showing peak at zero, indicating zero skew angle.

ference on Document Analysis and Recognition (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 and Applications (1989) 2:141-153.
6. A. Hashizume, P-S. Yeh, A. Rosenfeld, “A method of detecting the orientation of aligned components”, Pattern Recognition Letters, Val. 4, 1986, pop. 125-132.
7. L. O'Gorman, “The document spectrum for structural page layout analysis”, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, Nov. 1993, pp. 1162-1173.

4.3: Layout Analysis

Keywords: document layout analysis, structural layout analysis, geometric layout analysis, functional labeling, syntactic labeling, top-down analysis, bottom-up analysis, text lines, text blocks

After skew detection, the image is usually rotated to zero skew angle, then layout analysis is performed. Structural layout analysis (also called physical and geometric layout analysis in the literature) is performed to obtain a physical segmentation of groups of document components. Depending on the document format, segmentation can be performed to isolate words, text lines, and structural blocks (groups of text lines such as separated paragraphs or table of contents entries). Functional layout analysis (also called syntactic and logical layout analysis in the literature) uses domain-dependent information consisting of layout rules of a particular page to perform labeling of the structural blocks giving some indication of the function of the block. (This functional labeling may also entail splitting or merging of structural blocks.) An example of the result of functional labeling for the first page of a technical article would indicate the title, author block, abstract, keywords, paragraphs of the text body, etc. See Figure 1 for an example of the results of structural analysis and functional labeling on a document image.

Structural layout analysis can be performed in top-down or bottom-up fashion. When it is done top-down, a page is segmented from large components to smaller sub-components, for example, the page may be split into one or more column blocks of text, then each column split into paragraph blocks, then each paragraph split into text lines, etc. For the bottom-up approach, connected components are merged into characters, then words, then text lines, etc. Or, top-down and bottom-up analyses may be combined. We describe some specific methods below.

4.3.1: Top-Down Analysis

The horizontal and vertical projection profiles can be used for layout analysis. As already described in Section 4.2, text columns can be located on the vertical projection profile as plateaus separated by valleys. In a similar manner, the horizontal projection profile can be used to locate paragraph breaks within each column (when the paragraphs are separated by some extra blank space that is greater than the inter-line spacing). Within paragraphs, text lines can be found also from the horizontal projection profile. Some differentiation among text blocks can also be made

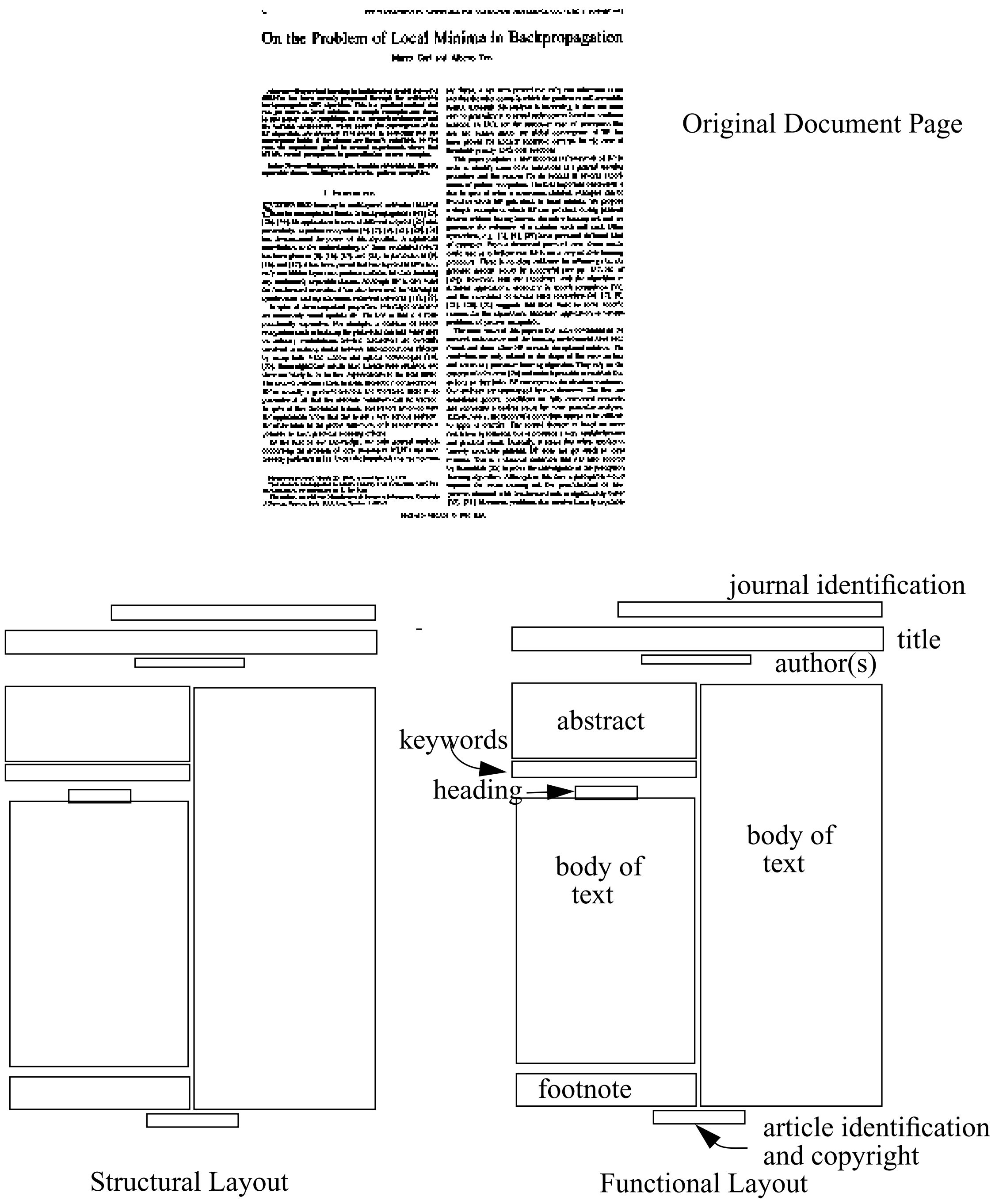


Figure 1. The original document page is shown with results from structural and functional layout analysis. The structural layout results show blocks that are segmented on the basis of spacing in the original. The labeling in the functional layout results is made with the knowledge of the formatting rules of the particular journal (IEEE Trans. Pattern Analysis and Machine Intelligence).

using this approach. For instance, often the horizontal projection profile of a title will indicate larger size text than for the body of text. Or a footnote may have smaller text line spacing than the body of text. In this top-down manner, much information can be determined relating to the layout of the page.

In practice, it isn't necessary to perform processing on all pixels of the original resolution image when using this projection profile approach to layout analysis. Instead the image can be reduced in size to improve efficiency and to improve results. One way to do this is to smooth characters into smaller, unrecognizable (as characters) blobs. Depending on the amount of smoothing, these blobs can be of a single character, a word, or text line. Projection profiles are constructed the same as before, but on the basis of fewer pixels. The run-length smoothing algorithm (RLSA) is a popular method for performing this smoothing [Wong 1982]. 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 inspecting white spaces between black pixels on the same lines and setting them to black if the lengths are less than a threshold. To merge within-word characters, the threshold is chosen greater than within-word character spacing and less than between-word spacing. Subsequent merging is accomplished similarly with appropriately sized thresholds. Besides reducing the number of pixels, this method can lead to better results, especially for sparse text that yields poor or noisy projection profiles otherwise. This use of horizontal and vertical projection profiles requires that the image is first skew-corrected and that spacing is known and uniform within the image.

A more structured top-down method that also uses projection profiles splits the document into successively smaller rectangular blocks by alternately making horizontal and vertical "cuts" along white space, starting with a full page, and continuing with each sub-block [Nagy 1984, Nagy 1992-a]. (See Figure 2.) The locations of these cuts are found from minima in the horizontal and vertical projection profiles of each block. Segmentation is aided by performing functional labeling at the same time, and this is based on a priori knowledge of features of the expected blocks, and the characteristics of their profiles, which are described in a document syntax. For instance, a title block may have large font titles that appear in the horizontal projection profile as wider peaks; whereas the horizontal profile of the text body will have narrower, periodic peaks and valleys. The results of segmentation are represented on an X-Y tree, where the top level node is for the page, each lower node is for a block, and each level alternately represents the results of horizontal (X-cut) and vertical (Y-cut) segmentation. Using this technique, segmentation can be performed down to individual paragraphs, text lines, words, characters, and character fragments. An advantage here is that the combination of structural processing and functional labeling enables the process to be directed and corrected. This requires of course the initial specification of block syntax that depend upon knowledge of block features (font size, line spacing, etc.). It should be noted also that it is critical that the image has no skew and that any salt-and-pepper noise is removed first. These two conditions are required by most segmentation methods.

Another top-down layout technique analyzes white space (i.e. background area versus foreground text) to isolate blocks and then uses projection profiles to find lines [Baird 1990]. (See Figure 3.) First, all locally-maximal white rectangles among the black connected components are enumerated from largest to smaller. The larger of these form a “covering” of the white background on the page, and thus a partition of the foreground into structural blocks of text. Rectangles may be chosen down to a chosen size such that segmentation is performed to a chosen level, for instance columns, paragraphs, or right down to individual characters. The page must have a Manhattan layout, that is, it must have only one skew angle, and be separable into blocks by vertical and horizontal cuts. One advantage of using background white space versus the foreground text for layout analysis is language-independence. This is the case since white space is used as a layout delimiter in similar ways in many languages. Another is that few parameters need be specified. A drawback is that the choice of what constitutes a “maximal” rectangle — longest, maximal area, etc. — may be non-intuitive, and different for differently formatted documents.

A primary advantage of top-down methods is that they use global page structure to their benefit to perform layout analysis quickly. For most page formats, this is a very effective approach. However, for pages where text does not have linear bounds, and where figures are intermixed in and around text, these methods may be inappropriate. For example, many magazines crop text around an inset figure so the text follows a curve of an object in the figure rather than a straight line. Table of contents pages of magazines and journals are also often formatted with inset figures, centering of column entries (rather than justified), and other non-Manhattan layouts. The bottom-up techniques described below are more appropriate for these formats, with the tradeoff that they are usually more expensive to compute.

4.3.2: Bottom-Up Analysis

We have already described that bottom-up layout analysis starts from small components and groups them into successively larger components until all blocks are found on the page. However, there is no single, general method that typifies all bottom-up techniques. In this section, we describe a number of approaches that can all be classified as bottom-up, but use very different intermediate methods to achieve the same purpose. This section also gives an idea of complete software systems for page layout analysis.

One approach combines a number of the techniques described above [Fisher 1990]. The skew is first found from the Hough transform as described in Section 4.1, then between-line spacing is found as the peak of the 1-dimensional Fourier transform of the projection profile for θ fixed at the computed skew angle. Run-length smoothing is performed, and within-line spacing is determined by finding the peak on a histogram of these lengths of white spaces (inter-character and inter-word spacing) and black lengths (words). Bottom-up merging of the text components is then done by a sequence of run-length smoothing operations in the directions of the skew for words and text lines, and perpendicular to the skew for paragraphs and text columns. The results are ON regions upon which connected component analysis is performed. Statistics are calculated on these connected components, for instance, ranges of word height, area, length, etc. This feature information is used to discern text blocks and to discriminate text and non-text. Esposito et al. [Esposito 1990] use a similar approach, except they first determine bounding boxes of individual characters, then operate with respect to these instead of on individual pixels to reduce computation.

The docstrum method [O'Gorman 1993] employs bottom-up *k*-nearest-neighbor clustering to group from characters into text-lines and structural blocks. (See Figure 4.) (The following describes the complete process of layout analysis by the docstrum, whereas we described only the first step, skew detection, in Section 4.1.) First, for each document component, *k* nearest-neighbor connections to neighboring components are found (where *k* is usually taken to be 4 or 5). The distances and angles of these connections are compiled in histograms. Because most connections will be made between characters on the same text lines, the peak angle will indicate the skew and the peak distance will indicate the inter-character spacing. Using these estimates, text lines are found as groups of characters and words that are along the page orientation. Text lines are then grouped into blocks using the document characteristic that text lines of the same block are usually spaced more closely than text lines of different blocks.

Similarly to the top-down method that used functional labelling information in the course of performing structural layout analysis [Nagy 1992-b], this combination of the two processes can also be used to advantage for bottom-up analysis. In [Akiyama 1990], segmentation is first performed using field separators (lines between text regions), then blank delimiters as for many other methods. Then, global and local text features are determined, and blocks are found and classified using generic properties of documents. The text features used here are measures on horizontal and vertical projection profiles, horizontal and vertical crossing counts (the number of black-white and white-black transitions along a line), and bounding boxes for each character. For example, headline text is identified by the property that the larger characters have smaller crossing counts. Individual characters are identified using their bounding box size relative to characters in the same region. This combination of structural layout segmentation and functional labeling has both advantages and disadvantages. Obviously, more is required from the user earlier in the processing stages to supply information on the expected features of labeled fields. Some researchers feel that it is advantageous to proceed as far as possible in early processing without the need for application-specific knowledge because this encourages robust techniques. However, others feel that, if the knowledge is available and useful, then it should be used to facilitate the problem. In the following section, functional labeling is discussed independently (as a following process) of structural layout analysis.

4.3.3: Functional Labeling

As described above, some of the layout analysis methods [Nagy 1992-b, Akiyama 1990] perform functional labeling in the course of structural blocking. Other methods perform these steps sequentially, first obtaining structural blocks, then applying functional labels. In either case, functional labeling is performed based on document-specific rules and using many features derived from the document. These features may include the relative and absolute position of a block on a page, the relative and absolute position of a field within a block, general typesetting rules of spacing, and the size and font style of the text. The formatting rules used to produce the page are those that are used in the reverse process of functional labeling. One problem is just the wide variation of formatting style across different documents. However, despite this, most humans can easily

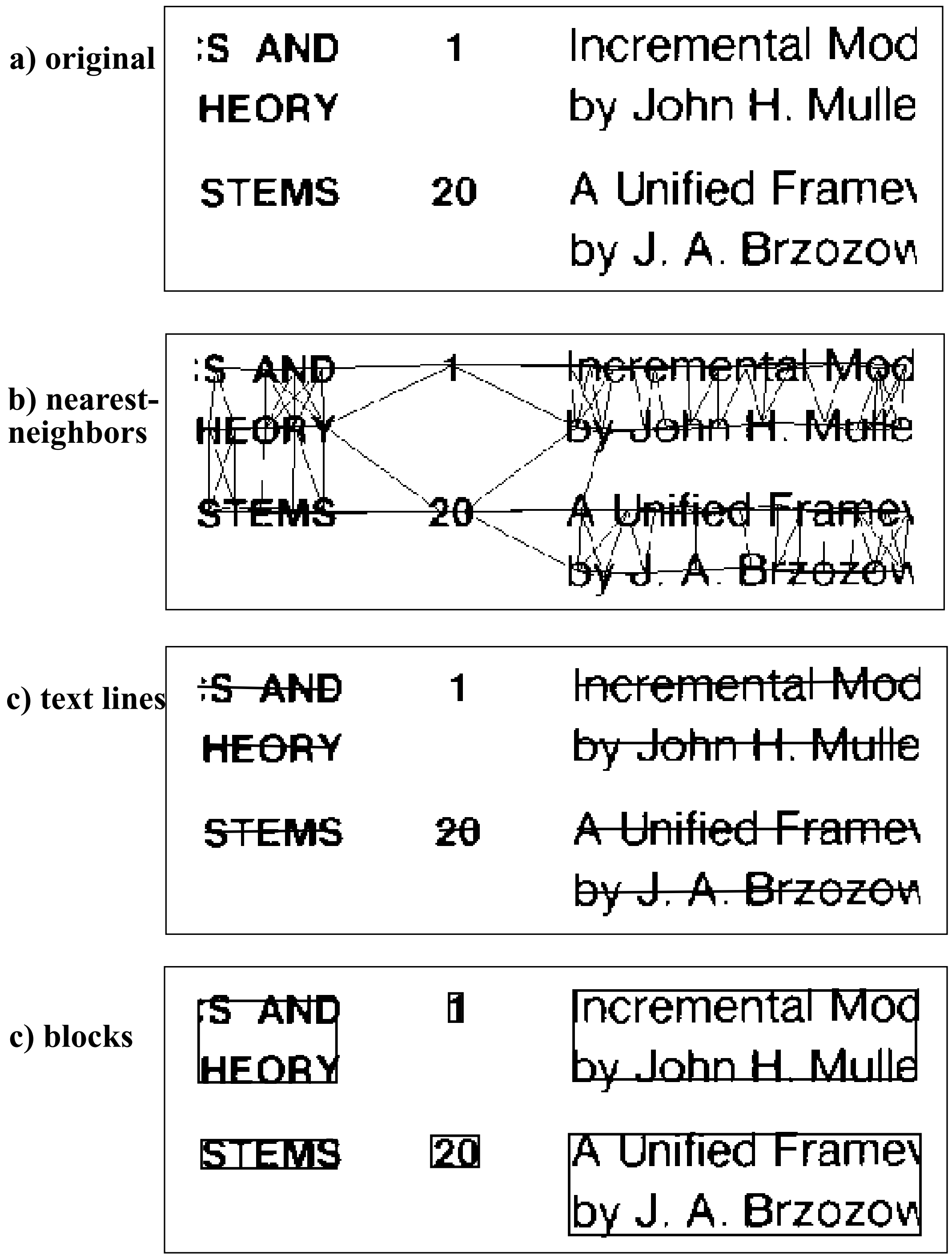


Figure 4. The docstrum method of bottom-up page layout analysis. a) Original portion of table of contents page. b) Nearest neighbors are found of document components. c) Least square fits are made to nearest-neighbor groups to find text lines. d) Structural blocks are found as groups of text lines.

locate blocks of interest using their acquired knowledge of formatting and also by using text recognition.

Due to the large variation of formats and also due to common image processing problems such as noise, erroneous feature detection, and imperfect text recognition results, most work in functional labeling has been restricted to particular domains. Much work has been done in the domain of postal recognition on the problem of locating addresses and address fields, e.g. [Palumbo 1992, USPS 1990]. For U.S. mail, the objective is usually to locate the address block, then to find the ZIP code on the last line, then other parts of the address block from there. This is a good example of a familiar format in which blocks at different locations and fields on different lines connote different functional information. Work has also been done on forms recognition, e.g. [Casey 1992, Taylor 1992], where the problem of structural blocking is facilitated by the lines of the form delimiting blocks. Functional labeling must still be performed to recognize, for instance, that a block at the bottom of a column of blocks is a “total” field, and that indented blocks are intermediate values. In [Dengel 1992], business letters are used as the domain for functional labeling. (See Figure 5.) The ODA (Office Document Architecture, see below) standard for document formatting is used here to hierarchically represent document components. In [Amano 1992], the domain of interest is Japanese journals and patents, and labeling is done in part using rules of relative horizontal and vertical placement, and ranges on the numbers of lines per block. Even within each of these restricted domains, variations of formatting are large so as to present challenging problems.

Since the process of structural and functional layout analysis are the reverse of the document formatting process, it is logical to ask if there is some standard for producing and describing document contents that can be used to perform layout analysis and to test its results. Two of the internationally recognized standards are ODA (Office Document Architecture) [Horak 1985, ISO 1988] and SGML (Standard Generalized Markup Language) [Goldfarb 1990]. Though both describe documents, they are very different in their description. ODA is an object-oriented document architecture that describes both structural and functional components of a complete document. These ODA descriptions are hierarchical, so a layout may be defined as a document, containing pages, and on each page are columns, paragraphs, etc. The functional hierarchy describes a document, which may contain sections or chapters, and each chapter has headings, subheadings, etc. The original purpose of ODA was to create a common standard for interchange of office documents. SGML is a language for “marking up” or annotating text with information that describes it as functional components. It was created for publishing documents. In the publishing process, these functional components are used together with publication-specific rules to format the document in a chosen style. In contrast to ODA, SGML can be thought of more as a bottom-up, description of a document, and this is in keeping with its purpose. Both these standards have grown beyond their original purposes, and, for instance, both are used for document layout analysis. However, while both are used for publishing and in document analysis, they are by no means used exclusively. Many publications use proprietary systems that are compatible with neither ODA or SGML, and most current document recognition research today still does not produce results that conform to any standard.

The use of page layout analysis is currently very limited in commercial document systems. Most general systems provide the capability only of specifying absolute block coordinates on which to scan or perform OCR — that is they do not do automatic page layout analysis. Two exceptions to this are systesms for special-purpose applications: postal reading and forms reading, both to be described more in Section 4.6.

We recommend three papers here. The top-down paper [Pavlidis 91] employs run-length smoothing and segmentation by examining both foreground and background regions (the use of background is similar to Baird’s method [Baird 1990] described above). The paper describing a bottom-up method is O’Gorman’s k-nearest-neighbor, docstrum approach [O’Gorman 1993]. The final paper describes functional labeling and classification to the ODA standard [Dengel 1992].

Other Figure Captions

Figure 2. Page layout analysis by top-down segmentation and labeling. On the left is original page (of IBM Journal of Research and Development) with superimposed cuts of X-Y tree. On the right is the automatically constructed labeled tree. [Nagy 1992-a, p. 59].

Figure 3. Page layout analysis by blank space covers. On the left is the original page showing bounding rectangles over components. On the right is the result of

segmentation showing black rectangles over blank regions from the original page. [Baird 1992, p. 1061].

Figure 5. Results of structural layout analysis are shown in top left, and hierarchical process is shown whereby business letter is functionally labeled, with final results at lower right. [Dengel 1992, p. 65].

References

1. K.Y. Wong, R.G. Casey, F.M. Wahl, "Document analysis system", IBM Journal of Research and Development, Vol. 6, Nov. 1982, pp. 642-656.
2. G. Nagy, S. Seth, "Hierarchical representation of optically scanned documents", Proceedings of the 7th International Conference on Pattern Recognition (ICPR), Montreal, Canada, 1984, pp. 347-349.
3. G. Nagy, S. Seth, M. Viswanathan, "A prototype document image analysis system for technical journals", IEEE Computer, July, 1992-a, pp. 10-22.
4. H.S. Baird, S.E. Jones, S.J. Fortune, "Image segmentation using shape-directed covers", Proceedings of the 10th International Conference on Pattern Recognition (ICPR), Atlantic City, NJ, June 1990, pp. 820-825.
5. J.L. Fisher, S.C. Hinds, D.P. D'Amato, "A rule-based system for document image segmentation", Proceedings of the 10th International Conference on Pattern Recognition (ICPR), Atlantic City, NJ, June 1990, pp. 567-572.
6. R. Esposito, D. Malerba, G. Semeraro, "An experimental page layout recognition system for office document automatic classification: An integrated approach for inductive generalization", Proceedings of the 10th International Conference on Pattern Recognition (ICPR), Atlantic City, NJ, June 1990, pp. 557-562.
7. L. O'Gorman, "The document spectrum for structural page layout analysis", IEEE Trans. Pattern Analysis and Machine Intelligence, Dec., 1993, (in press).
8. G. Nagy, "Toward a structured document image utility", in Structured Document Image Analysis, (H.S. Baird, H. Bunke, K. Yamamoto, eds.), Springer Verlag, Berlin, 1992-b, pp. 54-69.

9. T. Akiyama, N. Hagita, “Automated entry system for printed documents”, Pattern Recognition, Vol. 23, No. 11, 1990, pp. 1141-1154.
10. P.W. Palumbo, S.N. Srihari, J. Soh, R. Sridhar, V. Demjanenko, “Postal address block location in real time”, IEEE Computer, Vol. 25, No. 7, July, 1992, pp. 34-42.
11. USPS — Proceedings of the Fourth United States Postal Service Advanced Technology conference, Washington, D.C., Nov. 1990.
12. R. Casey, D.Ferguson, K. Mohiuddin, E. Walach, “Intelligent forms processing system”, Machine Vision and Applications, Vol. 5, No. 3, 1992, pp. 143-155.
13. S.L. Taylor, R. Fritzson, J.A. Pastor, “Extraction of data from preprinted forms”, Machine Vision and Applications, Vol. 5, No. 3, 1992, pp. 211-222.
14. A. Dengel, R. Bleisinger, R. Hoch, F. Fein, F. Hones, “From paper to office document standard representation”, IEEE Computer, Vol. 25, No. 7, July, 1992, pp. 63-67
15. T. Amano, A. Yamashita, N. Itoh, Y. Kobayashi, S. Katoh, K. Toyokawa, H. Takahashi, “DRS: A workstation-based document recognition system for text entry”, IEEE Computer, Vol. 25, No. 7, July, 1992, pp. 67-71.
16. W. Horak, “Office document architecture and office document interchange formats: Current status of international standardization”, IEEE Computer, October 1985, pp. 50-60.
17. ISO 8613, Information Processing — text and office systems — Office Document Architecture and Interchange format, Parts 1-8, 1988.
18. C. F. Goldfarb, *The SGML Handbook*, Clarendon Press, Great Britain, 1990.
19. H.S. Baird, “Anatomy of a versatile page reader”, Proc. of the IEEE, Vol. 80, No. 7, July 1992, pp. 1059-1065.
20. T. Pavlidis, J. Zhou, “Page segmentation by white streams”, Proceedings of the First Int. Conf. on Document Analysis and Recognition (ICDAR), St. Malo, France, Sept 1991, pp. 945-953.

4.4: Machine-Printed Character Recognition

Keywords: Character recognition, optical character recognition (OCR), word recognition, text recognition, omnifont recognition, lexical analysis, reading machines

Recognition of machine-printed characters, or more simply, printed characters, has been traditionally the most important goal of Optical Character Recognition (OCR) systems. Early systems were generally limited to recognizing characters printed in one font and one size; for example, typewritten text. Such systems relied upon uniform spacing of characters (fixed pitch) to segment individual characters. Each segmented character was then recognized using template matching [Mori 1992] or other similar methods. For such systems to perform satisfactorily, it was essential that there be no rotation, scale change, or other distortion of characters. Although such systems did serve a small niche of applications for a short period of time, it soon became evident that more sophisticated methods were needed to process documents created by laser printers and page typesetters. Thus, recent efforts have focussed on designing systems for omnifont recognition.

Omnifont recognition has been an extremely challenging task to OCR designers. There are over 300 common typefaces with commonly used character sizes ranging from 6 to 26 points, and many other variations such as different weights (light, regular, bold, etc.), ligatures, super and subscripts, etc. [Nagy 92-a]. A common paradigm for character recognition entails the following steps: page images are segmented to locate individual characters, next, discriminating features are extracted for each character, and finally, these extracted features are matched against those of different character classes in the database. Good features are those with small intra-class variance and large inter-class variance. Because of the wide variations in the characteristics of symbol sets from different font families, there are no ideal feature sets which would classify all inputs with 100% accuracy. A large variety of feature sets have been proposed and used in character recognition systems; such systems perform reasonably well as long as the input data are within the constraints assumed during the design or training phase. Many of the techniques that we have described in earlier sections of this book have been used in various stages of such systems. OCR systems and the features used in such systems have become progressively more sophisticated as a result of extensive experimentation using several test databases that have become available in recent years [Bradford 91, Hull 93, Nartker 92, Phillips 93, Toriachi 89, Wilson 90, Yamamoto

86]. We now briefly describe these steps used in character recognition systems.

4.4.1: Character Segmentation

Similarly to many applications in computer vision, the effectiveness of early processing steps, or low-level processing, is critical to the final recognition results. For OCR, it is the initial segmentation of characters that can mean the difference between very good and very poor results. The objective in character segmentation is to identify each individual character within words and sentences.

A common approach to segmenting characters is first to determine a vertical projection profile for each text line. (The projection profile has been described in Section 4.2.1 and an example of a vertical projection profile is shown in Figure 1 of that section.) The vertical projection profile for a text line is a histogram of the number of ON pixels accumulated vertically through words and characters along the text line. Therefore, this profile will have low peaks between words, higher peaks within words, and highest peaks for tall ascending or descending parts of letters. The ideal noiseless profile would have valleys at zero height between characters, indicating character segmentation locations. However, due to noise, characters may be joined or may be broken; or, due to italic font, there may be no vertical line clearly separating characters. Therefore, segmentation cannot be simply performed by identifying zero-value valleys between peaks along the profile.

Approaches to character segmentation usually begin with the projection profile. A threshold is set on the profile, and peaks above the threshold are said to correspond to locations of candidate segmented characters. This threshold may be adaptive; that is, its value may be raised if segmented widths are longer than expected character widths. Recognition is then performed on each candidate character, as described below. If recognition confidence is low (either of an individual character or of a complete word), then re-segmentation can take place to join or separate the initial candidates. This process can be performed iteratively until recognition confidence is high. (See [Tsujimoto 1992] for more detail on character segmentation.)

4.4.2: Features

Although a bit-map of a complete character is of little use as a template for omnifont character recognition, partial templates capturing salient features in a character have been found to be quite useful. For example, templates of regions surrounding the left and right junctions between the horizontal bar and vertical strokes in characters such as H or A are useful to discriminate such characters from others that do not have such junctions. Similarly, simple features such as the number of connected components and holes, ascenders and descenders are also useful for discriminating characters. Other simple features that have been used include aspect ratio of the bounding rectangle, perimeter to area ratio, convex hull and convex deficiencies, horizontal and vertical projection profiles, moment invariants etc. See section 3.5 for more details on these and other feature descriptors. Since bit-map-based features are sensitive to the thickness of character strokes, features based upon the core lines obtained by thinning algorithms have also been tried extensively. Such methods, of course, suffer from the artifacts introduced during the thinning step (see section 3.4). Critical points (e.g., corners, high curvature points, inflection points, junctions etc.) along the contours or on the core lines) as well as contour-based feature representation and matching methods such as Fourier descriptors have also been used to classify characters (see sections 3.3 and 3.5). Many of these simple methods used alone are not adequate to discriminate between characters that look alike (e.g., o and 0, l and 1) and those that are rotated versions of others (e.g., d and p, 9 and 6); they also often fail to correctly discriminate characters of different type families (e.g., Times Roman font characters with serifs versus Helvetica characters without).

4.4.3: Classification

In general, a single feature is seldom adequate to discriminate among all character classes of an alphabet. Instead, typical systems extract several features from each character image and then attempt to classify it based on the similarity of this feature vector with that of a character class. Many well-known decision-theoretic pattern classification methods as well as syntactic and structural methods have been used [Bokser 92, Mori 92, Nagy 92-a]. For example, in Bayesian pattern classification method, each character class is characterized by the conditional probability distribution given by $p(f/c_i)$ where f represents the feature vector and c_i represents character class i. A priori probability of various character classes are also assumed known. Then a measured feature vector $\boldsymbol{f}$ is assigned to class c_k which maximizes the a posteriori probability $p(c_k/\boldsymbol{f})$. The value of each feature is often quantized to take on one of several discrete values to provide a degree of tolerance on measured feature values. Quite often, binary-valued features (denoting presence or absence of corresponding features) are used. Selection of good (i.e, representative) training sets is extremely important to obtain satisfactory results. Efforts to improve the performance of such character-by-character feature extraction and classification methods have long since reached a point of diminishing return. It has now been recognized that any further improvement is obtained by exploiting contextual information [Nagy 92-b]. The classifier in such systems frequently outputs several classes for each input pattern and associates a degree of confidence to each class label. Final class assignment is made after analyzing the outputs from a string of characters (rather than making a decision based on a single character).

4.4.4: Use of Context in Character Recognition

One of the most obvious methods to improve the accuracy of OCR performance is by the use of a dictionary to disambiguate the identity of one character based on the identities of neighboring characters in the same word. In such a system, the classifier outputs a rank ordered list of several alternate labels for each symbol in a word. Those combinations that are not in the dictionary are discarded. In most cases, only one acceptable word remains after this step. In fact, it has been shown that it is possible to assign correct labels to symbols even without creating an initial list of possible labels to symbols [Nagy 87]. In such a method, the classifier simply groups together all symbols in the document that are of similar shape (but does not attempt to recognize the characters). A small sized dictionary is then used to substitute symbols in each group with a unique character; such substitutions are first done for short words since there are only a few possible combinations of such words in the dictionary.

Relative frequency of occurrence of different characters, character-pairs and character strings (n-grams), and words have also been used to enhance OCR performance. Syntax checking, based on grammar, and semantic interpretation of the word with respect to its context, have also been proposed as postprocessing methods to improve raw OCR [Nagy 92-a].

Sometimes an application can have a constrained grammar and vocabulary such that the use of context is very powerful in aiding OCR. A particularly interesting example of applying OCR to a constrained language problem was done for transcripts of chess games [Baird and Thompson 90]. The characters in this application are numbers and chess symbols describing games in the Chess Informant. A sample image of one of the games is shown in Figure 1. Each move that is recognized by OCR is checked for legality in the context of prior and later moves. By using this contextual information, an effective success rate is obtained of 99.995% at the character level and 98% at the complete game level.

We recommend five papers to represent different aspects of work on OCR of printed characters. The paper, “Historical review of OCR research and development,” by Mori, Suen, and Yamamoto [Mori 1992] gives a comprehensive overview and introduction to OCR technologies. The paper, “Omnifont technologies,” by Bokser [Bokser 1992] presents a more detailed description of the methodologies followed in a commercial OCR system. The paper, “A computational model for recognition of multifont word images” by Ho, Hull, and Srihari [Ho 1992] describes the combination of character recognition, segmentation, and word-shape for OCR. The paper, “Major components of a complete text reading system”, by Tsujimoto and Asada [Tsujimoto 1992] describes a document system whose major components are document analysis, understanding, and character segmentation and recognition. Finally, we recommend Baird’s “Document image defect model” [Baird 1992], which objectively details the types of noise and other defects upon which a document image is subjected, providing a model for OCR testing.

Other Figure and Table Captions

Figure 1. A sample text from the Chess Informant showing a game's header and opening moves [from Baird and Thompson 1990].

References

1. S. Mori, C.Y. Suen, and K. Yamamoto, “Historical review of OCR research and development,” Proceedings of the IEEE, 80(7):1029-1058, 1992
2. G. Nagy, “Optical character recognition and document image analysis”, Rensselaer Video, Clifton Park, New York, 1992-a.
3. R.B. Bradford, “Technical factors in the creation of large full text databases,” Department of Energy Infotech Conference, Oak Ridge, Tennessee, May 1991
4. J.J. Hull, “A database for handwritten word recognition research,” to appear in IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993
5. T.A. Nartker, R.B. Bradford, and B.A. Cerny, “A preliminary report on UNLV/GT1: A database for ground truth testing in document analysis and character recognition, Proc. Sympo-

sium on Document Analysis and Information Retrieval, Las Vegas, Nevada, pp. 300-315, 1992

6. I.T. Phillips, S. Chen, and R.M. Haralick, CD-ROM English database design, Department of Electrical Engineering, University of Washington, 1993
7. K. Toriachi, R. Mori, I. Sekita, K. Yamamoto, and H. Yamada, “Handprinted Chinese character database,” in Computer Recognition and Human Production of Handwriting, World Scientific, pp. 131-148, 1989
8. C.L. Wilson and M.D. Garris, “Handprinted character database,” National Institute of Standards and Technology, April 1990
9. K. Yamamoto, H. Yamada, T. Saito, and I. Sakaga, “Recognition of handprinted characters in the first level of JIS Chinese characters,” Proc. 8th International Conference on Pattern Recognition, Paris, pp. 570-572, 1986
10. S. Tsujimoto, H. Asada, “Major components of a complete text reading system”, Proc. of IEEE, July, 1992, Vol. 80, No. 7, pp. 1133-1149.
11. M. Bokser, “Omnidocument technologies,” Proceedings of the IEEE, 80(7):1066-1078, 1992
12. G. Nagy, “At the frontiers of OCR,” Proceedings of the IEEE, 80(7):1093-1100, 1992-b.
13. G. Nagy, S. Seth, and K. Einspahr, “Decoding substitution ciphers by means of word matching with application to OCR,” IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9(5):710-715, 1987
14. Baird, H.S., and K. Thompson, “Reading Chess”, IEEE Trans. PAMI, 12(6):552-559, June 1990
15. T. K. Ho, J. J. Hull, S. N. Srihari, “A computational model for recognition of multifont word images”, Machine Vision and Applications, Vol. 5, No. 3, 1992, pp. 157-168.
16. H. S. Baird, “Document image defect models”, in *Structured Document Image Analysis*, (H. S. Baird, H. Bunke, K. Yamamoto, eds.), Springer-Verlag, Berlin, 1992, pp. 546-556.

4.5: Handwritten Character Recognition

Keywords: Handwritten character recognition, Optical character recognition (OCR), handwriting analysis, handprinted character recognition, text recognition, pen-based computers, off-line recognition, on-line recognition, word recognition

While many of the methods for handwritten character recognition are similar to those of printed character recognition, there are some important differences. First, there are two distinct approaches for capturing data for handwriting recognition. In on-line recognition systems, captured data is represented as a time sequence of the position of pen tip, thus preserving rate of movement of pen as well as its position. In off-line methods, data is captured by conventional scanning and digitizing methods similar to those used in printed character recognition. Second, the segmentation process to isolate individual characters in cursive text is a major component of handwriting recognition systems (whereas simple connected component analysis methods are adequate for such separation in case of printed character recognition). Third, the variety of ways in which text is written is essentially unlimited since the same text written by the same writer at two different times is seldom identical in all respects. Handwriting recognition is an extremely active area of research at this time and the reader is referred to the proceedings of the international workshop on handwriting recognition [IWFHR].

It has been argued that additional temporal information (order in which strokes are written as well as the relative velocities of writing different parts of a stroke) readily available in on-line systems should facilitate design of improved recognition algorithms [Tappert 90]. This line of research has received increased attention in recent years due to the availability of low-cost electronic writing tablets and pen-based computers. For large alphabet languages such as Chinese, a pen-based system with on-line character recognition is an attractive alternative to character composition using keyboard. On-line recognition systems are also useful as writer identification and signature verification systems.

As noted earlier, one of the most difficult challenges for off-line handwriting recognition is that of connected character segmentation. One of the strategies described in [Bozinovic 89] to solve this problem is initially to presegment the word into minimal portions along the horizontal direction followed by a sequence of operations to hypothesize various letter assignments to one or more consecutive presegments; finally, lexical analysis (matching against a dictionary) is applied to recognize the complete word. Off-line text recognition has been an important practical problem for postal applications and research progress in this area has been described extensively in the literature.

We recommend representative papers describing off-line and on-line work in handwriting recognition: “Off-line cursive script recognition” by Bozinovic and Srihari [Bozinovic 1989], and “On-line recognition of handprinted characters: survey and beta tests” by Nouboud and Plamondon [Nouboud 1990] and “Large vocabulary recognition of on-line handwritten cursive words” by Seni et al. [Seni 1996]. For a discussion of the commercial state of handwriting recognition, see Section 4.7.

References

1. IWFHR, Proceedings of the IAPR International Workshops on Frontiers in Handwriting Recognition (Montreal, Canada, 1990, Bonnas, France, 1991, Buffalo, New York, 1993)
2. C.C. Tappert, C.Y. Suen, and T. Wakahara, “The state of the art in on-line handwriting recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-12(8):787-808, 1990
3. R.M. Bozinovic and S.N. Srihari, “Off-line cursive script word recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-11(1):68-83, 1989
4. F. Nouboud, R. Plamondon, “On-line recognition of handprinted characters: survey and beta tests”, Pattern Recognition, Vol. 23, No. 9, pp. 1031-1044, 1990.
5. G. Seni, R. K. Srihari, N. Nasrabadi, “Large vocabulary recognition of on-line handwritten cursive words,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, No. 7, July 1996, pp. 757-762.

4.6: Document and Forms Processing Systems

Keywords: document processing, forms processing, electronic library systems, postal processing systems, forms processing systems

A document processing system embodies many of the image and document processing techniques that have been described here. Typically, pixel level processing is first performed to reduce noise and modify the pixel data into a form (such as a binary or thinned image) that best facilitates subsequent analysis. Pertinent features are then found from this data to more succinctly describe the pixel regions contained in the image. These features are used to recognize textual format and content. For recognition of tables and forms, graphics recognition techniques (described in the next chapter) may also be employed to obtain information from the delimiting lines. An additional aspect of complexity with complete documents is that of multiple pages: continuing text across pages, groups of pages (e.g. chapters), etc. Though there are document representations for handling the production of multiple-page documents (e.g. SGML and ODA), this is a problem that has usually not been handled by document analysis systems.

One application of document processing techniques is to image-based electronic libraries. Three recent systems for storage and display of technical journals include [Nagy 1992], the RightPages system [Story 1992], and the Document Recognition System (DRS) [Amano 1992]. Some of the technical work associated with these systems includes noise reduction, structural and functional page layout analysis, OCR, OCR enhancement, and image subsampling for readable display of text. User interface design is also an important aspect of each of these systems, and document systems in general.

Document processing of postal mail is an important application area that is probably the most economically important application of current document system implementation. The objectives of these systems are to locate address blocks and read at least some portion of the addresses for automatic routing of each piece. One important aspect of this problem is that recognition and routing must be performed very quickly to compete with human operations. Some of the work in this area includes [Palumbo 1992] on real-time address block location, [Matan 1992] on a neural network system for reading ZIP codes, and [Kimura 1992] on segmentation and recognition of ZIP codes.

The application of forms recognition entails document images whose formats are more restricted than mail or journal publications, because fields are located in specified locations that are bounded by lines or highlighting. With the knowledge of locations of blocks of interest, these can be found and the contents recognized. Though this may seem a simpler problem than that of non-delineated block location, there are problems inherent to forms reading that are distinct from other applications. Two of these problems are location of boxes, usually by graphics recognition techniques upon the lines, and dealing with text that is misaligned with or intersecting the boxes. Also, since forms are often filled out by hand, handwriting recognition may be involved. Some work in forms recognition includes [Casey 1992] and [Taylor 1992]. These systems have been demonstrated on U.S. tax forms, among other data.

We recommend five papers describing different document systems: [Casey 1992, Taylor 1992] on forms reading, [Nagy 1992] describing an image-based electronic library, [Palumbo 1992] on a real-time postal document system, and [Schürmann 1992] on a general, textual document system.

References

1. G. Nagy, S. Seth, M. Viswanathan, “A prototype document image analysis system for technical journals”, IEEE Computer, Vol. 27, No. 7, July, 1992, pp. 10-22.
2. G. Story, L. O'Gorman, D. Fox, L.L. Schaper, H.V. Jagadish, “The RightPages image-based electronic library for alerting and browsing”, IEEE Computer, Vol. 25, No. 9, Sept., 1992, pp. 17-26.
3. T. Amano, A. Yamashita, N. Itoh, Y. Kobayashi, S. Katoh, K. Toyokawa, H. Takahashi, “DRS: A workstation-based document recognition system for text entry”, IEEE Computer, Vol. 25, No. 7, July, 1992, pp. 67-71.
4. P.W. Palumbo, S.N. Srihari, J. Soh, R. Sridhar, V. Demjanenko, “Postal address block location in real time”, IEEE Computer, Vol. 25, No. 7, July, 1992, pp. 34-42.
5. O. Matan, H.S. Baird, J. Bromley, C.J.C. Burges, J.S. Denker, L.D. Jackel, Y. Le Cun, E.P.D. Pednault, W.D. Satterfield, C.E. Stenard, T.J. Thompson, “Reading handwritten digits: A ZIP code recognition system”, IEEE Computer, Vol. 25, No. 7, July, 1992, pp. 59-63.

6. F. Kimura, M. Shridhar, “Segmentation-recognition algorithm for ZIP code field recognition”, Machine Vision and Applications, Vol. 5, No. 3, 1992, pp. 199-210

7. R. Casey, D.Ferguson, K. Mohiuddin, E. Walach, “Intelligent forms processing system”, Machine Vision and Applications, Vol. 5, No. 3, 1992, pp. 143-155.

8. S.L. Taylor, R. Fritzson, J.A. Pastor, “Extraction of data from preprinted forms”, Machine Vision and Applications, Vol. 5, No. 3, 1992, pp. 211-222.

9. J. Schürmann, N. Bartnesck, T. Bayer, J. Franke, E. Mandler, M. Oberländer, “Document analysis — from pixels to contents”, Proc. IEEE, Vol. 80, No. 7, July 1992, pp. 1101-1119.

4.7: Commercial State and Future Trends

We began in the Preface of this book by saying that the prevalence of fast computers, large computer memory, and inexpensive scanners has fostered an increasing interest in document image analysis. Hardware advances will continue to provide faster computation, greater storage, and easier computer input such as to improve the tools of document analysis. It is thus with some sense of dissatisfaction that the algorithms of the field seem to progress with less speed. However, this is not so much due to a slow rate of improvement as with the distance still required to reach our overall goal: recognition results similar to a human's. This will not happen soon.

Having stated that we have a far distance to go to reach the ultimate goal, we can report intermediate progress that has proven to be very productive for many applications. Furthermore, we can suggest where progress will continue and where it is especially needed:

- Most documents are scanned in binary form today, though gray-scale scanners are becoming increasingly prevalent. In the future, more gray-scale and color scanning will be employed. The primary advantage of this is simply more pleasing images, especially those with gray-scale and color pictures. However, this will also enable more sophisticated document analysis to be performed. Thresholding will be improved when more complex algorithms can be applied digitally rather than the simpler optical methods used today. Multi-thresholding can be done to separate, for instance, black text from red text from blue highlight box from white background. Segmentation can be performed to distinguish among text, graphics, and pictures.
- Noise processing to remove salt-and-pepper noise is available on current systems. Future systems will employ more sophisticated filters that will use some knowledge of the document components to provide recognition-based cleaning. The furthest extent of this, of course, is to just perform OCR along with font and format recognition, then redisplay a page exactly as before. The problem here is that the recognition must be flawless so as not to introduce more than the initial amount of noise.
- Document scanners and systems can be purchased today with software to correct for the small amount of skew often present when placing documents on the scanner, up to about 10 degrees. Future systems will be able to re-orient documents from any skew angle, even upside

down. Related to the increased ease and utility of scanning systems is the improvement and reduction of size in the hardware. There are portable and hand-held scanners (usually packaged with OCR software) that are currently available. In the future, scanners will be incorporated into portable computers so the portable office — computer, fax machine, copying machine, and telephone — are always available.

• While rudimentary structural layout analysis is already performed in some document systems (mainly to separate text columns), more complex tasks such as paragraph and caption segmentation will also be performed. On a further horizon is structural layout analysis combined with functional labeling. Since the latter requires knowledge of the document, automatic training procedures will be developed to facilitate this task. Document description will be made in some standard language such as SGML or ODA.

• For printed text, commercial OCR boxes currently yield 99.5% to 99.9% recognition accuracy of individual characters for high quality text [Rice 1993, Nartker 1994]. This recognition rate drops off to 90% to 96% accuracy for full words. Although these rates may appear to be quite good at first glance, they correspond to one incorrectly recognized word for every line or two of machine-printed text. We expect to see major progress in accuracy not by improved techniques for individual character recognition but by the use of word- and document-based information to augment that raw recognition. Currently, dictionary look-up is used to change unrecognized words into correct ones that have similar word features. In the future, increased computer power and memory (plus fast indexing methods) will enable practical searching of large spaces so correction can be made using syntactic and even semantic knowledge of the text. Even with these improvements, we don't expect to see flawless OCR in the near future. Near-flawless OCR will be available only for constrained situations, such as a limited character set of 0 to 9. Erroneous OCR can be tolerated in some applications such as one to search for keywords in an article where pertinent keywords will likely be repeated. (The University of Nevada at Las Vegas, UNLV, performs annual evaluations of OCR products. The evalulations are based on character and word recognition results for a range of document conditions such as skew and document quality. In 1993, systems from eight vendors were tested. We have referenced the current report here, but for up-to-date evaluations of OCR systems, the latest report can be requested from UNLV.)

• The bulk of application for most current OCR is on the standard character sets, in particular the Roman, Kanji, and Arabic alphabets, and numerals. However, there are other character sets and symbols that will be recognized in future systems. An example is the larger symbol set used for mathematics. A further problem for mathematical equations is that character sequence is not strictly linear along a text line: there are subscripts and superscripts and bounds of integrals and summations that are above and below other symbols. One more problem with equations is simply that sub- and super-scripts are usually in a smaller font size, making the problem of recognition even more difficult.

• For handwritten text, rates are lower than for printed text. Current off-line recognition rates for untrained data are about 96.5% for digits (0-9), 95% for upper letters only, and 86.5% for lower letters, all for isolated characters [Wilkinson 1992]. On-line recognition rates are between those of off-line handwritten text and machine-printed text, but are usually highly dependent upon training and the size of the character set. Although these recognition rates are unsatisfactory for many applications, there are those applications that do very well with them. For instance, for the postal application, though zip code reading is currently only about 70% accurate, this still reduces by a huge amount the mail that must be hand sorted. Besides applications like these, the most predominant applications of handwriting recognition in the short term will be highly constrained. These include trained and untrained digit recognition, recognition of pre-segmented block characters (from a form, for instance), signature storage and recognition, and trained, single-user handwriting recognition. Many of the other comments made above for printed text OCR also apply for handwritten text.

• Current systems use different techniques for machine and handwritten text recognition. Different approaches are also often used for different alphabets (e.g. Kanji, Arabic, Roman). Even tables and mathematical equations are most often analyzed using different modules (where they are analyzed at all in current commercial systems). A goal in character recognition is seamless and automatic multi-lingual, multi-symbol recognition where language and character type are first recognized then characters are recognized quickly and correctly.

References

1. S. V. Rice, J. Kanai, T. A. Nartker, "An evaluation of OCR accuracy," in UNLV Information

Science Research Institute 1993 Report; this is an annual report of the University of Nevada at Las Vegas on the current state of document (in particular OCR) technology.

2. T. A. Nartker, S. V. Rice, J. Kanai, “OCR accuracy: UNLV’s second annual test”, Inform, January, 1994, pp. 40-45.

3. R. A. Wilkinson, et al. “The first census optical character recognition systems conference,” report from the conference sponsored by the U.S. Bureau of the Census and the National Institute of Standards in May, 1992; NIST 4912.

Chapter 5

Recognizing Components of Graphics Documents

5.1: Introduction

In this chapter we deal with methods for analysis and recognition of graphics components in paper-based documents. Graphics recognition and interpretation is an important topic in document image analysis since graphics elements pervade textual material, with diagrams illustrating concepts in the text, company logos heading business letters, and lines separating fields in tables and sections of text. The graphics components that we deal with are the binary-valued entities that occur along with text and pictures in documents. We also consider special application domains in which graphical components dominate the document; these include symbols in the forms of lines and regions on engineering diagrams, maps, business charts, fingerprints, musical scores etc. The objective is to obtain information to semantically describe the contents within images of document pages.

We have stated that a high level description of the document is the objective, but why is this good and how is this used? One application is in the conversion of paper diagrams to computer readable form. Though simply digitizing and storing the document as a file of pixels makes it accessible by computer, much more utility is usually desired. For instance, for modification of the elements of the diagrams, a higher level description is required. For translation to different computer-aided design (CAD) formats and for indexing of components, the diagram must be semantically described. Even if storage is the only requirement, a semantic representation is usually much more compact than storing the document as a file of pixels. For example, just as an English letter is stored as its 8-bit ASCII representation in lieu of its larger-size image, for a compression of two to three orders of magnitude, a graphics symbol such as a company logo or electrical “AND” gate can also have a similarly compact “codeword” that essentially indexes the larger size image. Document image analysis can be important when the original document is produced by computer as well. Anyone who has dealt with transport and conversion of computer files knows that compatibility can rarely be taken for granted. Because of the many different languages, proprietary systems, and changing versions of CAD and text formatting packages that are used, incompatibility is especially true in this area. Because the formatted document — that viewed by humans — is semantically the same independent of the language of production, this form is a “protocol-less protocol”. If a document system can translate between different machine-drawn formats, the next objective is to translate from hand-drawn graphics. This is analogous to handwriting recognition and text recognition in OCR. When machines can analyze complex hand-drawn diagrams accurately and quickly, the graphics recognition problem will be solved, but there is still much opportunity for research before this goal will be reached.

A common sequence of steps taken for document image analysis of graphics interpretation is similar to that for text. Preprocessing, segmentation, and feature extraction methods such as those described in earlier chapters are first applied. An initial segmentation step that is generally applied to a mixed text/graphics image is that of text and graphics separation. An algorithm specifically designed for separating text components in graphics regions irrespective of their orientation is described in [Fletcher and Kasturi 88]. This is a Hough transform-based technique that uses the heuristic that text components are colinear. Once text is segmented, typical features extracted from a graphics image include straight lines, curves, and filled regions. After feature extraction, pattern recognition techniques are applied, both structural pattern recognition methods to determine the similarity of an extracted feature to a known feature using geometric and statistical means, and syntactic pattern recognition techniques to accomplish this same task using rules (a grammar) on context and sequence of features. After this mid-level processing, these features are assembled into entities with some meaning — or semantics — that is dependent upon the domain of the particular application. Techniques used for this include pattern matching, hypothesis and verification, and knowledge-based methods. The semantic interpretation of a graphics element may be different depending on domain; for instance a line may be a road on a map, or an electrical connection of a circuit diagram. Methods at this so called high level of processing, are sometimes described as artificial intelligence techniques.

Most commercial OCR systems will recognize long border and table lines as being different from characters, so will not attempt to recognize them as characters. Graphics analysis systems for engineering drawings must discriminate between text and graphics (mainly lines). This is usually accomplished very well except for some confusion when characters adjoin lines, causing them to be interpreted as graphics; or when there are small, isolated graphics symbols that are interpreted as characters.

We point the reader to one reference containing recent work in graphics recognition [Kasturi and Tombre, 1995].

References

1. Fletcher, Kasturi, "A robust algorithm for text string separation from mixed text/graphics images", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 10, No. 6, Nov. 1988, pp. 910-919.
2. R. Kasturi, K. Tombre (ed.s) *Graphics Recognition -- Methods and Applications*, Lecture Notes in Computer Science, Springer Verlag, 1995.

5.2: Extraction of Lines and Regions

Keywords: line graphics, region graphics, segmentation

The amount and type of processing applied to graphics data in a raster image is usually application dependent. If the graphics image is a part of a predominantly textual document and the objective is simply data archiving for later reconstruction, then use of any one of the well known image compression techniques [Jain 89] is adequate. On the other hand, if information is to be extracted from data for such applications as indexing image components from a pictorial database [Grosky and Mehrotra 89], modification of graphics in a CAD system [Karima et al. 85], or determining locations in a geographical information system, then extensive processing to extract objects and their spatial relationships is necessary. Such a level of description is attained through a series of intermediate steps. After the graphics data have been segmented, the next step is to perform processing to locate line segments and regions. Extracting the location and attributes of these is an important step for graphics interpretation. In this section, we describe methods for extraction of regions and lines.

We first define how regions and lines differ in this context. A group of ON-valued pixels can be a region or a line. Consider the drawing of a barbell, with two filled disks joined by a line of some thickness. The two disks are regions that can be represented by their circular boundaries (from contour detection). The thick line between can be represented by its core-line (from thinning). The representation of this drawing by boundaries for regions and core-lines for the line has advantages in later steps of graphics analysis. Though lines can have thicknesses and regions can sometimes be described by the core-lines that form their “skeleton”, the distinction between the two is usually obvious based on meaning within the domain.

To represent an image by thin lines and region boundaries, the lines and regions must first be segmented. One approach to segmentation is by erosion and dilation operations [Harada et al. 85, Bow and Kasturi 90, Nagasamy and Langrana 90, Modayur et al. 93]. (See also Section 2.3 where these morphological operations are described in the context of noise reduction.) Erosion and dilation usually proceed iteratively, deleting or adding a one-pixel thickness from the boundary on each iteration until the specified thickness is reached. For graphics segmentation, erosion precedes dilation, and the thickness removed is specified to be greater than the maximum line thickness in the image so that lines disappear. Thus, the result of erosion is an image containing only eroded regions. Then when dilation is performed, this only takes place around these remaining regions. If the image result of these erosion and dilation steps is combined by a logical AND operation with the original image, only the regions remain. (Note that, since dilation is not an exact inverse of erosion, the image should be dilated a thickness of one or two more pixels than it is eroded to completely recover the regions.) The difference image obtained by subtracting this region image from the original image then contains the remaining thin line components. After segmentation, the boundaries are easily found in the region image using contour following methods. Lines are thinned to obtain their core-lines.

There are other methods for segmentation; some that simultaneously extract boundaries and core-lines. In [Shih and Kasturi 89] several picture decomposition algorithms for locating solid symbols are evaluated. In one method, Wakayama's Maximal Square Moving (MSM) algorithm [Wakayama 82], squares are fit to the ON-valued pixel regions. The union of these squares (possibly overlapping with one another) exactly cover these regions. Thus, knowing the center and size of these maximal squares it is possible to exactly reconstruct the original image. The centers of these squares may also be connected to form an approximation to the core-line; however, since these squares are of different sizes, in general, their centers are generally not connected. A system of pointers are necessary to keep track of squares that are adjacent to one another in order to generate the core-lines and boundaries. This process, along with the procedure for keeping track of the squares and their attributes as they are being grown is quite unwieldy. A simpler method in which core-lines can be found simultaneously with boundaries is the *kxk* thinning method [O'Gorman 90].

We defer recommendations on references to the following section, where the systems papers deal with this issue in an application-dependent fashion.

References

1. Jain, A.K., *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989
2. Grosky, W.I. and R. Mehrotra, Special Issue on Image Database Management, Computer,

22(12), 1989

3. Karima, M., K.S. Sadhal, and T.O. McNeil, “From paper drawings to computer aided design,” IEEE Computer Graphics and Applications, pp. 24-39, Feb. 1985
4. Harada, H., Y. Itoh, and M. Ishii, “Recognition of free-hand drawings in chemical plant engineering,” Proc. IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management, pp. 146-153, 1985
5. Bow, S. and R. Kasturi, “A graphics recognition system for interpretation of line drawings”, in Image Analysis Applications, R. Kasturi and M.M. Trivedi (eds.), Marcel Dekker, 1990
6. Nagasamy, V., and N.A. Langrana, “Engineering drawing processing and vectorization system”, Computer Vision, Graphics and Image Processing, 49:379-397, 1990
7. Modayur, B.R., V. Ramesh, R.M. Haralick, and L.G. Shapiro, “MUSER: A prototype musical score recognition system using mathematical morphology,” Machine Vision and Applications, 6(2):, 1993
8. Shih, C-C. and R. Kasturi, “Extraction of Graphic Primitives from Images of Paper-based Drawings,” Machine Vision and Applications, 2:103-113, 1989
9. Wakayama, T., “A core line tracking algorithm based on maximal square moving,” IEEE Trans. PAMI, 4(1):68-74, Jan. 1982
10. O'Gorman, L., “kxk Thinning”, Computer Vision, Graphics, and Image Processing, 51:195-215, 1990

5.3: Graphics Recognition and Interpretation

Keywords: Line drawing analysis, engineering drawing analysis, graphics recognition, raster to vector conversion, vectorization, line drawing interpretation, fingerprint analysis, music reading, electronic circuit diagram conversion, map reading, map conversion

Recognition of graphical shapes and their spatial relationships is an important final task in most document analysis systems. The recognition process essentially consists of two main steps: processing the input image to obtain representational primitives (as described in the previous sections), and matching these primitives against similar primitives derived from known models. Techniques used for the latter step of matching are strongly application-dependent. To recognize isolated symbols of fixed size and orientation, simple template matching applied directly to a bit-mapped image may be adequate. However for many applications, this simple technique is inappropriate, and features as described above must be extracted. In certain applications, it may be adequate to approximate closed contours by polygons. In others, more sophisticated algorithms that hypothesize possible matches, compute scene/model transformations, and verify the hypotheses are used. In more complex images such as maps and engineering drawings, context-dependent, knowledge-based graphics recognition and interpretation techniques have been used. Different algorithms exhibit varying degrees of flexibility, accuracy, and robustness.

In this section, various graphics recognition and interpretation techniques are described, including hierarchical decomposition and matching, interpretation based on structural analysis of graphics and recognition using contextual information and domain-dependant knowledge. The techniques described here make use of the line and feature data obtained using techniques described in the previous sections. Recognition algorithms that operate directly on bit-mapped data or those that are based on well known techniques such as signature analysis, Fourier descriptors, etc. are not described here (see Section 3.5). Many of the methods described here were originally developed for use in a specific application domain although they can be adapted for use in other domains; thus, we organize this section by application domains.

5.3.1: Recognition of graphical shapes in line art

Many business documents routinely include line art in the form of organizational charts, bar graphs, block diagrams, flow charts, logos, etc. Techniques such as polygonal approximation are useful to recognize simple isolated shapes found in such documents. For complex shapes or in situations in which different shapes are interconnected, structural analysis methods are useful.

Structural analysis methods, for the most part, use a bottom up approach where the input pixel image of the document is first processed to obtain vectorized line and feature data. Some form of structural analysis is then performed on this vectorized data to extract sensible objects. What is a sensible object depends on the domain of the application. It could be an electrical circuit symbol, a standard geometric figure like a rectangle or a pentagon, a component in an engineering drawing etc. The extracted object may then be compared with a database of generic objects for recognition. Structural analysis refers to the actual search that is performed on the input data for extracting meaningful objects. This analysis stage can be very time consuming due to the large number of line segments and other feature primitives present in the vectorized data and the enormous number of ways in which they could be combined to potentially form sensible objects. A simple brute force but an highly inefficient method would be to systematically check all possible combinations to see if any of them form an object of interest in our domain. As it can be clearly seen, what is needed at this stage is an intelligent method that would make use of heuristics or context sensitive information to speed up the analysis. The various structural analysis based graphics recognition systems described in the literature primarily differ in the way they perform this analysis of vectorized line and feature data.

For recognition of line art such as flow-charts, tables, and block diagrams, structural analysis of lines and their interconnections is required to generate meaningful and succinct descriptions. For example, the flow chart shown in Figure 1 contains only 57 line segments (excluding hatching lines and filling patterns); but they form 74 different closed loops [Kasturi et al. 90]. A meaningful interpretation would be to identify the seven simple shapes, and describe other lines as interconnecting segments. Similarly, an unconstrained algorithm could interpret a table as a collection of many rectangles, whereas it would be more useful to describe it as a rectangle with horizontal and vertical bars. In case of electrical schematics it would be necessary to separate lines belonging to symbols from connecting lines.

An algorithm for generating loops of minimum redundancy is described in [Kasturi et al. 90]. In this algorithm, all terminal line segments are first separated since they cannot be a part of any closed loop. This removes the two short line segments at the top and bottom of Figure 1. All self loops are then separated (top rectangle and small shape fillings in Figure 1). This process is repeated, if necessary (to delete the short segment at the bottom of the top rectangle in Figure 1). A heuristic search algorithm is then applied to the remaining network to identify simple closed loops that are made up of no more than a predetermined number of line segments. This search algorithm is described with the help of Figure 2. In this figure, the current point C has been reached by following line segments AB and BC. The objective is to assign priorities to line segments at C for continuing the search for a closed loop. Line segment j is assigned the highest priority since it is collinear with BC. Next priority is assigned to k (angles a1 and a2 are equal) since it has the potential to form a regular polygon. Next priority goes to segment l that is parallel to AB (potential parallelogram, trapezoid). Final priority goes to segment m since it forms the sharpest convex corner at C. If none of these segments continues the path to form a closed loop, segment n is chosen during the final search for a closed loop. This loop finding algorithm employs a depth first search strategy. It correctly identifies the seven simple shapes that are present in the figure. In particular, the algorithm traces the outline of the hatched rectangle and thus separates hatching patterns from enclosing lines. The loops that are extracted are then compared with a library of known shapes shown in Table 1 for recognition and description. Those shapes that are not recognized are analyzed to verify if they can be described as partially overlapped known shapes. All other segments are described as interconnecting lines or hatching patterns. The system outputs all recognized shapes, their attributes and spatial relationships. The output generated by the system corresponding to the test image of Figure 3 are shown in Tables 2 and 3. The algorithm has been extended to describe graphics made up of circular arc segments. Some of these techniques have also been used in a map-based geographic information system [Kasturi et al. 89]. For a description of a method for interpretation of tables see [Chandran and Kasturi 93].

Occasionally it would be necessary to recognize graphical shapes in which a portion of the shape is missing or hidden behind other shapes. In such a situation methods that do not require complete object boundaries are required. Also, when the number of possible different shapes to be recognized is large, it may be efficient to represent complex parts as a combination of already known simple shapes along with their spatial relationships. An object recognition system that creates a library of parts by hierarchical decomposition is suitable for use in such situations. In this method, the library organization and indexing are designed to avoid linear search of all the model objects.

The system has hierarchical organization for both structure (whole object-to-component subparts) and scale (gross-to-fine features). Features used are corner, end, crank, smooth-join, inflection and bump, which are derived from discontinuities in contour orientation and curvature. Subparts consist of subsets of these features, which partition the object into components. The model libraries are automatically built using the hierarchical nature of the model representations. A constrained search scheme is used for matching scene features to model features. Many configurations are pruned in the search space early in the search process using simple geometric constraints such as orientation difference, distance, and direction between pairs of features.

5.3.2: Conversion of Engineering Drawings and Maps

[Antoine et al. 92] describes REDRAW, a system for interpretation of different classes of technical documents. It uses *a priori* knowledge to achieve interpretation at a semantic level. Here the aim is to build a general model-driven recognition system that can be completely parameterized. The model contains specific knowledge for each document class. The interpretation system is then driven by this model using a general tool box for low level operations. Two applications, city map and mechanical drawing interpretation are described. The *a priori* knowledge about the domain induces particular interpretation process for each document class. Among the low level operations they describe a method for extraction of parallel lines (hatched areas). A single scan algorithm in horizontal and vertical directions determines an order relation for each extremity point for each segment. From these relations, classes of parallel lines in a same area are deduced by assuming that parallel lines from a hatched area must overlap each other in the two main directions. A prototype system for extracting higher level structures for knowledge-based analysis of mechanical drawings is described in [Vaxivière and Tombre 92] which is reprinted in this collection.

[Joseph and Pridmore 92] describe a schema-driven system called ANON, for the interpretation of engineering drawings. Their approach is based on the combination of schemata describing prototypical drawing constructs with a library of low-level image analysis routines and a set of explicit control rules. The system operates directly on the image without prior thresholding or vectorization, combining the extraction of primitives (low level) with their interpretation (high level). ANON integrates bottom-up and top-down strategies into a single framework. The system has been successfully applied to piece-part drawings. We have also included this paper in this volume.

One of the major tasks in automated conversion of engineering drawings is that of recognizing and segmenting arcs. [Dori 92] describes the Machine Drawing Understanding System (MDUS) for this purpose. Vectorization is first done by the orthogonal zig-zag algorithm (described in the reference) to obtain connected segments through the lines of the diagrams. Arcs are detected by finding chains of triplets of vector segments. Their centers are found iteratively, and from this, the radius of curvature. Since the approach deals with line features (from vectorization) rather than pixels, it is both efficient and effective.

Another task in automated conversion of mechanical engineering drawings is that of separating dimensioning lines and their associated text from object lines. Algorithms for performing this segmentation are described in [Lai and Kasturi 93]. The complete system includes algorithms for text/graphics separation, recognition of arrowheads, tails, and witness lines, association of feature control frames and dimensioning text to the corresponding dimensioning lines, and detection of dashed lines, sectioning lines and other object lines. Similar methods have also been applied to interpret telephone system manhole drawings [Arias et al. 93].

A method for interpreting the 3-D shape of an object corresponding to multiple orthographic views in an engineering drawing has been described in [Lysak and Kasturi 90]. The technique is based on a bottom-up approach in which candidate vertices and edges are used to generate a set of possible faces, which are in turn assembled into enclosures representing the final object. A minimum of two views is required, and a maximum of six orthogonal views in a standard layout can be accommodated. All possible interpretations consistent with the input views are found, and inconsistent input views are recognized. The method can handle input views with small drafting errors and inaccuracies.

[Ejiri et al. 90] apply structural analysis methods for recognition of engineering drawings and maps. To process LSI cell diagrams, solid lines and two types of broken lines in any one of six colors are recognized using the color digitizer described earlier. A loop-finding routine facilitates detection of overlapped lines denoted by a special mark. Structural analysis methods have also been used to recognize characters and symbols in logic circuit diagrams, chemical plant diagrams, and mechanical part drawings.

A system for automatic acquisition of land register maps is described in [Boatto et al. 92]. The semantics of land register maps are extensively used to obtain correct interpretation in this system.

We have included a reprint of this paper in this collection.

5.3.3: Conversion of Electronic Circuit Diagrams

[Fahn et al. 88] describe a topology-based component extraction system to recognize symbols in electronic circuit diagrams. The objective is to extract circuit symbols, characters and connecting lines. Picture segments are detected using segment tracking algorithms. These are then approximated using a piecewise linear approximation algorithm. A topological search is done to form clusters of symbols or characters. A decision tree is used to assign picture segments to a class of primitives. Segments are clustered into component symbols using a context-based depth-first search method. The system is designed to recognize circuit symbols in four orientations, and connection lines that are horizontal, vertical or diagonal. The system has also been used to generate fair copies of hand-drawn circuit diagrams.

A model-based learning system for recognizing hand-drawn electrical circuit symbols in the absence of any information concerning the pose (translation, rotation and scale) is proposed by [Lee 92]. A hybrid representation called, attributed graph (AG), which incorporates structural and statistical characteristics of image patterns, is used for matching an input symbol with respect to model symbols. Model AG's are created interactively using a learning algorithm. The poses of input object AG's are estimated based on a minimum square error transform and they are classified based on minimum distance. An average error rate of 7.3% is reported.

5.3.4: Other Applications

Document image analysis techniques have also been used to solve problems in many interesting and innovative applications such as classification of finger prints, musical score reading, extraction of data from microfilm images of punched cards, etc. These applications are briefly described in this section.

Fingerprint Analysis: The processing of fingerprint images can be classified as a graphics analysis application where the difference from most other document applications is that fingerprints contain natural (versus machine-made) features. There are two levels of features in a fingerprint. On a “macroscopic” scale, a fingerprint can be classified by the overall pattern of ridges (the lines making up the pattern). There are three main classifications of patterns at this level, a whorl (where the ridges form a bull’s-eye pattern), a loop (where the ridge lines start at a lower corner, flow up and around back to the same corner in the opposite direction that they started), and an arch (where the ridges flow from one lower corner, up, then down to the opposite corner). There are sub-classifications of these patterns, but these are few, usually about ten in all. Also at the macrcoscopic level, one can identify singular points: the center of the fingerprint pattern, called the core (for example, the center of the bull's-eye), and the delta (usually at the lower side of a fingerprint where three macroscopic patterns abut). The other level of features can be termed "microscopic", and involves the individual line features, called minutia. These are ridge endings and bifurcations (branches in the ridges).

A general sequence of fingerprint processing for matching is the following. First the gray-scale fingerprint image undergoes pixel-level processing to reduce noise, enhance the ridges, and binarize [O'Gorman 1989]. Upon this binary image, the macroscopic pattern and singular points can be detected, usually by forming a direction map image and matching this with a pattern [Srinivasan 1992]. For example, a whorl pattern will have all perpendicular directions of the ridges pointing approximately in the direction of the core. Following this, the fingerprint is thinned, then chain coded to locate line endings and bifurcations. It is then described by its macroscopic pattern and a feature vector of minutia locations relative to the core and delta and/or relative to each other. This information is then used for matching. Figure 4 shows a fingerprint example (the same as originally shown in Section 2.4 on Thinning), and some of the features are pointed out here.

Printed music processing: A top-down approach can be used for recognizing printed piano music. Although all symbols in music are not handled, this system is aimed at recognizing complex music notations where symbols are drawn with high density. The system uses musical knowledge and constraints to overcome the difficulties involved in processing such complex notations, both at the pattern recognition and the semantic analysis stages. [Carter 1992] describes initial processing steps for segmenting musical symbols from word-underlay and decorative graphics in early seventeenth century madrigals notated in white mensural notation. Application of graph grammars for extracting information content in complex diagrams using music notation as an example has been described in [Fahmy and Blostein 1993].

Extraction of information from microfilm images of punched cards: The United States National Archives has a large collection of microfilm reels containing information about World War II enlisted personnel in the form of images of punched cards. Unfortunately, original punched

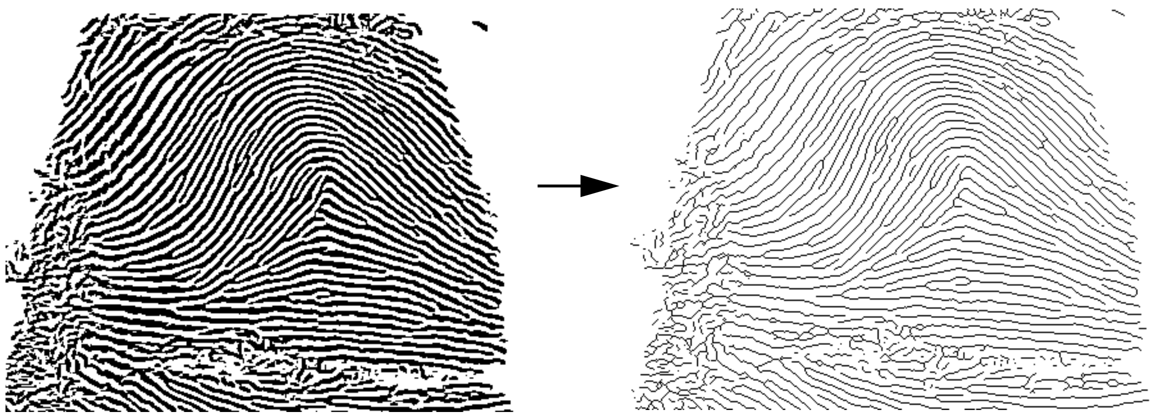


Figure 4. The fingerprint image on the left is processed and thinned to yield the result on the right. The macroscopic pattern of this print is an arch. The core is at the apex of the ridge lines to the right center. The delta is to the lower left of the core, where a triangular meeting of patterns occurs. One can also locate many bifurcation and line ending minutia.

cards or other forms of computer readable data are available. It is impractical to re-key all this information although the information stranded in these reels is not usable until it is converted to a computer readable form. A document image analysis system which includes semantic analysis steps has been described in [Kumar and Kasturi 92] to convert these microfilm images.

We recommend four papers describing applications mentioned in this section: [Vaxivière 92] and [Joseph 92] on mechanical drawing recognition, [Boatto 92] on map recognition, and [Carter 92] on musical score recognition.

Other Figure and Table Captions

Figure 1. A test image containing a flow chart, to illustrate graphics segmentation and recognition.

Figure 2. Figure explains assignment of priorities for determining closed loops in diagram (and is explained in detail in the text).

Table 1. Table shows library of known shapes for loop detection and recognition on diagram of Figure 1.

Figure 3. A test image containing different shapes, with results of graphics recognition shown in Tables 2 and 3 [Kasturi et al. 90].

Table 2. Table shows recognized objects in diagram of Figure 3.

Table 3. Table shows recognized lines and their attributes of Figure 3.

References

1. Kasturi, R., S.T. Bow, W. El-Masri, J. Shah, J. Gattiker, and U. Mokate, "A system for interpretation of line drawings," IEEE Trans. PAMI, 12(10):978-992, October 1990
2. R. Kasturi, R. Fernandez, M. L. Amlani, W. C. Feng, "Map data processing in geographical information systems", Computer, 22(12): 10-21, 1989.
3. Chandran, S. and R. Kasturi, "Interpretation of tables," Proc. Second International Conference on Document Analysis and Recognition, Tsukuba Science City, Japan, 1993
4. Antoine, D., S. Collin and K. Tombre, "Analysis of technical documents: The REDRAW system", in *Structured Document Image Analysis*, H.S. Baird, H. Bunke and K. Yamamoto (Eds.), Springer-Verlag, pp. 385-402, 1992
5. P. Vaxivière, K. Tombre, "Celesstin: CAD conversion of mechanical drawings", IEEE Computer, July 1992, Vol. 25, No. 7, pp. 46-54.
6. Joseph, S.H., and T.P. Pridmore, "Knowledge-directed interpretation of mechanical engineering drawings," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-14(9):928-940, 1992
7. D. Dori, "Vector-based arc segmentation in the machine drawing understanding system environment," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 17, No. 11, Nov. 1995, pp. 1057-1068.
8. Lai, C.P. and R. Kasturi, "Detection of dimension sets in Engineering Drawings," IEEE Trans. Pattern Intelligence and Machine Intelligence, Vol. 16, No. 8, 1994, pp. 848-455.
9. Arias, J.F., C.P. Lai, S. Chandran, and R. Kasturi, "Interpretation of telephone system manhole drawings," Proc. Second International Conference on Document Analysis and Recognition,

Tsukuba Science City, Japan, 1993

10. Lysak, D.B. and R. Kasturi, "Interpretation of line drawings with multiple views," Proc. 10th ICPR (Computer Vision Sub-conference), pp. 220-222, 1990

11. Ejiri, M., S. Kakumoto, T. Miyatake, S. Shimada, and K. Iwamura, "Automatic recognition of engineering drawings and maps", in *Image Analysis Applications*, R. Kasturi and M.M. Trivedi (eds), Marcel Dekker, 1990

12. Boatto, L., V. Consorti, M. Del Buono, S. Di Zenzo, V. Eramo, A. Esposito, F. Melcarne, M. Meucci, A. Morelli, M. Mosciatti, S. Scarci, and M. Tucci, "An interpretation system for land register maps," Computer, 25(7):25-33, 1992

13. Fahn, C.S., J.F. Wang, and J.Y.Lee, "A topology-based component extractor for understanding electronic circuit diagrams", Computer Vision, Graphics and Image Processing, 44:119-138, 1988

14. Lee, S.W., "Recognizing hand-drawn electrical circuit symbols", in *Structured Document Image Analysis,* H.S. Baird, H. Bunke and K. Yamamoto (Eds.), Springer-Verlag, pp. 340-358, 1992

15. O'Gorman, L. and J.V. Nickerson, "An approach to fingerprint filter design," Pattern Recognition, 22(1):29-38, 1989

16. Srinivasan, V.S. and N.N. Murthy, "Detection of singular points in fingerprint images," Pattern recognition, 25(2):139-153, 1992

17. Carter, N.P., "Segmentation and preliminary recognition of madrigals notated in white mensural notation," Machine Vision and Applications, 5(3):223-229, 1992

18. Fahmy, H. and D. Blostein, "A graph grammar programming style for recognition of music notation," Machine Vision and Applications, 6(2): 83-99, 1993

19. Kumar, S.U. and R. Kasturi, "Text data extraction from microfilm images of punched cards," Proc. 11th International Conference on Pattern Recognition, The Hague, Netherlands, pp. 230-233, September 1992

5.4: Commercial State and Future Trends

In the mid-1980's there was much hope for commercial systems to automatically convert the vast archives of engineering drawings from paper to electronic form. Where these products simply binarized, compressed and stored the data, they have been successful. However, where they promised component recognition, they have been less so. The realization has come about from this experience that the promise of completely automatic products was premature. Today's products for graphics recognition depend on human interaction to make decisions where the machine cannot, and the well-designed system is one that makes use of human aid most easily and quickly.

Though completely automatic graphics recognition is not in the near future, there have been commercial successes in this field. We list some of these here as well as suggestions on where progress will continue and where it is especially needed:

• For the archives of paper-based engineering drawings, products that simply digitize and compress — that is, perform no recognition — are a popular choice. Slightly more ambitious are products that employ some simple analytical methods such as thinning and contour following to store drawings in a more compact form. The most ambitious products, ones that perform some recognition, are now realized to require human interaction, as mentioned above. The up-front costs involved in interactively digitizing are deemed to be worthwhile for applications where a drawing will subsequently undergo much updating or when it must be merged into more recent drawings done in electronic (CAD) format. Future work in this area includes: faster hardware for digitizing and vectorizing, better noise reduction techniques for separating components, fast matching and indexing techniques for recognizing components, and better human interaction tools for checking and correcting computer results.

• Computer graphics packages for drawing on personal computers and pen-based computers often contain some recognition techniques. These translate rough, hand-drawn geometric shapes into straight lines, perfect circles, and spline-based smooth curves. In the future, these will recognize more complex shapes.

• Graphics libraries currently contain symbols for particular applications (e.g., electronic components for electrical drawings and mechanical components for mechanical drawings). As recognition becomes more reliable, more general-purpose machines will be built similar to multi-font OCR machines. These will recognize the character set(s) from symbols in the image, and perform component recognition accordingly. Also, graphics recognition will be merged with OCR for increased recognition of graphics components such as company logos, lines in tables and forms, and diagrams.

• Besides planar diagrams, more complex drawings include shading, and renditions of 3-dimensional shapes. These are the characteristics of computer vision images, that is, 2-dimensional images of 3-dimensional objects, and this suggests that as graphics recognition deals with these types of drawings, the distinction between the two areas will become blurred.