Computer Processing of Line-Drawing Images

HERBERT FREEMAN*

This paper describes various forms of line drawing representation, compares different schemes of quantization, and reviews the manner in which a line drawing can be extracted from a tracing or a photographic image. The subjective aspects of a line drawing are examined. Different encoding schemes are compared, with emphasis on the so-called chain code which is convenient for highly irregular line drawings. The properties of chain-coded line drawings are derived, and algorithms are developed for analyzing line drawings to determine various geometric features. Procedures are described for rotating, expanding, and smoothing line structures, and for establishing the degree of similarity between two contours by a correlation technique Three applications are described in detail automatic assembly of jigsaw puzzles, map matching, and optimum two-dimensional template layout.

Key words and phrases. image processing, line-drawing processing, computer graphics, line structure encoding, pattern recognition, map matching optimum two-dimensional layout.

CR categories: 3 10; 3.20, 3.63, 8.2

1. INTRODUCTION

A line drawing is one of man's most common and effective means of communication, and its processing by computer has attracted the attention of computer engineers for more than a decade [26, 57, 71, 81, 129, 143, 145]. In a simple sense, a line drawing is a picture that conveys to its viewer information through the shape, size, and manner of interconnection of thin lines on a contrasting background. The thickness of the lines, their color, or the color or texture of the background are either of no or at most of symbolic significance. We find line drawings in highway maps, engineering drawings, weather maps, temperature charts, graphs of statistical data, and alphanumeric text. Not only are there many graphical presentations which are directly recognizable as line drawings, but also many photographs and halftones carry the information of interest in what is essentially a lineal rather than a textual form. In this paper we shall be concerned with the computer processing of line drawings. However, it is important to keep in mind that a line drawing is a medium of communication, and that in speaking of "processing a line drawing" we are in fact referring to the processing of the information conveyed by the line drawing [87].

Before proceeding further it is necessary to define three terms of which we shall make frequent use—"image," "line drawing," and "line structure." An *image* is a natural visual object that is characterized by the two-dimensional spatial variation of brightness or color or both. It is objectively sensed by the normal human eye, and does not depend on the assignment of meaning for its definition. Every photograph, every painting, every

* Center for Interdisciplinary Programs, New York University West 177th Street & Harlem River, Bronx, New York 10453.

The author's work described here was supported in part by the Directorate of Mathematical and Information Sciences, Air Force Office of Scientific Research, under grant AF-AFOSR-70-1854 and in part by the Information Systems Branch, Office of Naval Research, under contract N00014-67-A-0467-0010.

CONTENTS

printed page—these are all examples of images.

A *line structure* is a geometrically defined concept, consisting of an assembly of points, line segments, and curve segments in Euclidean space. The assembly need not be connected. Line structures can be defined precisely and unambiguously in an appropriately selected coordinate system. For example, we may describe a particular two-dimensional line structure in a right-handed Cartesian coordinate system by the set of statements:

line segment from (3, 2) to (6, 2)
line segment from (6, 2) to (6, 6)
line segment from (6, 6) to (3, 6)
line segment from (3, 6) to (3, 2)
circle of radius 1.5, centered at (6, 2)

where (x, y) is the usual geometric notation for a point. If desired, this line structure can be represented visually by an image, as shown in Figure 1. The image then serves as a physical model for an abstract concept, which it represents to a certain limited approximation.

The use of an image to convey information is, as in all forms of modelling, a highly subjective process, dependent on the observer, his past experience, the time, the place, and the context in which the information is to be conveyed [77]. We shall use the term *line drawing* to denote an image used to convey information about a two-dimensional (2D) line structure. Note that the line structure must only be perceived for an image to convert into a line drawing—it need not be

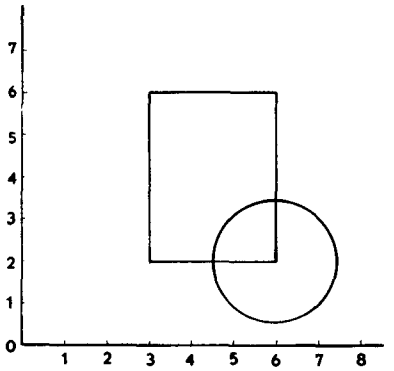


FIG 1. A line structure modelled by an image

recognized (i.e., its meaning need not be identified). Thus there is little difficulty in perceiving a 2D line structure in Figure 2. The question as to whether the particular 2D line structure represents a valid isometric projection of a three-dimensional object is irrelevant for establishing Figure 2 as a line drawing [69].

Any line drawing is first of all an image—it can become a line drawing in the mind of the beholder if he so perceives it. Thus a motorist looking for the best route between two cities will regard a highway map as a line drawing; but the map is merely an image when examined for its value as a decorative item. Similarly, a newspaper page is fundamentally an image; when viewed by a literate person, it may be subjectively converted into a line drawing, conveying information via the shape of the printed characters. From the point of view of conveying purely graphical information (i.e., the 2D line structure), it is sufficient that the characters be abstracted (perceived) as distinct graphical objects represented by shape and size, and, possibly, by the relative positioning of multiple parts; it is not necessary that the characters be "recognized" and a meaning assigned to them. Nor is it required that the observer understand the message carried by the characters. The message can be "in code" or merely gibberish. The three symbols of Figure 3a are all readily perceived as line drawings. Similarly, the image of Figure 3b is easily perceived as a line drawing, conveying some (as yet unspecified) information via the boundaries between the black and white areas. No message is apparent. If the page is turned top to bottom, the reader with some imagination may recognize a basketball game in progress.*

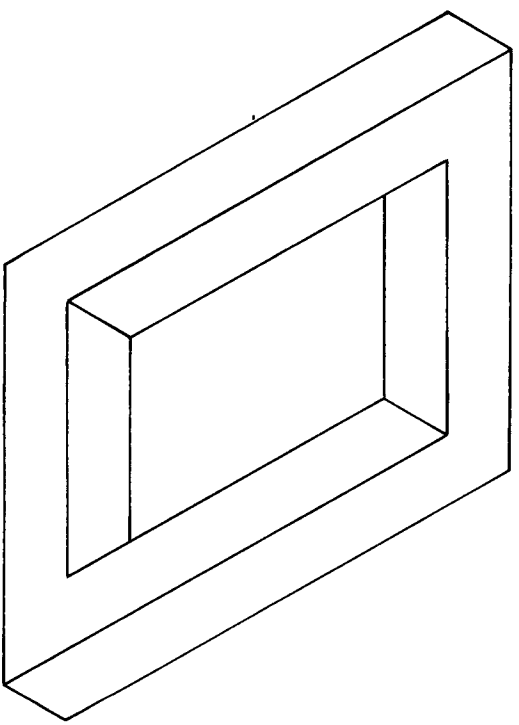
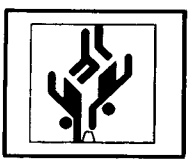


FIG 2 An image perceivable as a line drawing

(a)



(b)

FIG. 3. Two other images readily perceived as line drawings

As with any communication medium, actual transmittal of information via line drawings can occur only when there is an agreed-upon convention between the person "transmitting" and the person "receiving." This communication convention, commonly referred to as the communication language, is as essential an ingredient of line-drawing communication as it is of ordinary person-to-person conversation. Expressed in another way, we can regard a 2D line structure as a "statement" in a particular visual communication language [22, 32, 77, 89]. There can be many different 2D line-structure languages, and each can have its own set of dialects. (An example of this is provided by the different

* The basketball symbol for the 1972 Olympics.

sets of traffic signs used around the world.) Awareness of the inherent linguistic nature of line structures is prerequisite to an understanding of their processing by means of a digital computer [35, 36, 81, 95].

The phrase "processing of line drawings" is at times used to refer to two very different operations. One is the extracting of 2D line structures from line-drawing images, and the other is the processing of line structures to extract information of interest.* The first of these operations is largely context-independent and is commonly referred to as preprocessing. We shall be concerned almost entirely with the second—the processing of line structures after they have been extracted from their source media.

We have here seemingly placed emphasis on the processing of 2D line structures. They are the most common; however, 3D line structures are also important, especially for describing objects in the physical world (e.g., architectural structures). Although we are able to process 3D line structures (and even those of higher-order dimensions), only 2D line structures can be *rendered* uniquely via line drawings. To render 3D line structures, we must generate 2D line-structure projections, and such rendering is invariably fraught with ambiguities. Of course, for a 3D line structure there is still the possibility of building a physical wire-frame model. Such a model can display a 3D structure in a manner analogous to that in which a line drawing displays a 2D structure.

A line structure can originate in one of three ways, and the nature of its origin significantly affects the way in which it can be processed. First, a line structure may be abstractly specified in a geometric sense, as was done here for the rectangle and circle subsequently displayed by means of the line drawing of Figure 1. This is the case where the line drawing serves as a model for the line structure.

A second source of line structures is what we shall call *tracing*. The path of a draftsman's hand in tracing a figure in a photograph, the course of a ship on the high seas, or the track of an ant crawling over a table top can all be directly represented in the form of line structures. Tracing in a plane leads to 2D line structures, and tracing in 3-space leads to 3D line structures. Tracings are derived from the physical world, and, if planar, can be represented by line drawings.

Images provide the third source of 2D line structures. In this case, the line structure serves as a model for the information conveyed by the image, rather than conversely. (It is the line structure that is now the approximation!) Commonly, the line structure is used to represent equi-density contours (isophotes) [67, 118, 127], medial-axes functions ("skeletons") of constant-density "blobs" [9, 90, 102] or boundary lines separating portions of different density, color, or texture in an image [48, 56, 86, 118, 125, 126].

It is instructive to group line-structure processing problems into four categories on the basis of the input and output of the operation, as illustrated in Figure 4. In *analysis*, a line structure is given and its characteristics are to be determined. If the structure is that of a highway map, a typical problem might be to determine the road with the fewest number of intersections between two cities. If it is that of an engineering drawing of a machine part, the problem may be to find the location of the centroid of the part. *Synthesis* is concerned with the generation of line structures having certain specific characteristics, and then with their display by means of labelled line drawings. Engineering and architectural drawings are among the most common examples of synthesis. *Manipulation* encompasses those problems in which line structures are subjected to transformations. An example is provided by the generation of Mercator map projections from Earth satellite photographs. *Pattern recognition* has as its objective the classification of line structures. It has much in common with analysis; however, the determination of characteristics (features) is merely a means toward establishing membership in classes rather than the prime objective. Examples are: recognizing terrain features in aerial photographs, machine reading of printed characters, and the automatic classification of fingerprints.

* There is a possible third operation—the one engaged in by workers in a paper recycling plant, who may also speak of "processing line drawings."

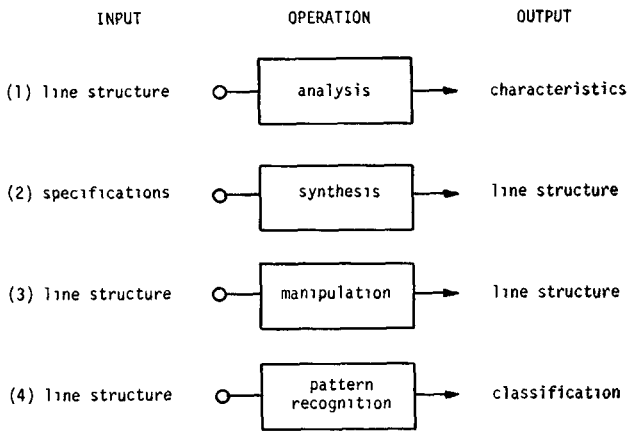


FIG 4. The four basic types of line-structure processing problems

2. DESCRIPTION OF LINE STRUCTURES

As a first step in the computer processing of line drawings, we must find a way to describe the line drawings in the language of the computer. A computer line-drawing description —as distinguished from a computer image description—will necessarily be a line structure, expressed precisely and unambiguously in terms of a (not-necessarily connected) set of straight-line segments, arcs of known curves, and predefined standard symbols.* By imposing restrictions on the admissible line segments, arcs, and symbols, we obtain a variety of schemes for line structure representation [58, 77, 89]. The selection of a particular scheme is governed in part by the competing objectives of efficiency of storage (economical use of computer memory) and efficiency of processing (economical use of computer time), and depends very much on the application—the amount of significant fine detail in the line structure, the source of the line-drawing data, the processing to be performed, and the display requirements.

If the line drawing to be processed is already described by a line structure, as in the case of the rectangle and circle modeled in Figure 1, only translation of the given description may be required. An appropriate translation routine (i.e, graphic-language compiler) in the computer will readily accomplish this. Many graphic languages for describing line structures have been developed in recent years, mostly for on-line use with graphic terminals [4, 12, 21, 23, 30, 76, 97, 128, 133, 137, 138, 149]. The various languages differ primarily in the applications for which they are intended, in the hardware with which they are to be used, and in their degree of formalism, elegance, and practical utility.

If the line structure is to be derived from a tracing or abstracted from an image, the source data must first be obtained by means of some physical measurements (an operation known as "sensing"). The process of converting the measurements into a data form acceptable to a computer is referred to as "encoding." Since encoding is a descriptive process and since there can be no description without quantization, we must first quantize the given tracings or images before we can "describe" them to the computer.

Quantization is inherent in any description process [20]. It requires that the whole (i.e., the total range of variability) be broken up into standardized elementary parts. The parts are "elementary" in the sense that in the particular context in which they are used, no further subdivision (i.e., finer detail) is of interest. The assigning of a name to the elementary parts then constitutes encoding. In general, the more precisely we wish to describe something, the more finely we must quantize it and the more extensive or complex the subsequent encoding.

* The standard symbols are, of course, also composed of straight-line segments or arcs of known curves. Their role here is analogous to that of *macros* in a computer program.

Line Structures Derived from Tracing

One source of line structures is the operation of tracing, the guiding of a point along a specified path. We shall suppose here that the point's position in 3-space is given by the continuous functions $x(t)$, $y(t)$, and $z(t)$, and by the binary-valued function $D(t)$, where t is a continuous, monotonically-increasing ordering parameter. $D(t) = 1$ indicates valid tracing data and $D(t) = 0$ indicates a path traced solely to preserve the spatial relationship between otherwise non-connected tracings. The line structure is the set of all tracings for which $D(t) = 1$. The reader will note that a tracing operation may be explicit, as when a draftsman traces the outline of a human figure in a photograph with a data tablet,* or it may be implicit, as when instruments record the course of a ship on the high seas. We shall initially limit consideration to tracing in a plane, for which, of course, $z(t) \equiv 0$.

Quantization for the purpose of encoding requires that the traced path be segmented. The segments in turn can then be approximated by standardized curves to which names can be assigned. Segmentation can be achieved by sampling the functions $x(t)$, $y(t)$, $z(t)$, and $D(t)$ at regularly spaced increments in t; however, this makes the size of the segments dependent on the tracing speed. Alternatively, segmentation can be performed whenever the change in x, y, or z from the last point of segmentation reaches a preset quantity.

Quantizing a tracing by breaking it up into segments raises some important questions. Should the segments be large and few in number, or should they be small and in greater number? If the former, the mathematical representation of the segments will be relatively complex (e.g., high-order polynomials may be required); if the latter, a straight-line approximation may be sufficient. Further, should the segments all be equal or should segmentation be variable and depend on some characteristic of the tracing itself? Uniform segmentation has the advantage of simplicity; however, segmentation based on some characteristics of the tracing may lead to more efficient encoding and processing [111].

For a quantization scheme to be useful, it should be simple, highly standardized, and universally applicable to all tracing. Further the scheme should be such that it leads to simple encoding (i.e., "naming") and facilitates digital computer processing. These objectives appear to be best met by straight-line segmentation using either a single length or a small number of standard lengths [39]. For special purposes, other approaches may have some advantages, as for example, segmentation at points of maximum curvature or at points of inflection [5, 152]. However, these more sophisticated approaches cannot match the simplicity and universal applicability of fixed straight-line segmentation [92, 105, 111].

A particular scheme of straight-line quantization which makes use of a square grid of arbitrary fineness to prevent the build-up of quantization error is the following. Consider a simple curve (to be traced) and superimpose a uniform, square grid. An x, y coordinate system can be identified with the grid so that every node can be described in terms of its discrete coordinates (mT, nT), where m and n are integers and T is the separation between adjacent grid lines. A quantized description suitable for entry into a digital computer can then be obtained by giving (in sequence) the coordinates of the grid nodes that lie closest to the given curve as the curve is traced out from end to end (i.e., as t varies from 0 to its maximum). For convenience we shall regard T as a scale factor and denote the coordinates of the nodes simply by (m, n).

Determining the grid nodes that lie "closest" to a tracing is, however, not as straightforward as it might appear. Various possibilities exist [13, 41]. In the most direct scheme, the curve is traced, and for each square

$$(m - \tfrac{1}{2})T < x \leq (m + \tfrac{1}{2})T$$

$$(n - \tfrac{1}{2})T < y \leq (n + \tfrac{1}{2})T$$

* A *data tablet* is a device with a stylus for manually tracing a figure. The position of the stylus is electrically sensed and then is made continually available, either directly or after some coordinate transformation, for entry into a computer. A large variety of such devices are commercially available. Their precision typically ranges from 1% to 0.1% of full scale.

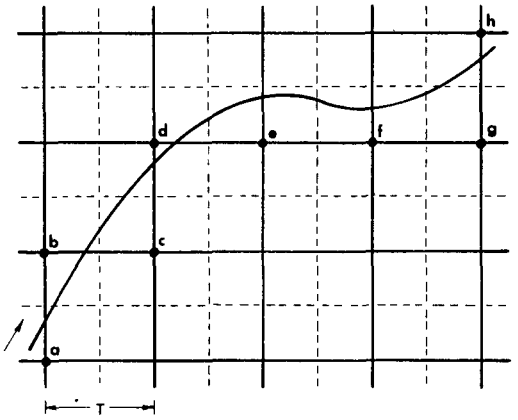


FIG 5. Square-box quantization of tracings

passed through by the tracing, the node (m, n) is selected. This scheme is known as *square-box quantization* and is illustrated in Figure 5 One observes that with this scheme, it is essentially impossible for two adjacent data nodes to be located along the diagonal of a grid square. (In Figure 5, the sequence of data nodes is *abcdefgh*.)

In another scheme, referred to as *grid-intersect quantization*, nodes are selected according to the following rule:

> Trace the curve from end to end, and whenever t and m are such that $x(t) - mT = 0$, select n such that $(n - \frac{1}{2})T < y(t) \leq (n + \frac{1}{2})T$; when t and n are such that $y(t) - nT = 0$, select m such that $(m - \frac{1}{2})T < x(t) \leq (m + \frac{1}{2})T$.

This scheme is illustrated in Figure 6, where the sequence of data nodes generated is *abcdef*, for the identical tracing as in Figure 5. Note that in contrast to square-box quantization, grid-intersect quantization does permit adjacent data nodes to lie at diagonally opposed corners of a grid square. For random tracings, the probability for this to occur is approximately 0.41 [41].

Additional trace quantization schemes have been devised that are variations of those already described; however, they offer no particular advantages, neither in ease of implementation nor in fidelity of representation [39, 41]. The grid-intersect method has the advantage of being relatively low in quantization noise because it permits diagonal adjacency of data nodes. It is an excellent scheme for initial quantization, from which, if desired, more coarsely quantized representations can later be generated [92, 111]. In all subsequent discussions we shall assume that grid-intersect quantization is used, unless a different scheme is expressly indicated.

If in Figures 5 and 6 the data nodes are connected in sequence by means of short line segments, straight-line approximations (i.e., quantized representations) of the traced paths will be obtained. The finer the grids, the more faithfully the straight-line approximations will represent the given tracings. For a particular-size grid, the grid-intersect scheme will yield a more faithful representation than the square-box scheme and will accomplish this with somewhat fewer segments [48], which will, however, be of two fixed lengths, T and $T\sqrt{2}$. This is readily seen from Figures 5 and 6, where the identical tracing has been quantized using the two schemes.

The type of straight-line approximation resulting from grid-intersect quantization of a tracing is also precisely the one which applies when a curve is drawn by any of the common types of "digital" plotters. These plotters accept computer commands for moving the pen a small, fixed-size increment in either of two mutually perpendicular directions (positively or negatively), or in both directions simultaneously. The result of this is that all curves are plotted as polygonal approximations using only the lengths T and

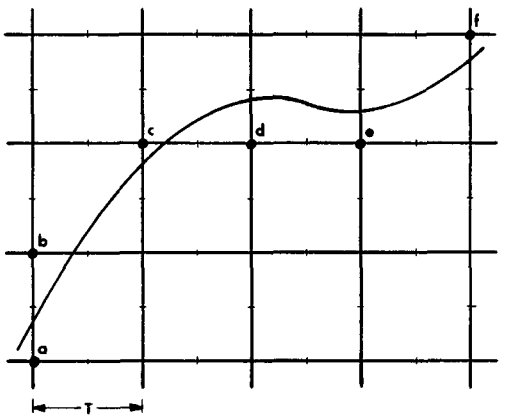


FIG 6. Grid-intersect quantization of tracings.

$T\sqrt{2}$ for the straight-line segments, where T now is the basic incremental motion of the plotter (typically 0.01 or 0.005 inch).

Quantization is a many-into-one mapping and is, of course, not reversible. The domains of all possible tracings that would yield the data-node sequences of Figure 5 (square-box quantization) and Figure 6 (grid-intersect quantization) are shown as the unshaded areas in Figure 7 (a) and (b), respectively. (Note that the tracings may not cross any of the dark barrier lines in Figure 7(b)). Although it is not possible to recover a given tracing from a data-node sequence, it is possible to define a unique smooth curve for a particular data-node sequence. A curve that is intuitively appealing for this purpose is the so-called *minimum-energy curve* described by the medial axis of a bent, thin, elastic beam that satisfies the constraints of the node sequence with minimum strain energy. This curve can be regarded as the "smoothest possible" curve that is consistent with the node sequence. Unfortunately, this curve is time-consuming to compute; however, its properties can be used to establish a criterion for the required quantization fineness [46, 51, 91].

Line-structure quantization is also encountered when the description of a line structure is changed from one form to another, less precise one. This occurs, for example, when a circle (or any other analytically defined plane curve) is to be represented

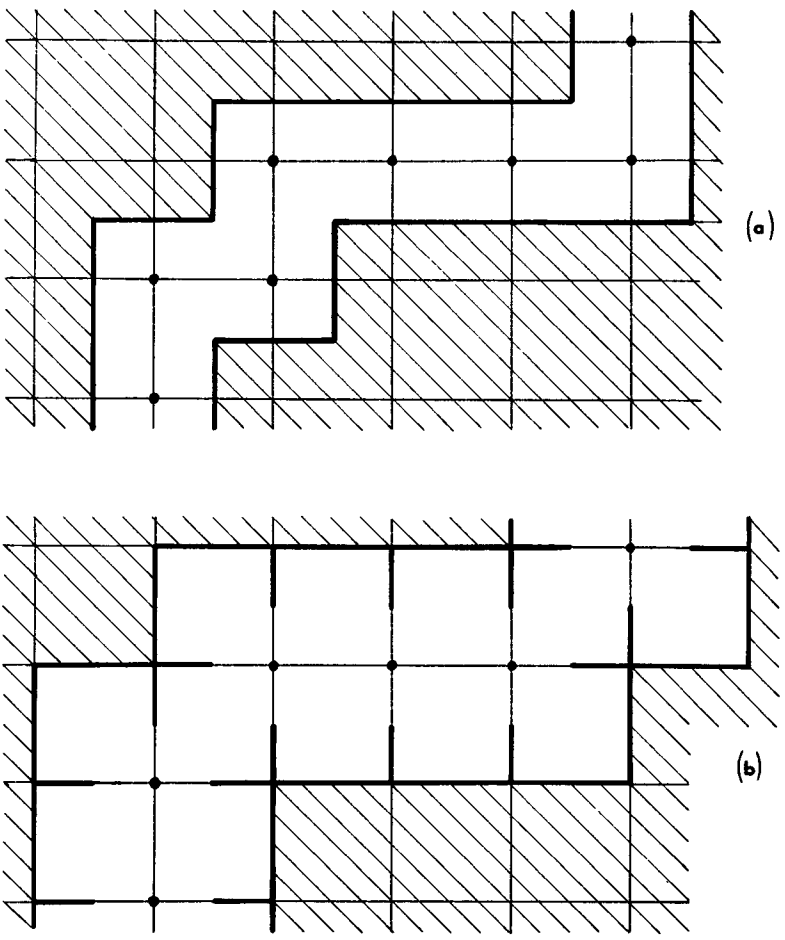


FIG. 7. Quantization domains for tracings for Fig's 5 and 6. (a) Square-box quantization, (b) Grid-intersect quantization.

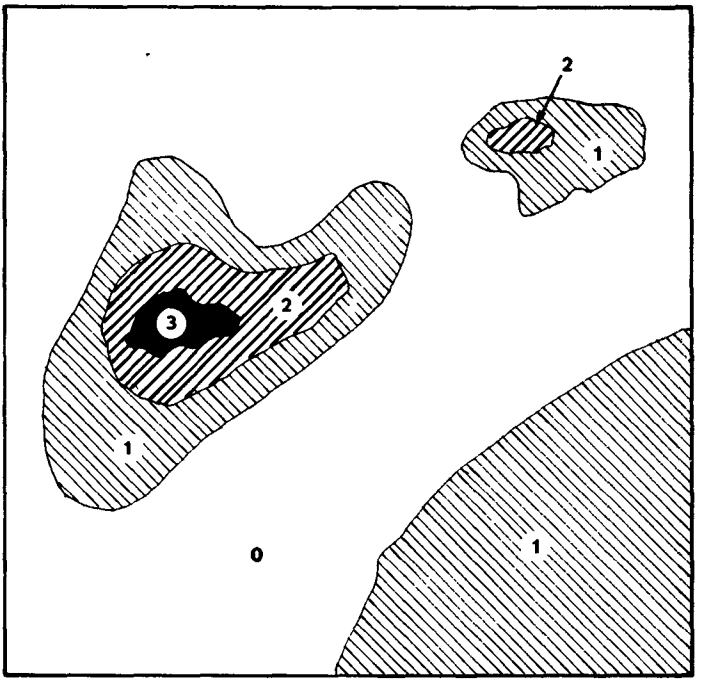


FIG. 8. Sample image, quantized to 4 levels of brightness

by an approximating polygon [101, 105], or when a polygon with a large number of edges is to be approximated by one with a smaller number of edges [70, 92].

Line Structures Derived From Images

One of the most challenging problems of computer image processing is that involving the extraction of line structures from images [67, 71, 117, 126, 130]. Here as in the case of tracings, the first step is quantization, and this is customarily achieved by dividing the two-dimensional image space into an $m \times n$ array of uniformly sized squares (image elements) and then assigning to each square of coordinates (i, j), $i = 1, \cdots, m$ and $j = 1, \cdots, n$, a k-bit binary number a_{ij} corresponding to the average brightness (quantized to 2^k levels) over the square. The array $[a_{ij}]$ then becomes the quantized representation of the image—which is said to be quantized $m \times n \times 2^k$—and it is this array that is actually then processed in the computer. Figure 8 shows an image in which the brightness is quantized to 4 levels. The corresponding computer array, quantized 28 × 29 × 4,

```
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000000000000
0000000000000000000011111100
0000000000000000000122211100
0000001000000000000011111100
0000011110000001000000110000
0000011111000001110000010000000
0000011111000111100000000000
0000112221111211100000000000
0000122222222221000000000000
0001223332222110000000000000
0001233333222110000000000000
0011223222211100000000000011
0011122221110000000000001111
0011112211100000000000111111
0011111110000000000001111111
0011111100000000000011111111
0011111000000000000111111111
0001110000000000011111111111
0000000000000000111111111111
0000000000000000111111111111
0000000000000001111111111111
0000000000000001111111111111
0000000000000011111111111111
0000000000000011111111111111
```

FIG 9. Computer representation of image of Fig. 8, quantized 28 × 29 × 4.

is shown in Figure 9. The reader should note that the intermediate grey levels between white (0) and black (3) perceived by the eye

in Figure 8 are achieved by closely-spaced black lines of different thickness rather than as uniformly grey areas. This is a variation of the so-called halftone technique of rendering grey levels by means of black-ink printing [74].

The array of Figure 9 is merely a two-dimentional list of the 4-level grey values of the image of Figure 8 quantized into 28 × 29 squares. The array should not be regarded as a quantized *graphical* rendition of Figure 8, both because of the spatial distortion (the horizontal/vertical spacing ratio is 55/30 instead of 1/1) and because the grey-level appearance of the numerals bears no relation to their value (the 0 appears darker than the 1, for example). A graphical presentation of Figure 8 quantized spatially as well as in brightness is shown in Figure 10; the square in the lower left-hand corner indicates the spatial quantum.

Although it is difficult to conceive of reasons for employing anything other than uniform spatial quantization, there may be definite advantages to having the brightness quantization vary in some nonlinear manner. The nature as well as the total range of the brightness variation (i.e., the set of admissible values a_{ij}) can be adjusted to optimize the probability of extracting the information of interest from the image [80, 118].

In contrast with a quantized tracing, in which the desired line structure is always explicitly available, a quantized image will yield a desired line structure often only after extensive processing. The processing required when we merely seek a line-structure representation for a given image [48, 67, 117] will differ markedy from when we want to extract a particular type of line structure which, on the basis of a priori information, we have reason to believe is "present" in the image [27, 112]. The difference between these two situatons is one of approach, not of ultimate objective. If we have a priori information about the line structure described by the given image, this information can—and should—be utilized for extracting the line structure. For example, if it is known that a given image is expected to contain a visible-edge projection of a polyhedral object, the line structure will consist of connected, relatively long, straight-line segments which must obey the topologic and geometric constraints implicit in such a projection [11, 61, 82]. The line segments are likely to be well-defined and indicated by substantial discontinuities in brightness. Small "blobs" in otherwise uniform areas or short gaps in long runs of straight boundary sections can be regarded as noise.

Once it is possible to define certain variations in the given data as noise, one can attempt to remove these interfering signals and enhance the desired data by means of suitable computer algorithms. Such filtering may be relatively simple and context-independent, as in most instances of pre-processing [26, 118], or it may be very elaborate and sophisticated, as in the so-called scene analysis problems [14, 27]. For a detailed discussion the reader is referred to the extensive literature available on this subject [115].

The array shown in Figure 9 represents a quantized, single-valued function of two discrete variables (the spatial coordinates). A function of this type can be conveniently represented by a contour map, in which the contour lines serve as boundaries between areas of constant value. A contour map for the quantized image of Figure 10 is shown in Figure 11. The contour lines are assigned a value and a direction. The value is the higher of the two values separated by the boundary, and the direction indicates that the area of higher value lies to the right. Contour lines may not cross each other (though they may overlap), a line drawn from a contour line of one value to a contour line of another value must intersect all contour lines of intermediate value an odd number of times, and all contour lines must be either closed curves or must close through the exterior boundary of the image [44, 93]. (The contour line in the lower right-hand corner of Figure 11 is closed through the exterior boundary in the lower right.) A contour-map representation for an image is, in general, more compact and (at least for some applications) intuitively more descriptive than the corresponding array representation [56, 68, 70, 118, 127].

In examining the contour map of Figure 11, one drawback that is at once noted is the

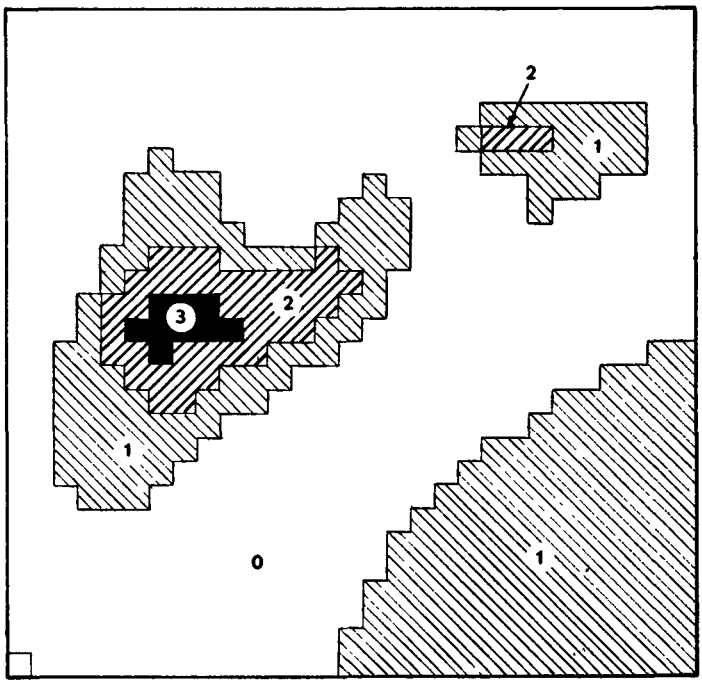


FIG 10 Image of Fig 8, quantized 28 × 29 × 4.

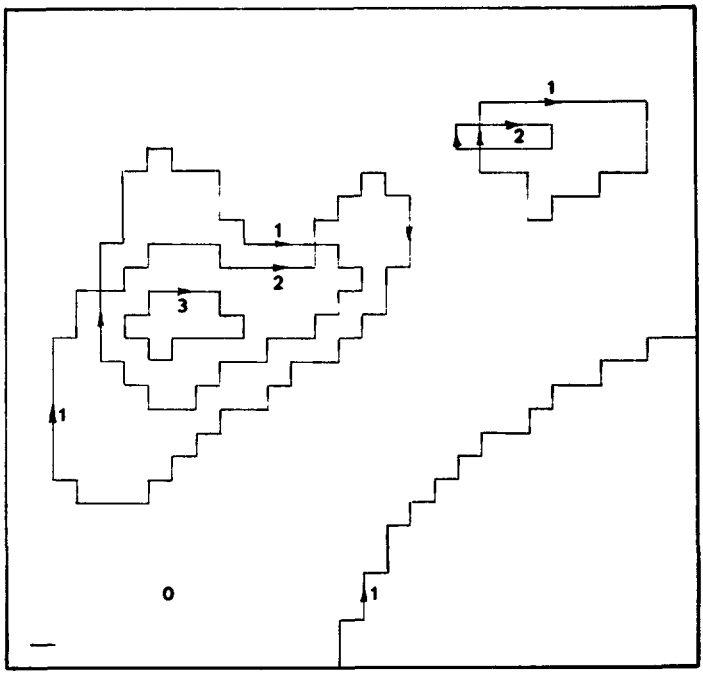


FIG. 11 Map of the boundary lines for the equal-brightness areas of Fig. 10.

presence of a strong component of quantization noise (the "staircase effect"). To a considerable extent this is caused by the restriction that the line segments making up the contour lines must necessarily lie along the grid lines of the original array and cannot cross a grid square diagonally. (This is the same characteristic also considered objectionable in the case of tracings quantized with the square-box scheme illustrated in Figure 5.) Smoother contour lines, giving a more faithful rendition of the shape of the equal-brightness areas, can be obtained by connecting the centers of adjacent upper-value boundary elements. An upper-value boundary element is defined as an element a_{ij} that has at least one side in common with an element of lower value. Upper-value boundary elements are said to be adjacent to each other if they have at least one node in common. Expressed more formally, the element a_{ij} is an upper-value boundary element if there exists an element a_{rs} (called a lower-value boundary element) such that

$$a_{ij} - a_{rs} = q$$

where

$$|i - r| + |j - s| = 1$$

and q is the brightness quantum (taken as unity in Figure 8). Two upper-value boundary elements a_{ij} and a_{nm} are adjacent if for $|i - m|$ and $|j - n|$ each equal to either 0 or 1,

$$|i - m| + |j - n| \geq 1$$

Two lower-value boundary elements a_{pq} and a_{rs} are adjacent if

$$|p - r| + |q - s| = 1$$

Boundary lines formed in this way may contain both segments that parallel grid lines, and segments that cut diagonally across grid lines. Adoption of these two forms of adjacency insures that the boundary lines divide the two-dimensional space into disjoint areas [92, 119].

When the foregoing scheme is applied to the array of Figure 8, the contour map of Figure 12 is obtained. Comparison with the boundary map of Figure 11, as well as with the original image of Figure 8, shows the new contour map to be of improved smoothness and fidelity. It should be noted, however, that the contour lines of Figure 12 enclose slightly less area than the corresponding geometric boundaries of Figure 11 because they are always wholly inscribed within the latter. Except for this area-shrinkage effect, the contour lines of Figure 12 are similar (though, of course, not identical) to the contour lines which would have been obtained had the boundary lines in Figure 8 been traced and quantized in accordance with the grid-intersect quantization scheme illustrated in Figure 6.

The Chain Coding Scheme

The line structures that result from quantizing tracings with the grid-intersect scheme or from connecting the centers of adjacent upper-level boundary elements in an image array are characterized by the property that each data node in sequence coincides with one of the eight grid nodes that surround the previous data node. If we label these eight neighboring grid nodes from 0 to 7 in a counterclockwise sense starting from the positive x axis, we can represent the line structures simply by sequences of octal digits. This is illustrated in Figure 13, where this coding scheme, shown separately in the lower right, has been applied to a section of Figure 12. Thus, for example, the contour line for level 3 in Figure 12 (marked "C" in Figure 13) is encoded as 31007445, starting from the node marked × and proceeding clockwise. Each octal digit corresponds to one directed straight-line segment. We shall refer to these straight-line segments (or the integers with which they are labelled) as *links*, and a sequence of links representing a line structure will be called a *chain* [39, 40]. This is stated more precisely as follows:

1) A *link* a_i is a directed straight-line segment of length* $T(\sqrt{2})^p$ and of angle $a_i \times 45°$ referenced to the x axis of a right-handed Cartesian coordinate system, where a_i may be any integer 0 through 7, and p is the modulo-2 value of a_i; that is, $p = 0$ if a_i is even and $p = 1$ if a_i is odd.

2) A *chain* is an ordered sequence of

* As stated previously, the grid spacing T is normally set to unity.

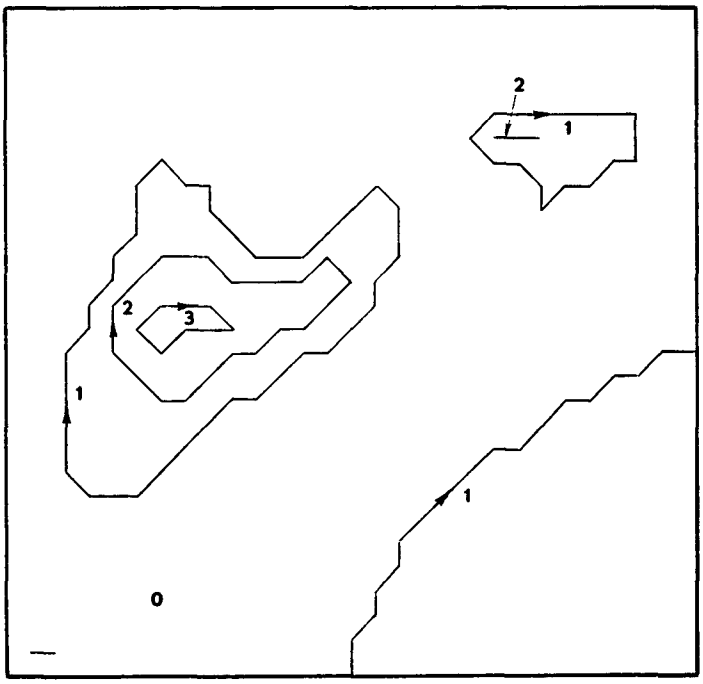


FIG. 12 Brightness contour map for image of Fig 8, quantized 28 × 29 × 4, and represented in chain form

links with possible interspersed signal codes. It is written in the form

$$A = a_1 a_2 \cdots a_n$$

or

$$A = \overset{n}{\underset{i=1}{C}} a_i \qquad (1)$$

where the latter expression is read "the chain a_i from 1 to n." Signal codes are explicitly identified. The *initium* of a chain is the node from which a_1 departs, and the *terminus* is the node to which a_n is directed.

3) The links a_i and a_j form an *inverse pair* if

$$a_j = a_i \dot{+} 4 = a_i^{-1} \qquad (2)$$

where the dot over the plus sign indicates modulo-8 addition. Clearly,

$$(a_i^{-1})^{-1} = a_i. \qquad (3)$$

4) The octal digit sequence $04d_1d_2$ is reserved as a *signal code* indication and does not denote a pair of links unless $d_1d_2 = 04$, in which case the four-digit sequence denotes a single pair of inverse links, 04. A certain number of octal digits following an $04d_1d_2$ sequence belong to the signal code and do not denote links, as specified by the signal code identified by the d_1d_2 combination. A list of signal codes is given in Table I.

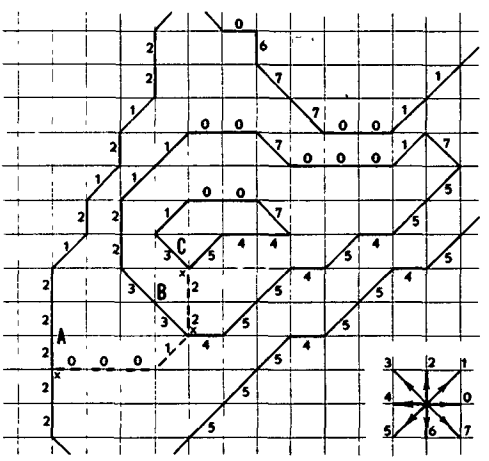


FIG. 13 Chain coding scheme (lower right), and some chain-coded boundary lines from Fig 12

The chain-code representation is invariant with translation over the grid. If desired,

TABLE I. CHAIN SIGNAL CODES

Code	*Description*
0400	Indicates end of chain.
0401	Invisible chain follows. (If chain is being plotted, this code causes pen to be lifted off paper.)
0402	Visible chain follows. (Negates code 0402.)
0403	In fixed-word-length machines, this code indicates that remainder of word is to be skipped.
0404	Valid 04 combination. Chain is to contain the digits 04 as valid data; the second set of digits 04 is ignored.
0405XYZ	Marker number XYZ XYZ is a 3-digit octal number (000 to 777). Used for identifying particular points in a chain.
0407UV	Serial-number indicator Immediately following this code is a serial number (written as an octal number) which is UV digits in length with maximum of 77 digits. (Example: 04070512345 identifies serial number as 12345) This code may typically appear as the start code of a chain
0410WXYZ	Non-chain data (e g , comment) follows, of length WXYZ digits
0411	Non-chain data follows as a sequence of octal digits, until terminated by end-of-non-chain-data indicator Non-chain data may be of any length.
04127777	End of non-chain data. Caution If 0411 and 0412 are used, non-chain data itself may not contain code 04127777, as this would immediately terminate non-chain data However, all other 04 combinations in non-chain data are interpreted as being part of the non-chain data
0413UWXYZ	Node number WXYZ Indicates that chain connects with U other chains (containing a similarly numbered node) at this point If U = 0, number of connecting chains is unspecified
0414UVWXYZ	Rotation indicator. Indicates chain data that follows has been rotated (using a rotation algorithm) by an angle of U VWXYZ radians Negative angles are expressed as magnitudes subtracted from 2π.
0415TUVXYZS	Scale change indicator Indicates chain data that follows has been modified (using a scale-change algorithm) by an amount UVXYZ, in which S indicates the location of the radix point, measured from right to left. The digit T indicates the nature of the scale modification, as follows T = 0, + overall (x and y) T = 1, − overall (x and y) T = 2, + x component only T = 3, − x component only T = 4, + y component only T = 5, − y component only (Example 04153123453 indicates that the chain has been expanded in the x direction by a factor of 12.345 and that the direction of the x components has been reversed.)
0417UXYZ	Link-repeat code. The link U is to be repeated XYZ times
0420UN	Link-repeat code The link U is to be repeated as many times as specified by the $(N + 4)$-digit octal number that follows Code used for very long link repetitions—up to 8^{11} times
0421TUVWXYZ	Group-repeat code. The sequence consisting of the immediately following TUV digits (000 to 777) is to be repeated WXYZ times.
0422U	Color indicator For plotters with multiple color pens, code indicates that choice U (one out of eight possible colors) is to be used until a new such code is encountered.
0423WXYZ	For use in contour-map chains Chain that follows is of elevation value $WXYZ_8$
0424XYZ	For use in image-array boundary curves. Chain that follows is of grey-level value XYZ_8
0425U	Check code U plus modulo-8 sum to this point in chain from either beginning of chain or last check code should be zero. Modulo-8 sum is taken over all digits encountered, including those of signal codes
0426VWXYZ	Resets the absolute x coordinate to VWXYZ (00000 to 77777)
0427VWXYZ	Resets the absolute y coordinate to VWXYZ (00000 to 77777)

the position of a chain can be fixed by giving the absolute x, y coordinates of its initium. The signal codes 0426 and 0427 are provided for this purpose.

To assure that the various contour lines in a chain-encoded contour map will be properly positioned, one can either specify the absolute coordinates of each contour line's initium, or provide "invisible" connecting chains from one initium to the next. The latter scheme is illustrated in Figure 13, where the invisible chain 0001 (shown dashed) ties the outermost contour line A (at left) to the contour line B, which in turn is then tied to contour line C by means of the invisible chain 22. (The initium of each chain is marked with an $\times$.) Invisible connecting chains are preceded by the signal code 0401 and terminated with the signal code 0402, and then simply inserted between the chains they connect to form a new combined chain.

If the brightness variation from one contour line to the next is uniform, it need not be explicitly indicated each time. However, if the brightness value (grey level) must be identified, this can be done through use of signal code 0424.

Long runs of identical links can be represented in a compact form by means of the link-repeat codes 0417 and 0420. Code 0417 permits repetitions up to 777_8 times, and 0420 permits repetitions up to 8^{11} (decimal) times. For example, a sequence of 45_8 links of value 2 can be replaced by the coded form 04172045, which requires only eight octal digits instead of the former 45_8. The code 0421 serves a similar function for the repetition of groups of up to 777_8 arbitrarily-valued links as many as 7777_8 times. For example, to repeat the link group 000001 exactly 130 (i.e., 202_8) times, one would use the group sequence 0421 006 0202 000001. (The spaces are inserted solely for readability.) This sequence requires $17 \times 3 = 51$ bits, whereas the 130-fold repetition of 000001 would require $130 \times 6 \times 3 = 2340$ bits!

In addition to using special codes for long sequences of repeated links, the storage efficiency of chains can also be improved by utilizing the property that successive elements in a chain are not statistically independent (except in a two-dimensional random walk!). In the chain code the eight digits 0 through 7 are assigned to the eight possible directions leading from one data node to the next. If a data node's eight surrounding grid nodes are all equally probable locations for the next data node, no improvement in the coding scheme can be made. However, under the assumption that the quantization is sufficiently fine to preserve all detail of interest, the changes in direction from link to link will normally not exceed $\pm 45°$. Turns of ± 90 degrees will be infrequent, and turns in excess of 90°, most rare. This suggests a *chain-difference* coding scheme, in which the difference in angle between successive links is represented by a two-bit code for differences not exceeding $\pm 45°$, and by progressively longer codes for larger differences. In general the chain-difference code will require only slightly more than two-thirds of the storage required for the chain code [41].

The chain coding scheme can be easily extended to three-dimensional line structures quantized on a cubic lattice. For each data node, the links can now take any of three lengths—T, $T\sqrt{2}$, and $T\sqrt{3}$—and from each data node there are 26 possible directions to the next data node. A five-bit binary code is required if each possible direction is to be uniquely designated. A particular coding arrangement which has been found useful is illustrated in Figure 14 [123, 124, 151].

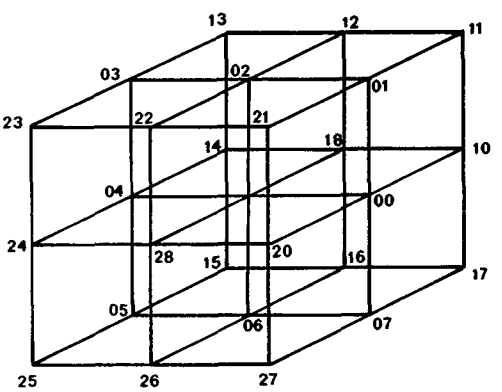


FIG 14 Chain-code extension to 3D line structures.

Filtering and Smoothing

When line structures are derived from tracings or from images, the initial forms of the line structures are likely to contain a considerable amount of noise. This noise may have been present in the original path or image, or it may have been introduced during the extraction process. Thus noise in a line structure obtained from tracing a path in a highway map with a manually guided tracing device may be due to either printing defects in the map itself or to jitter of the operator's hand in guiding the tracing stylus. Similarly, a line structure of the equi-brightness contours of an image is likely to contain many small contours which are due to imperfections in the photograph, non-uniformities in illumination (e.g., bright spots caused by glare) and dust particles. Since noise obscures the data of interest and also increases the volume of data to be handled when chain coding is employed, it is important that it be removed at the earliest possible stage.

Noise in a chain-encoded tracing will normally manifest itself in frequent and large differences between successive links. The noise effect will increase with a decrease of the grid spacing, that is, with an increase in the spatial resolution. There are two techniques for reducing noise. One can introduce smoothing into the tracing process itself by using an appropriately smoothed tracing mechanism. This will eliminate jitter and the tracking of insignificant irregularities. Alternatively, the noise can be filtered out in the computer subsequent to encoding. The latter approach requires the application of smoothing techniques which will remove the noise without also removing desired sharp corners. Such "nonlinear" filtering requires that we have available some a priori knowledge about the line structure and utilize heuristic procedures [14, 116, 130].

For line structures derived from images, there are three opportunities for removing noise. The first consists of filtering the original image by means of optical techniques [118, 140]. The second is to smooth the quantized grey-level image by means of array processing algorithms in a digital computer [26, 118, 126]. The third possibility is to filter the line structures after they have been extracted from the image [68, 116]. The line structure smoothing techniques would be the same as those applicable in the case of tracings, except that when line structures are obtained from images, not only must excessive irregularities be filtered out, but now there may also be extraneous as well as missing line segments that require correction.

General Polygonal Line Structures

A chain represents a line structure as a sequence of short, fixed-length, straight-line segments. In the hierarchy of pattern description languages [32, 72, 89, 95], the chain code is a primitive scheme for representing line structures, and, therefore, is ideal when the line structures are expected to be highly irregular or when there is little a priori knowledge about them [32]. This is usually the case when tracing a natural rather than a man-made path (e.g., the track of an animal searching for food as against the track of a ship on the high seas). The same applies for line structures extracted from images—those describing natural objects such as geographical contour maps, or the shapes of chromosomes tend to be irregular and unpredictable, whereas those describing man-made objects such as architectural structures or automobiles tend to contain predictable straight-line segments and curved arcs. Hence in many instances where a line structure representing some man-made object is initially represented as a chain—and perhaps subjected to some preprocessing (especially noise filtering) in this form—it is desirable to convert from the chain representation to a high-level polygonal or curved-arc representation. The line structure then becomes an assembly of a relatively small number of long straight-line segments or long arcs of mathematically defined curves (e.g., circles, parabolas). Although this is especially applicable to line structures describing man-made objects, natural objects are by no means excluded. For many natural objects, a chain-type line structure—though initially valuable for per-

mitting the identification of critical features —may be unnecessarily detailed for the ultimate data processing objectives.

It has long been known that information about shape is conveyed via the curvings of an object's boundary lines (i.e., the changes in direction of the tangent vector) and that the information content is greatest where the curvings are strongest [5, 143]. Conversely, small-magnitude, rapidly-reversing curvings can, in general, be dismissed as being noise.

One of the weaknesses of the chain code is that it does not discriminate between noise and significant curvings. In recent years a number of researchers have directed their attention to developing procedures for constructing high-level polygonal structures from chain-coded line structures [70, 92, 109, 111]. In one approach, as a chain is traversed, sequences of links describing small, rapidly reversing slope changes are replaced by long straight-line segments, and successive angular changes in the same direction are combined into a single large angular change. The result is a polygonal approximation, coarser than the chain from which it is derived, but one which highlights the significant information-conveying features of the contour [75, 126]. Such a scheme, by placing emphasis on significant shape features, can also be utilized for developing a classification system for certain families of line structures (such as those describing chromosomes, fingerprints, and type fonts) as well as for providing a basis for a syntactical description of shape [152]. A high-level polygonal approximation of the chain-encoded contour map of Figure 12 is shown in Figure 15.

Another scheme for constructing high-level line structures for chains consists of fitting straight lines to sequences of links, subject to some error criterion. A criterion used for this purpose has been the minimum mean square perpendicular distance from the chain to the approximating line, with the location of the end points and the line lengths subject to variation [70]. A third scheme utilizes a dynamic programming approach to find the polygons of minimum length, subject to the constraint that the polygon's vertices lie within a small distance (usually $T/2$) from a chain node [92].

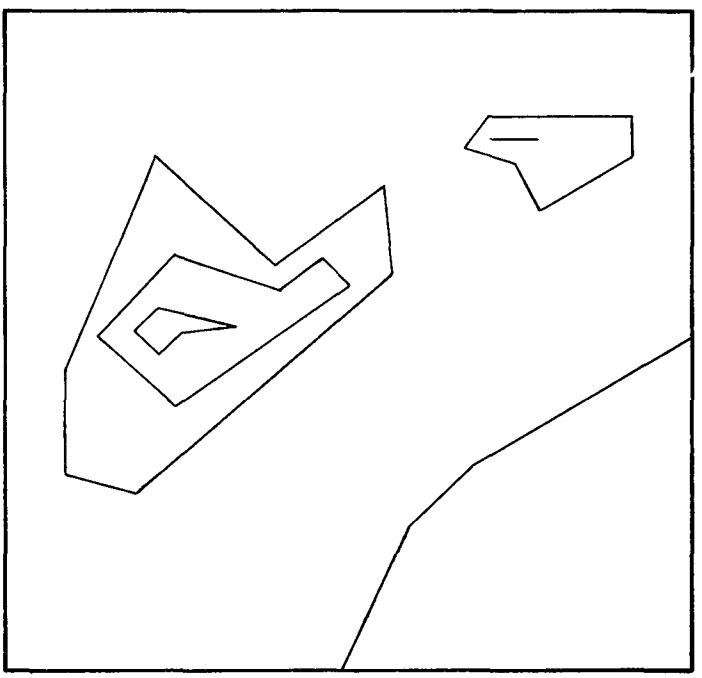


FIG 15 High-level polygonal approximation of contour map of Fig 12.

A review of line structure description methods would not be complete without mention of the so-called *skeleton* techniques [9, 78, 90, 118]. In these, a boundary line, instead of being chain-encoded, is progressively shrunk until the shrinking "wavefronts" from opposing sides meet in a medial line or point (the "skeleton"). If the distances from the boundary to points on the skeleton are recorded, the process is reversible; that is, a boundary can be reconstructed from its skeleton. The method has attracted considerable attention for classifying shapes of natural objects (e.g., chromosomes, cloud-cover photographs) but its applications to date have been limited [102, 122]. Processing of skeletons to extract features of interest or to transform them from one coordinate system to another is relatively awkward, and, except in certain pattern recognition applications, the chain code appears to be a more versatile and convenient line structure representation.

Linguistic Methods

Already during the earliest attempts at using digital computers for processing line structures, it was realized that line structures could be regarded as possessing a syntax, and that line structure descriptions could be parsed in a manner analogous to the parsing of computer-language statements [26, 66, 72, 95, 129, 145]. The approaches followed have been characterized by various degrees of formalism, but all represent adaptations of Chomsky's general concepts about natural languages to two-dimensional line structures (i.e., to so-called picture languages). The work has helped to provide considerable insight; however, its usefulness has been largely limited to furnishing guidance for the design of graphic languages [12, 23, 133] or for the development of pattern classification systems [22, 32, 89, 95, 118, 130]. In the case of graphic languages, linguistic specification has been imbedded in the data structure, that is, in the hierarchical relationships by means of which a line structure is specified in terms of constituent line structures, and these ultimately in terms of the primitives of the language (line segments, arcs, symbols, and points) in a way that reflects the functional relationships implicit in the depicted object. In the case of pattern classification systems, a syntax analysis of the line structure is made (after extraction from an image) and the structure is disassembled into a set of primitive classes. Class membership, which serves to categorize the line structure, is established by the ability of the line structure to be parsed according to a specific set of linguistic rules (of the pattern language) [32, 95].

Other Coding Schemes

Other coding schemes have been developed for representing irregular line structures. Some of these are simply variations on the chain code [75, 131], some are more compact representations [41, 111], and some have features that make them attractive for particular processing applications. In general, a coding scheme for line structures must satisfy three objectives: 1) it must faithfully preserve the information of interest; 2) it must permit compact storage and be convenient for display, and 3) it must facilitate any required processing. The three objectives are somewhat in conflict with each other, and any code necessarily involves a compromise among them.

A coding scheme developed by Merrill [88] deserves special mention. It differs considerably from the chain code and has some distinct advantages over it in certain applications. We shall refer to it as the *parallel-scan* coding scheme. In this scheme, a closed boundary contour is scanned along a set of closely spaced parallel lines, and the intersections with these scan lines are used to represent the boundary.

As in all computer representations, the boundary contour is assumed to be drawn on (or approximated by) a square grid, and the parallel scan lines are taken as the horizontal grid lines. These grid lines are scanned from the lowest-valued grid line to intersect the contour to the highest-valued one. For each scan, the x-coordinates of the boundary grid nodes are recorded. The set of all scans, each consisting in turn of a set of the x-coordinates of the intersected nodes, forms the

coded representation of the contour. The scheme utilizes the familiar topological property that a line crossing a closed boundary contour will intersect the contour an even number of times.

Three special conditions must be satisfied to make the representation unambiguous:

(1) The contour must be 8-connected (i.e., successive boundary nodes must be no further apart than $\sqrt{2}$ times the grid spacing—just as for the chain code).

(2) If two or more successive boundary nodes have the same y-coordinate and the previous and next y-coordinates are the *same* (indicating that the scan line is here tangent to the contour), and if the number of such boundary nodes is *odd*, then the last of these boundary nodes must be listed twice.

(3) If two or more successive boundary nodes have the same y-coordinate and the previous and next y-coordinates are *different* (indicating that the scan line is here passing through an inflection of the contour), and if the number of such boundary nodes is *even*, then the last of these boundary nodes must be listed twice.

Conditions (2) and (3) are required to assure that the previously mentioned topological property is not violated for a boundary contour consisting of connected nodes on a square grid.

We shall now illustrate the parallel-scan coding scheme with the aid of the closed contour shown in Figure 16. It is assumed that initially the full (x, y)-coordinates of all the boundary nodes are given, arranged in sequence (interior to the right) starting with the node marked $\times$:

(1, 3) (2, 4) (3, 3) (4, 3) (5, 2)
(4, 2) (3, 1) (3, 2) (2, 3) (2, 2)

To fulfil condition (2) the nodes (2, 4), (3, 1), (2, 3), and (2, 2) must each be listed twice. Nodes (4, 3) and (4, 2) must be listed twice to fulfill condition (3). Next, the nodes are sorted into groups having the same y-coordinate and then arranged according to increasing x-coordinate:

$y = 1$, (3, 1) (3, 1)
$y = 2$, (2, 2) (2, 2) (3, 2) (4, 2) (4, 2) (5, 2)
$y = 3$, (1, 3) (2, 3) (2, 3) (3, 3) (4, 3) (4, 3)
$y = 4$, (2, 4) (2, 4)

In coded form the contour is now represented simply by the ordered set of variable-length strings.

(2; 3, 3)
(6; 2, 2, 3, 4, 4, 5)
(6; 1, 2, 2, 3, 4, 4)
(2; 2, 2)

where the numbers to the left of the semicolons indicate how many x-coordinates follow. (Because of the intersection property this number will always be even.) The first string corresponds to the lowest y-coordinate, which must be specified separately; the remaining strings correspond to the other y-coordinates and differ successively by unity.

The code greatly facilitates the solution of certain problems. Suppose one desires to know whether a particular point (x_r, y_r) lies inside or on the closed contour. One need merely examine the string corresponding to y_r and determine whether x_r lies between two neighboring entries in the string such that the left entry is in an *odd* position in the string (the position to the right of the semicolon is taken as No. 1). This follows from the code's characteristic of bounding each area section of the closed contour with an odd-even pair of x-coordinates.

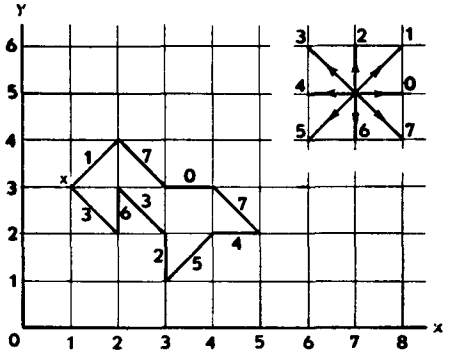


FIG. 16. A closed contour, chain-encoded.

The parallel-scan code is, of course, much less efficient than the chain code in terms of storage requirements; however, it is more convenient for certain *area-oriented* problems (such as, for example, determining whether a given point lies on or within a closed contour). For applications that are primarily shape-oriented, that is, applications in which the main interest is in the *shape* of the line structures themselves rather than in, say, the area enclosed by them, the chain code is more advantageous.

3. PROCESSING OF LINE STRUCTURES

The particular form of line-structure representation one selects can have an important effect on the relative ease with which line structures are processed in a digital computer. Ease-of-processing considerations can often outweigh efficiency of storage or convenience of display. For some applications, an ad hoc description scheme may have special advantages. Normally, however, a general-purpose description is likely to prove most convenient.

For general, irregular 2D line structures, the chain form of representation has shown itself to be well-suited for most of the processing tasks likely to be encountered. Simple, neat computer algorithms are easily derived. This has been especially true of analysis; but in a somewhat lesser sense it also applies to synthesis and manipulation. We shall illustrate the versatility of the chain representation by describing some of the more basic processing algorithms [40, 42, 48].

Some Simple Algorithms

*Inverse of a Chain**

Chains that are inverses of each other are geometrically congruent but oppositely directed. Thus

$$(a_1 \cdots a_n)^{-1} = a_n^{-1} \cdots a_1^{-1} \tag{4}$$

* All algorithms for the computation of chain properties require, of course, that any signal codes be separated out.

Length of a Chain

$$L = T(n_e + n_o\sqrt{2}) \tag{5}$$

where n_e and n_o are the numbers of even- and odd-valued links, respectively.

Width and Height of a Chain

Let

$$x_i = \sum_{j=1}^{i} a_{jx} + x_0 \tag{6}$$

$$y_i = \sum_{j=1}^{i} a_{jy} + y_0 \tag{7}$$

where a_{jx} and a_{jy} are, respectively, the x and y components of the link a_j (taken as a vector in 2-space). Then, for $j = 0, 1, \cdots, n$,

$$\text{width} = \max_j x_j - \min_j x_j \tag{8}$$

$$\text{height} = \max_j y_j - \min_j y_j \tag{9}$$

where x_0 and y_0 are the coordinates of the initium and may be arbitrarily selected.

Integration with Respect to x Axis

$$S = \sum_{i=1}^{n} a_{ix}(y_{i-1} + \tfrac{1}{2}a_{iy}) \tag{10}$$

where

$$y_i = y_{i-1} + a_{iy} \tag{11}$$

and y_0 is the ordinate of the initium. For a closed chain (initium = terminus), y_0 can be arbitrarily selected, and this formula will compute the net area encircled in a clockwise sense. (A negative sign will indicate net counterclockwise encirclement.)

The computation of length, width, height, and enclosed area for the chain of Figure 16 is illustrated in Table II. The results are easily verified by referring to the figure. Note that $(x_0, y_0) = (x_{10}, y_{10})$ as required for a closed chain.

First Moment About x Axis

$$M_1^x = \sum_{i=1}^{n} \tfrac{1}{2}a_{ix}[y_{i-1}^2 + a_{iy}(y_{i-1} + \tfrac{1}{3}a_{iy})] \tag{12}$$

TABLE II. COMPUTATIONS FOR CHAIN OF FIGURE 16

i	a_i	x_i	y_i	$a_{ix}(y_{i-1} + \frac{1}{2}a_{iy})$
0	—	1	3	—
1	1	2	4	$3\frac{1}{2}$
2	7	3	3	$3\frac{1}{2}$
3	0	4	3	3
4	7	5	2	$2\frac{1}{2}$
5	4	4	2	-2
6	5	3	1	$-1\frac{1}{2}$
7	2	3	2	0
8	3	2	3	$-2\frac{1}{2}$
9	6	2	2	0
10	3	1	3	$-2\frac{1}{2}$

Length $= n_e + n_o\sqrt{2} = 4 + 4\sqrt{2}$
Width $= \max x_i - \min x_i = 4$
Height $= \max y_i - \min y_i = 3$
Area $= \sum_{i=1}^{10} a_{ix}(y_{i-1} + \frac{1}{2}a_{iy}) = 4$

Second Moment About x Axis (moment of inertia)

$$M_2 = \sum_{i=1}^{n} \tfrac{1}{3}a_{ix}[y_{i-1}^3 + \tfrac{3}{2}a_{iy}y_{i-1}^2 + a_{iy}^2 y_{i-1} + \tfrac{1}{4}a_{iy}] \quad (13)$$

Formulas for computing moments about the y axis or either of the two diagonal axes (45° and 135°) are similar and can be easily derived [40].

Residue of a Chain

The residue of chain A, denoted by $\mathcal{R}(A)$, is the chain of minimum length from the initium to the terminus of A and which has its links arranged in ascending value.

To compute $\mathcal{R}(A)$, we shift the initium of A to the origin of a new, x', y' coordinate system. Thus

$$x_n' = x_n - x_0 = \sum_{i=1}^{n} a_{ix} \quad (14)$$

$$y_n' = y_n - y_0 = \sum_{i=1}^{n} a_{iy} \quad (15)$$

Next we determine the octant containing the terminus (x_n', y_n'). If (x_n', y_n') lies in the octant defined by the link angles $m \cdot 45°$ and $(m + 1) \cdot 45°$ relative to the x axis ($m = 0, 1, \cdots, 7$), then $\mathcal{R}(A)$ will consist solely of a concatenation of links of value m and $m + 1$ arranged in ascending value. The value of m and the respective occurrences of links of value m and $m + 1$ can be found from Table III. Figure 17 illustrates the residue determination for the chain 32301 061210. We find $x_n' = 4$, $y_n' = 6$, $x_n' > 0$, $y_n' > 0$, and $|x_n'| - |y_n'| < 0$. Hence (x_n', y_n') lies in octant II, $m = 1$, and the residue will consist of 4 links of value 1, and 2 links of value 2; that is $\mathcal{R}(A) = 111122$, as can be readily verified from Figure 17.

TABLE III DETERMINATION OF CHAIN RESIDUE

Signum			*Octant*	m	*Occurrences of links of value*	
x'	y'	$\lvert x' \rvert - \lvert y' \rvert$			m	$m+1$
+	+	+	I	0	$x'-y'$	y'
+	+	−	II	1	x'	$-x'+y'$
+	−	+	VIII	7	$-y'$	$x'+y'$
+	−	−	VII	6	$-x'-y'$	x'
−	+	+	IV	3	y'	$-x'-y'$
−	+	−	III	2	$x'+y'$	$-x'$
−	−	+	V	4	$-x'+y'$	$-y'$
−	−	−	VI	5	$-x'$	$x'-y'$

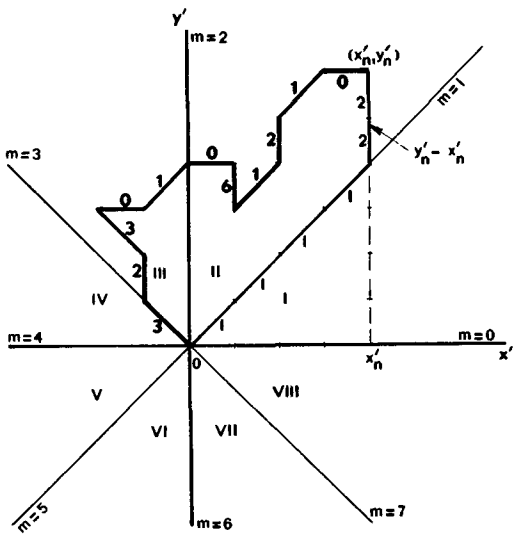


FIG 17. Determination of chain residue.

Link-Pair Residues

The residue of many two-link chains, $a_r a_s$, is a single link, a pair of even-valued identical links, or a null chain. These link-pair residues are listed in Table IV. The replacement within a chain of a pair of (not-necessarily-adjacent) links by their link-pair residue will reduce the length of the chain without affecting the relative positions of the chain's initium and terminus. When judiciously carried out, such substitution can also smooth a chain. If *all* possible link-pair residue substitutions are made, the result will be a minimum-length chain from initium to terminus. Rearrangement of the remaining links in ascending numerical order will yield the residue of the original chain [39].

Distance Between Two Points

The distance between two points connected by the chain A is given by

$$d = [(\sum_{i=1}^{n} a_{ix})^2 + (\sum_{i=1}^{n} a_{iy})^2]^{\frac{1}{2}} \quad (16)$$

or, alternatively, by

$$d = T[(r_e + r_o)^2 + r_o^2]^{\frac{1}{2}} \quad (17)$$

TABLE IV Link-Pair Residues

a_r \ a_s	0	1	2	3	4	5	6	7
0			1	2	X	6	7	
1				22	2	X	0	00
2	1				3	4	X	0
3	2	22				44	4	X
4	X	2	3				5	6
5	6	X	4	44				66
6	7	0	X	4	5			
7		00	0	X	6	66		

X denotes a null residue A blank space indicates that the link pair is already a minimum-length chain

where r_e and r_o are the respective occurrences of even- and odd-valued links in $\mathfrak{R}(A)$. For the chain of Figure 17, $r_e = 2$, $r_o = 4$, and hence $d = \sqrt{52}$.

Mirror-Inverse Chains

For certain manipulation purposes it is desirable to generate a chain that is the mirror inverse (mirror "image") of a given chain relative to one of the four major axes of the chain grid—horizontal (x) axis, antidiagonal ($x = y$) axis, vertical (y) axis, and diagonal ($-x = y$) axis. We define two links to be mirror inverses of each other, relative to a major grid axis, if rotation of 180° about this axis lying in the plane of the links will map either link into the other. Thus, for example, the links 1 and 3 are vertical-axis mirror inverses, and the links 1 and 7 are horizontal-axis mirror inverses. Mirror-inverse link pairs can be determined from the following relations, which are easily deduced from the chain coding matrix in the lower right-hand corner of Figure 13.

$$\begin{aligned} &\text{horizontal axis} && a_i^h = 8 \dot{-} a_i \\ &\text{antidiagonal axis} && a_i^a = 2 \dot{-} a_i \\ &\text{vertical axis} && a_i^v = 4 \dot{-} a_i \\ &\text{diagonal axis} && a_i^d = 6 \dot{-} a_i \end{aligned} \quad (18)$$

Let A^u be the mirror-inverse chain to chain A relative to axis u. We write:

$$A^u = (a_1 a_2 \cdots a_n)^u = a_1{}^u a_2{}^u \cdots a_n{}^u \quad (19)$$

Also,

$$(A^u)^u = A \quad (20)$$

For illustration, the h-axis mirror-inverse chain for the chain 32301 061210 of Figure 17 is 56507 027670.

Chain Rotation

To rotate a chain about an axis through the initium and perpendicular to the grid plane by a multiple of 90°, it is merely necessary to add (modulo 8) the same multiple of 2 to each link of the chain.

To rotate a chain by any angle other than a multiple of 90°, all link nodes must be individually rotated and the resulting ro-

tated line structure must be re-quantized. For this purpose it is sufficient to compute the coordinates of the rotated link nodes and round them off to the nearest grid node. The rotated chain is then readily obtained by connecting the resulting sequence of grid nodes. Some post-editing of the chain may be required to reduce the distortion introduced by the re-quantization. Such editing would consist primarily of replacing adjacent links differing in value by more than unity by their link-pair residue—unless the two links describe a corner which was also present in the original chain [48, 111].

Chain Expansion and Contraction

To expand or contract a chain by a specified scale factor, one must appropriately scale each chain link and then re-quantize using the same scheme as described for chain rotation. In contraction, a number of links may merge into one, and in expansion one link may cause a string of many links to be generated. The generation of these new nodes is most easily handled by means of a straight-line approximation algorithm [13, 105, 111].

High-Level Polygonal Line Structure Transformation

The chain representation is not particularly convenient for rotation, expansion, contraction, or any of the more general transformations (e g , conic projection to Mercator map projection) [83, 111]. Hence—unless the chain form is required to preserve an extensive amount of fine detail—it is likely to be advantageous first to convert to a high-level polygonal representation before carrying out the transformation [92, 109, 111]. This may be true even for some applications in which it is necessary eventually to return to the chain form for display purposes (i.e., if the results are to be displayed with a digital plotter).

Contour Correlation

To determine the degree of similarity in shape and orientation between two contours we can make use of the *chain correlation function,* so named because it resembles the familiar correlation function of communication theory. Chain correlation can be applied to chains describing open or closed contours. The procedure is as follows [33, 41]:

Given two chains

$$A = a_1 a_2 \cdots a_n$$

$$B = b_1 b_2 \cdots b_m$$

where $n \leq m$, we define a chain crosscorrelation function $\Phi_{ab}(j)$ for chain A with chain B by

$$\Phi_{ab}(j) = \frac{1}{n} \sum_{i=1}^{n} \cos\,(a_i - b_{i+j})\pi/4 \quad (21)$$

This function provides a measure of the average pair-wise alignment between the links of A and B, and thus gives an indication of the degree of shape congruence for different shifts of B relative to A. Note that

$$|\,\Phi_{ab}(j)\,| \leq 1 \qquad \text{for all } j \quad (22)$$

and, if $n = m$,

$$\Phi_{ab}(j) = \Phi_{ba}(-j) \quad (23)$$

If we let $A = B$, we obtain the *chain autocorrelation function,* $\Phi_{aa}(j)$, which characterizes the chain and can be used for contour classification purposes.

The chain crosscorrelation function is sensitive both to the relative shapes of the contours as well as to their relative orientation. Hence to determine similarity in shape independent of orientation, one chain must first be rotated to align it with the other. One way of accomplishing this is to compute the angle between the x axis and the vector from the initium to the terminus for each chain, and then to rotate one chain to reduce the angular difference to zero. (The angles are readily computed from the ratio of the sums of the x and y components of the chain elements.) Some care must be exercised that the initium-to-terminus line is indeed representative of the orientation of the chain, and is not unduly affected by end effects. If this assumption is not valid, a more sophisticated measure of orientation may be required [111].

Smoothing of Chains

A well-quantized chain should not contain any two adjacent links whose values differ by more than unity, except at sharp corners.

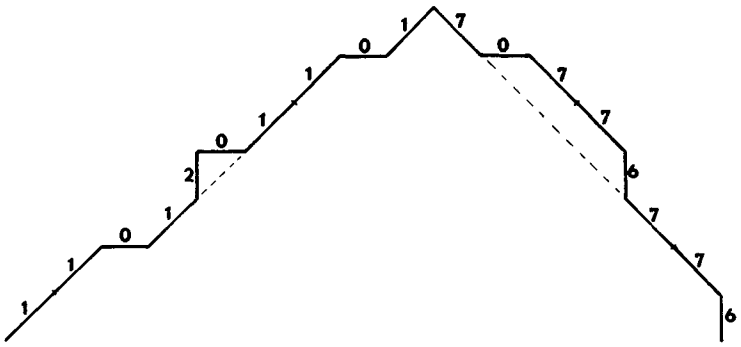


FIG. 18. Smoothing of chains.

Where this requirement is not fulfilled, smoothing is indicated. Smoothing can be both "local" (link-to-link) or "global" (averaged over a large number of links). The presence of a sharp corner can be detected by a change in a global trend. Thus in Figure 18, the link change from 2 to 0 should be regarded as noise and replaced by a link of value 1 (the corresponding link-pair residue). However, the link change from 1 to 7 reflects a global trend change from an extended link-0, 1 direction to a predominantly link-7 direction, and should be retained (or possibly even inserted if not already present). The more gradual 0776 turn at the right must be judged in terms of subsequent curvature trends; if the section following the 07 turn is to resemble a straight line, the 0776 sequence should be replaced by its residue, that is, by 777, as indicated by the dashed line segment [42, 111, 114].

The chain of a properly encoded straight line will possess three important properties [48]: 1) it will contain links of no more than two different values; 2) the two link values will differ by unity, modulo 8; and 3) one of the link values will always occur singly and will be as uniformly spaced as possible among links of the other value. Thus a chain known to represent a straight line must possess these properties; if it does not, the chain has been corrupted by noise (perhaps from a rotation or other transformation) and specific noise removal is indicated. An analogous approach can be taken for chains representing gently curving arcs [111]. Also, conversion from the chain form of representation to a high-level polygonal form can prove to be an effective method for introducing smoothing [70, 109].

Line Structure Synthesis

Since in synthesis a line structure is specified to possess a given set of properties, such line structures tend to be fairly regular and describable by a modest number of long line segments, arcs of circles and (only rarely) other conic sections. One reason for this is that a synthesized line structure is likely to serve as a description for some machine part that is to be manufactured, and the machining of other than flat or cylindrical surfaces is costly. Some exceptions are in the making of dies and cams (two-and-three dimensional), where highly complex and irregular surfaces may be required.

An extensive set of computer languages has been developed during the past decade for specifying line structures, for generating line-structure displays (both for 2D line structures as well as for 2D projections of 3D line structures), for manipulating the *data structures* in which the hierarchical relationships of the line structures and their components are imbedded, and for analyzing the line structures at various stages of their development [21, 23, 29, 97, 133, 137]. In all these computer graphics languages, line structures are described as high-level polygonal structures, usually in a hierarchical manner, in which one polygonal structure becomes a substructure, possibly repeated many times, in a higher-level structure, to as many levels as deemed desirable [30, 149].

There is seldom a need for the chain form of representation when synthesizing line structures. One common exception, though, is the generation of the display commands for a digital plotter [13, 105]. An instance in which chain synthesis has been found to be a valuable tool has been the description of curves (2D and 3D) for which the determination of an analytic expression was not feasible, and where a point-by-point computation would have been prohibitive [151]. With chain synthesis it is necessary only that one point on the curve be determined by conventional means. Thereafter one need merely test the grid nodes surrounding the last data node to determine which node lies closest to the desired curve. Once the general direction of the desired curve is known, the testing of two nodes should suffice in most instances (four, for 3D curves) [47].

4. SOME APPLICATIONS

Applications for line structure processing are numerous and widespread. Those involving line structure synthesis are generally known under the term of "computer graphics." Increasingly, computer graphics synthesis procedures are taking over the traditional drafting function in engineering, from the design of machine parts and electronic circuits, to the lofting of ships' hulls, the layout of offices, and the design of aesthetic soft-drink bottles [21, 107]. A closely related application area is computer animation, where the many images required for a moving picture sequence can be economically generated under computer control [6, 60, 73, 139].

Analysis, manipulation, and pattern recognition applications tend to be grouped under the heading of picture or image processing in the literature [118]. Specific application areas include aerial and biomedical photography interpretation [24, 28, 64, 65, 106, 122, 147], map generation [2, 8, 93, 99], and scene analysis ("robotics") [14, 27, 36, 61, 84]. We shall now briefly describe a few applications to illustrate the use of the processing techniques described earlier.

Pattern Fitting Problems

One important class of problems involving line structure processing consists of those that can be classed as "pattern fitting" problems. These problems exist in a great variety of forms, but they are all characterized by the need to translate, rotate, and otherwise manipulate irregular contours with the ultimate objective of matching certain geometric features among them. Examples of this class of problems are found in map matching [141], optimum template layout for sheet metal cutting [1], the computer-guided assembly of ancient broken pottery [45], and the karyotyping of chromosomes [65]. Each of these problems is a kind of "jigsaw puzzle", and, indeed, the computer solution of an actual jigsaw puzzle is perhaps the best way in which to illustrate the solution of problems of this type. A jigsaw puzzle exhibits all the typical requirements for line structure processing encountered in pattern fitting problems, and yet is devoid of the peripheral complexities that are not of interest here but attend every real-life engineering problem.

Let us consider the six-piece puzzle shown in Figure 19; the problem is to rotate and translate the arbitrarily oriented pieces shown in (a) into the tightly fitting cluster shown in (b). We shall utilize only the shape (contour outline) of the pieces, and assume that the pieces contain no usable pictorial information or texture. That is, we shall develop a solution method that does not assume the existence of such information; if pictorial or textural clues are available, they can be used to simplify the techniques later. In most practical applications such clues are, of course, likely to be present. The method should be usable for puzzles of any reasonable size, and degrade only gradually as the number of pieces grows without bound. Clearly, therefore, all unstructured searching and hit-or-miss procedures must be avoided.

A necessary condition for two pieces to fit together is that within their respective contours they each contain a section having a shape common to both. Thus if piece 1 in Figure 19(a) is to fit with some other piece along the section marked by the points *a*, *b*,

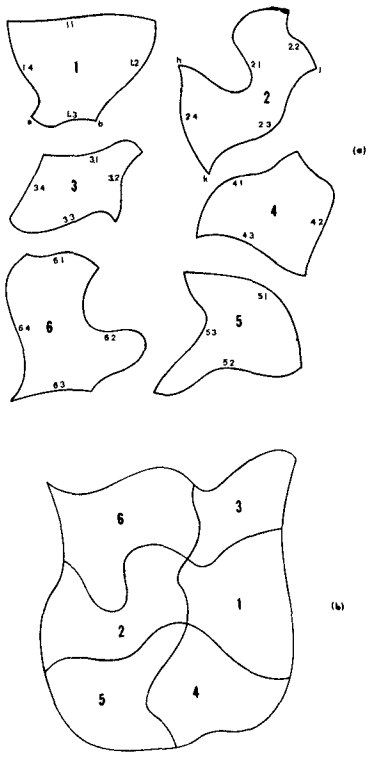


FIG 19 A six-piece jigsaw puzzle (a) Given set of pieces, (b) Completed assembly.

then the other piece must have within its contour a section which, except for being rotated, is geometrically identical to this section. Of course, what constitutes a contour section is a priori unknown, and, in any case, those contour sections facing the exterior will not have a matching section. Also, in practice, fitting sections will not be precisely identical because of imperfections in the fitting and limitations in the contour description (quantization noise).

A possible approach for assembling a jigsaw puzzle thus might be to identify all piece-contour sections, classify these sections according to some of their rotation-invariant, geometric features, and then to select matching pairs of sections. This is indeed the general approach we shall follow. The key subproblems are: 1) determining the contour sections; 2) classifying the sections; and 3) carrying out the actual assembly in spite of the presence of wrongly-formed sections and ambiguous section pairs. We shall concentrate here only on the aspects of the problem that are pertinent to line structure processing. A complete discussion of the jigsaw puzzle problem would take us too far afield, and is, anyway, given elsewhere [43].

There is no way of determining with certainty the section of a piece contour that will fit against some other piece. The only clue for the extent of a contour section is the presence of slope discontinuities in the contour. Inspection of Figure 19(b) will show that at every junction, that is, at every point at which three or more pieces join, at least one of the pieces must have a sharp discontinuity in the contour slope.* Thus although a particular piece may not have a slope discontinuity at a junction, and though slope discontinuities may exist at points that are not junctions, the existence of a slope discontinuity in a piece contour provides a strong hint that the point of discontinuity is a junction. Hence contour sections with a high probability of fitting in their entirety to another contour section can be obtained by cutting the piece contours at all points of slope discontinuity—such as at points h, i, j, and k of piece 2 in Figure 19 (a). The sections are then numbered in decimal fashion to correspond to the piece to which they belong; e.g., 2.1, 2.2, 2.3, and 2.4.

For an application such as the assembly of jigsaw puzzles it is best to identify the slope discontinuities at the time of chain encoding. Alternatively, they can also be identified in the piece contour chain, and this chain then segmented into "chainlets" corresponding to the contour sections. The latter scheme, however, requires finer quantization for the contour, and the probability of introducing errors is increased. Of course, some errors can be tolerated since normally each piece "mates" with three or four other pieces, and there is always another chance for finding the way in which a particular piece is to fit into the whole.

Once all the contours have been segmented into contour sections, the corresponding section chains ("chainlets") must be classified according to some rotation-invariant geometric features. The best features for this

* More precisely, at an n-piece junction, at least $n - 2$ of the pieces must have a slope discontinuity.

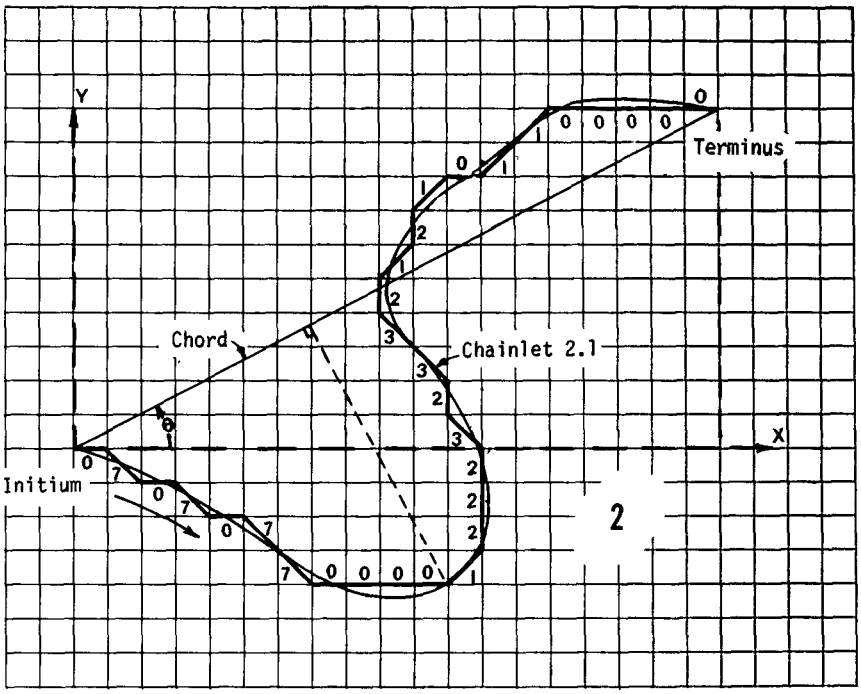


FIG 20. Feature determination for a contour section.

purpose are those that are relatively insensitive to measurement and quantization noise, and also do not involve excessive computation. Five features that meet this objective are: 1) length of the chainlet, 2) length of the chainlet's chord (i.e., the straight line between the chainlet's end points); 3) the net positive area of the piece lying between the chainlet and its chord; 4) the maximum separation between chainlet and chord to the left side of the chord (maximum peninsula); and 5) the maximum separation between chainlet and chord to the right of the chord (maximum bay). The features are illustrated in Figure 20, where the contour section 2.1 of Figure 19 is shown in enlarged form, together with its chain approximation. Note that the piece interior lies to the right of the chainlet as the latter is traversed from initium to terminus.

Chainlet 2.1 is represented by the sequence

07070 77000 01222 32332 12101 10000 0

It consists of 19 even-valued and 12 odd-valued links, giving a chain length of $19 + 12\sqrt{2} = 35.97$. The calculation for the remaining features is illustrated in Table V. To find the chord length, the sums of the x and y components, $\sum_{i=1}^{n} a_{ix}$, $\sum_{i=1}^{n} a_{iy}$, are computed, yielding the differences in x and y between the terminus and initium, 19 and 10, respectively. From these, the chord length is found to be $\sqrt{19^2 + 10^2} = 21.47$, and the angle between chord and x axis is given by $\tan^{-1} 10/19 = 27.8°$. The net positive area is found by taking the total area under the chainlet, using formula (10), and then subtracting the area under the chord. From Table V, this yields $52 - \frac{1}{2}(19)(10) = -43$ square units, with the negative sign indicating that the total "bay" area exceeds the total "peninsula" area by this amount. (Peninsula area is counted as positive and bay area as negative.)

To determine the maximum distance to the chain from the chord, to both the left and the right, one determines the components of the chain links perpendicular to the chord, and then monitors the running sum for both its maximum (maximum peninsula) and minimum (maximum bay). Figure 21 illustrates how the links are resolved into

TABLE V. FEATURE CALCULATIONS FOR CHAINLET OF FIG. 20

i	a_i	a_{ix}	a_{iy}	y_i	ΔS_i	S	a_{iv}	Σa_{iv}	i-max	i-min
0	—	—	—	0	—	—	—	—	0	0
1	0	1	0	0	0	0	$-b$	(0, −1, 0, 0)	0	1
2	7	1	−1	−1	−½	−½	$-c$	(0, −1, −1, 0)	0	2
3	0	1	0	−1	−1	−1½	$-b$	(0, −2, −1, 0)	0	3
4	7	1	−1	−2	−1½	−3	$-c$	(0, −2, −2, 0)	0	4
5	0	1	0	−2	−2	−5	$-b$	(0, −3, −2, 0)	0	5
6	7	1	−1	−3	−2½	−7½	$-c$	(0, −3, −3, 0)	0	6
7	7	1	−1	−4	−3½	−11	$-c$	(0, −3, −4, 0)	0	7
8	0	1	0	−4	−4	−15	$-b$	(0, −4, −4, 0)	0	8
9	0	1	0	−4	−4	−19	$-b$	(0, −5, −4, 0)	0	9
10	0	1	0	−4	−4	−23	$-b$	(0, −6, −4, 0)	0	10
11	0	1	0	−4	−4	−27	$-b$	(0, −7, −4, 0)	0	11
12	1	1	1	−3	−3½	−30½	d	(0, −7, −4, 1)	0, 12	11
13	2	0	1	−2	0	−30½	a	(1, −7, −4, 1)	0, 13	11
14	2	0	1	−1	0	−30½	a	(2, −7, −4, 1)	0, 14	11
15	2	0	1	0	0	−30½	a	(3, −7, −4, 1)	0, 15	11
16	3	−1	1	1	−½	−31	c	(3, −7, −3, 1)	0, 16	11
17	2	0	1	2	0	−31	a	(4, −7, −3, 1)	0, 17	11
18	3	−1	1	3	−2½	−33½	c	(4, −7, −2, 1)	0, 18	11
19	3	−1	1	4	−3½	−37	c	(4, −7, −1, 1)	0, 19	11
20	2	0	1	5	0	−37	a	(5, −7, −1, 1)	0, 20	11
21	1	1	1	6	5½	−31½	d	(5, −7, −1, 2)	0, 21	11
22	2	0	1	7	0	−31½	a	(6, −7, −1, 2)	0, 22	11
23	1	1	1	8	7½	−24	d	(6, −7, −1, 3)	0, 23	11
24	0	1	0	8	8	−16	$-b$	(6, −8, −1, 3)	0, 23	11, 24
25	1	1	1	9	8½	−7½	d	(6, −8, −1, 4	0, 23, 25	11, 24
26	1	1	1	10	9½	2	d	(6, −8, −1, 5)	0, 23, 26	11, 24
27	0	1	0	10	10	12	$-b$	(6, −9, −1, 5)	0, 23, 26	11, 24, 27
28	0	1	1	10	10	22	$-b$	(6, −10, −1, 5)	0, 23, 26	11, 24, 28
29	0	1	0	10	10	32	$-b$	(6, −11, −1, 5)	0, 23, 26	11, 24, 29
30	0	1	0	10	10	42	$-b$	(6, −12, −1, 5)	0, 23, 26	11, 24, 30
31	0	1	0	10	10	52	$-b$	(6, −13, −1, 5)	0, 23, 26	11, 24, 31
		—	—		—					
		19	10		52					

Evaluations of (a, b, c, d): $a = \cos\theta$ $c = \cos\theta + \sin\theta$
$b = \sin\theta$ $d = \cos\theta - \sin\theta$

$0 - (0, 0, 0, 0) = 0 = 0$
$11 - (0, -7, -4, 0) = -11\sin\theta - 4\cos\theta = -8.66$ (max bay)
$23 - (6, -7, -1, 3) = -11\sin\theta + 8\cos\theta = 1.95$
$24 - (6, -8, -1, 3) = -12\sin\theta + 8\cos\theta = 1.49$
$26 - (6, -8, -1, 5) = -14\sin\theta + 10\cos\theta = 2.33$ (max peninsula)
$31 - (6, -13, -1, 5) = -19\sin\theta + 10\cos\theta = 0$

components directed along the chord (u axis) and perpendicular to the chord (v axis). The respective v-axis components, a_{iv}, are:

0	$-b$	4	b
1	d	5	$-d$
2	a	6	$-a$
3	c	7	$-c$

where $a = \cos\theta$, $b = \sin\theta$, $c = \cos\theta + \sin\theta$, and $d = \cos\theta - \sin\theta$.

Rather than numerically evaluate the running sum $\sum_{j=1}^{i} a_{jv}$ for each i, it is preferable to keep the sum as a four-tuple (a, b, c, d), retaining only those four-tuples that are not covered by others for either the maxi-

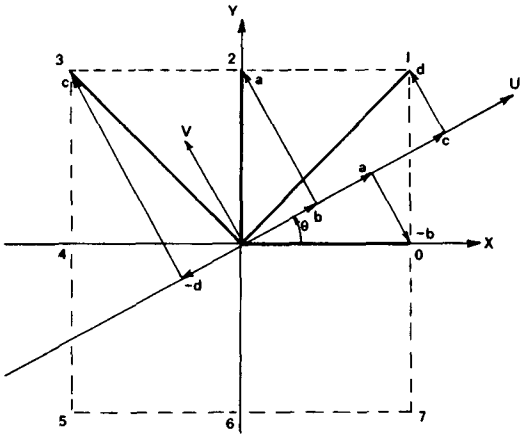


FIG. 21. Resolution of chain links into rotated coordinate system (u, v) aligned with chord of Fig. 20.

mum or the minimum, and evaluating only those that remain at the end. This avoids the possibility of error accumulation due to round-off in the trigonometric functions. In Table V, the column headed "a_{iv}" lists the v-axis components, "$\sum a_{iv}$" gives the running sum as the four-tuple (a, b, c, d), and "i-max" and "i-min" list the values of i for which the four-tuples are not as yet covered by other four-tuples for either maximum or minimum, respectively. Final evaluation then shows the maximum separation to the left (peninsula) to be 2.33 and to occur following link number 26, and the maximum separation to the right (bay) to be 8.66 and to occur following link number 11, as is readily confirmed from an examination of Figure 20. In summary, we have for the five-element feature vector of chainlet 2.1:

chain length	35.97
chord length	21.47
net area	−43
max peninsula	2.33
max bay	−8.66

Similar calculations are now made for all the contour sections in the jigsaw puzzle, and the feature vectors are then sorted according to the identification number of the contour section. Next, a five-dimensional file is created, in which the contour sections are sorted according to each of the five features, creating a kind of "feature catalog."

To start the actual assembly process, one selects an initial contour section at random, and looks up its features in the catalog, using the section identification number (e.g., 2.1). Taking the first feature, one lists all chainlets for which this feature is of a value that agrees within a specified (but relatively wide) tolerance with that of the selected chainlet. One then selects a second feature, and deletes from the list all chainlets for which the second feature does not satisfactorily agree with that of the selected chainlet. This is continued until only those chainlets remain on the list for which all the measured features agree within specified tolerance with those of the selected chainlet. More formally, given chainlet i, we retain on the list all chainlets j, $j = 1, 2, \cdots, n$, $j \neq i$, such that

$$\bigwedge_k \{ |f_{ik} - f_{jk}| < T_k \} \tag{24}$$

where T_k is the threshold (tolerance) applied for feature k.

The remaining chainlets on the list are next ordered according to their weighted distance from the selected chainlet in the multidimensional feature space. We write

$$S_{ij} = [\sum_k (f_{ik} - f_{jk})^2 w_k]^{\frac{1}{2}} \tag{25}$$

where S_{ij} ranges from a minimum of 0 (perfect match) to an upper bound of

$$S_{\max} = [\sum_k T_k^2 w_k]^{\frac{1}{2}} \tag{26}$$

The weights w_k are assigned to reflect the relative importance of the features and depend on reliability of measurement, sensitivity to noise, anticipated range of variation, and similar considerations.

The chainlet with the lowest S_{ij} value is then taken as the one most likely to fit to chainlet i. Since its orientation will normally be different from that of chainlet i, it must be rotated before the actual fit can be verified. The required rotation angle is the difference in the angles that the two chords make with the x axis. Rotation of a chain (by other than a multiple of 90°) requires re-quantization and re-encoding. For illustration, let us consider chainlet 6.2 (corresponding to contour section 6.2 in Figure 19(a)). It is given

by

45455 56566 67070 77666 55444 34445 4

Its feature vector is given by:

chain length	36.38
Chord length	21.63
net area	33.5
max peninsula	7.97
max bay	−2.78

and its angle with the x axis is $\tan^{-1}\frac{3}{2} = 56.3°$. The feature vectors for 2.1 and 6.2 agree well in chain length and chord length. The only moderate agreement in the remaining three features is attributable here to the relatively coarse quantization. However, for this example, where only six pieces are to be assembled, the quantization is adequate to indicate that 6.2 should fit to 2.1 (rather than some other chainlet). Chainlet 6.2 is shown in Figure 22. Upon rotation by the difference in the chord angles (27.8° − 56.3° = −28.5°), re-quantization and inversion, chainlet 6.2R is obtained:

00707 60700 01122 23332 21111 00170 0

as also shown in Figure 22.

Chainlets 2.1 and 6.2R now have their chords parallel, and precise shape matching

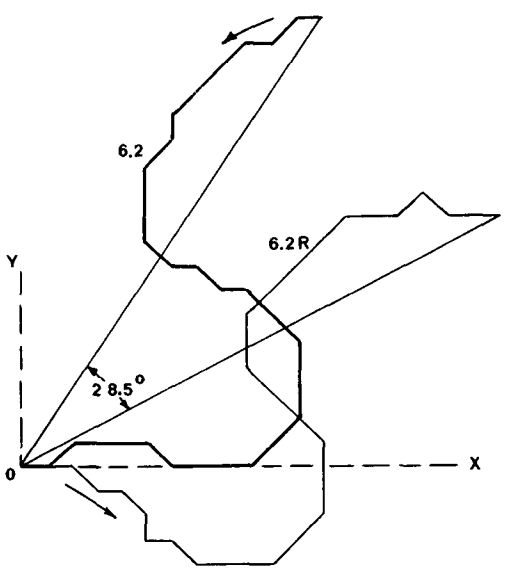


FIG. 22. Chainlet 6.2 before and after rotation and inversion.

is possible. One means for matching is to use the chain correlation function (21), with a shift range of, say, ±3 links to allow for quantization-error shifts in the respective end points. If this is done for chainlets 2.1 and 6.2R, a maximum cross-correlation value of 0.883 is obtained at a shift of +1. This is a satisfactorily high correlation. As a final check, the two pieces are merged into one by forming a new combined piece, consisting (in sequence) of sections 2.2, 2.3, 2.4, 6.3, 6.4, and 6.1. The new combined contour may not have any self-intersections. If this condition is also satisfied, one may proceed under the assumption that pieces 2 and 6 indeed fit along sections 2.1 and 6.2, respectively. If the fit had failed, the chainlet with the next lowest value of S_{ij} would have been tried next. If no fitting chainlet is found, it must be assumed that either 1) chainlet 2.1 belongs to the exterior boundary; or 2) the fitting is not pairwise with another contour section. In either case, one would then simply go to chainlet 2.2 and repeat the procedure until the assembly is completed.*

Map-Matching Problem

A variation on the jigsaw puzzle problem is the geographic map matching problem in which a large contour, representing, say, a coastline, is to be searched to determine where, if at all, a segment of coastline obtained from an aerial photograph will fit to the contour [33]. In this, as in most geographic map problems, it can be presumed that the North direction is known to a high degree of precision, and that fitting thus requires only the carrying out of an efficient translational search. The general problem is illustrated in Figure 23, where the contour segment A is to be located for best fit along the contour B.

To solve this problem, one encodes both the large contour as well as the contour segment in the chain code and utilizes the cross-correlation function (21). A complete segment-to-contour correlation is, however, both overly tedious as well as unnecessary.

* In solving a complex jigsaw puzzle, one must provide for many special situations that may arise. The interested reader is referred to the literature for a more detailed description of this problem [43].

Instead, one makes a series of searches, each progressively finer but limited to only those sections of the large contour that according to the previous search are likely sites for the best fit. This is accomplished by slightly modifying (21) so as to permit two kinds of skipping—a skip (in multiples of p) in the successive link products that are taken into the correlation sum (a skip in the index i) and a skip (in multiples of q) in the successive correlation shifts that are computed (a skip in the index j). Equation (21) then takes on the form

$$\Phi_{ab}(qj) = \frac{1}{\Gamma n/p} \sum_{i=1}^{\Gamma n/p} \cos\,(a_{pi-p+1} - b_{pi-p+1+qj})\pi/4 \tag{27}$$

where the symbol Γx denotes the smallest integer not exceeded by x.

For the initial search, the largest reasonable values of q and p are selected* and a full correlation sweep is made. Those portions of the correlation function showing the three highest peaks are then identified and, for the next step, only these portions (constituting perhaps 15% of the whole range) are searched, using values of p and q that are about $\frac{1}{3}$ of the initial values. From the resulting correlation functions sections, the three highest peaks are identified, and the process is repeated, at this step possibly with $p = q = 1$. The position of the highest peak then corresponds to the position of best fit. The closer the peak value approaches 1.0, the better the fit and the lower the noise level. This procedure is easily implemented, is fairly fast, and has been found to be effective for problems of this type [33].

A somewhat related map processing problem is the so-called *contour map search problem*, in which an elevation profile is to be fitted to a terrain contour map. An example of this problem is offered by a pilot desiring to make a precise aerial survey of a wilderness area. It is assumed that a terrain contour map of the area is available. (Such a map could, for example, be obtained from sets of aerial stereo photographs.) The pilot carries out his survey by flying a particular level flight path and recording both the exact shape of the path as well as the terrain elevation below his aircraft, on magnetic tape. Upon returning to his base, the recorded path data and the terrain elevation data are entered in a computer, together with the data for the terrain contour map. The objective of the computer analysis is to fit the flight path, with its terrain elevation profile, to the contour map and thus determine the precise geographical location of the flight path on the map. A typical chain-encoded terrain contour map is shown in Figure 24. Exhaustive searching of the contour map would be prohibitive. Instead, a heuristic approach is required in which sections of the map where the flight path could not possibly fit are progressively eliminated. This problem is another instance where some of the line structure processing techniques de-

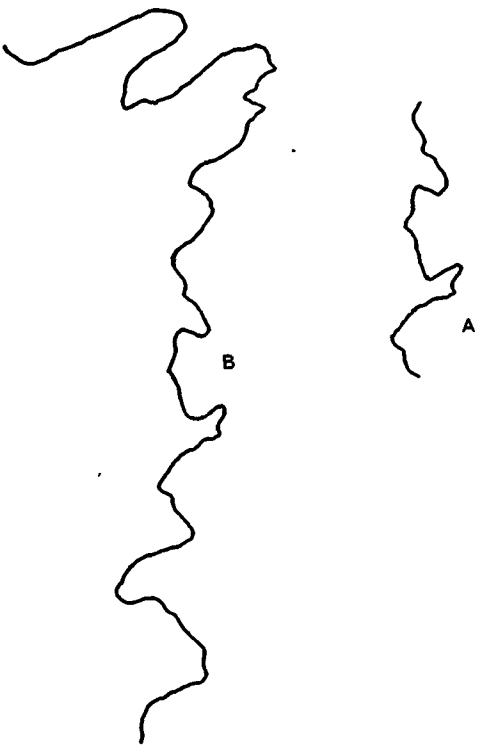


FIG 23. Illustration of the map-matching problem

* As based on experience, the fineness of the quantization relative to the significant detail, the coherence among strings of links, and the amount of noise expected If q is too large the position of best fit may be entirely missed. Normally, both p and q should not exceed 10

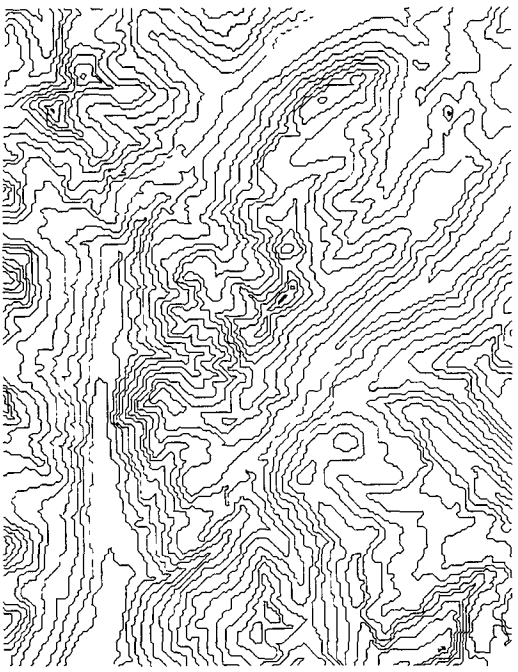


FIG. 24 Chain-encoded contour map. (From S. P Morse, "Generalized computer techniques for the solution of contour-map problems", Ph.D dissertation, New York University, 1967)

scribed earlier have been effectively utilized [8, 44, 88, 93].

Optimum Two-Dimensional Layout

A graphics problem of great interest to industry is that of optimum two-dimensional layout. This is the problem of determining how to cut irregularly-shaped pieces out of given stock sheets in an optimum manner without exhaustively trying all possible arrangements. Optimum layout problems* are associated with sheet metal cutting, clothing manufacture, leather cutting, glass cutting, and paper stock cutting, among others. In a general form, the problem is also encountered in factory and office layout, in parking-lot design, and in land development. Optimum layout has some similarity with the problem of jigsaw puzzle assembly, however, in its critical aspects it is in fact very different since no unique, tightly fitting solution can normally be expected [1, 15, 49, 63, 85].

* Sometimes also referred to as "nesting problems."

The general objective in all optimum layout problems is to find an arrangement for cutting the pieces that most effectively utilizes the available resource (i.e., the stock sheet). This objective can be expressed in terms of an objective function that is to be optimized by means of one of the standard mathematical optimization techniques [7]. In its simplest form, the objective function may simply describe the waste area. In more complex problems, the objective function may have to take into account that the pieces to be cut inherently have different economic worths, that the value of the pieces may vary with location and orientation on the sheet, and that the amount of cutting itself has an associated cost.

Optimum two-dimensional layout problems divide into two broad classes. In the so-called *template-layout* problems the objective is to cut as many pieces as possible from a single stock sheet. In the so-called *cutting-stock* problems, the objective is to cut a fixed number of pieces from as small a stock sheet (or as few stock sheets) as possible. In the first the objective is to make optimum use of a fixed resource, whereas in the second it is to meet a fixed demand in an optimum manner. The two classes of problems, though closely related, require different approaches for their solution. The discussion that follows is limited to the template layout problem.

Because of the vast number of possible layout arrangements a procedure that searches for an optimal arrangement by working directly with the irregularly shaped pieces does not seem feasible. Instead one utilizes a two-step procedure in which the problem is converted from one of placing irregularly shaped pieces to one of allocating rectagonal pieces.

In the first step, the piece templates are encased in minimum-area rectangles, first singly, then in combinations of twos, threes, up to perhaps fives, or sixes. These minimum-area rectangles, called modules, will include all combinations of the pieces, including, of course, also multiples of the same piece.

The process of generating a minimum-area encasing rectangle is illustrated in Figure 25. Given an irregular template, one first encodes it using the chain-coding scheme. Next

one encloses it in a rectangle with sides parallel to the x and y axes. This is accomplished simply by finding the chain nodes for maximum and minimum x, as well as maximum and minimum y, using Equations (6) and (7) (p. 76). In Figure 25(a), these four extremal points are the points marked 1, 2, 3, and 4.

Next one forms a four-sided polygon by connecting successive extremal points by means of straight lines, and also divides the chain into segments at these points. A search is now made of each of the four chain segments to locate the chain node maximally distant (in the outward-directed sense only) from the corresponding quadrilateral edge. The technique used is the same as that described earlier in connection with Figure 20 (p. 83). For the example here, one thus finds point 5 for the segment 1-2, and point 6 for the segment 2-3. The maximally outward point for the segment 1-4 is judged to lie within specified tolerance of the line segment and need not be considered.

A new polygon is now formed, incorporating the newly found nodes. This is the polygon 1-5-2-6-3-4-1 in Figure 25(a). The process of looking for new nodes is repeated, but, of course, only for the newly formed chain segments (i.e., 1-5, 5-2, 2-6, and 6-3). If a maximally outward node lies within a specified tolerance of the line segment, no new polygon vertex is formed. In the example here, this is judged to be the case for chain segments 2-6 and 6-3. The process is repeated until no further vertices are generated. The final polygon can be regarded—within the specified tolerance—to be the *convex hull* of the template.

Determining the minimum-area encasing rectangle is now a simple matter since at least one of its sides must lie along an edge of the convex hull. Using each of the polygon's vertices in succession as the origin of a Cartesian coordinate system with x axis along a polygon edge, one determines the extremal nodes of the chain in both x and y directions and uses these to form rectangles. The areas of the rectangles are then calculated and the rectangle with minimum area is selected. For the template of Figure 25(a), the minimum-area rectangle is found when the origin is at point 4 and the x axis along 4-3. The result is shown in Figure 25(b).

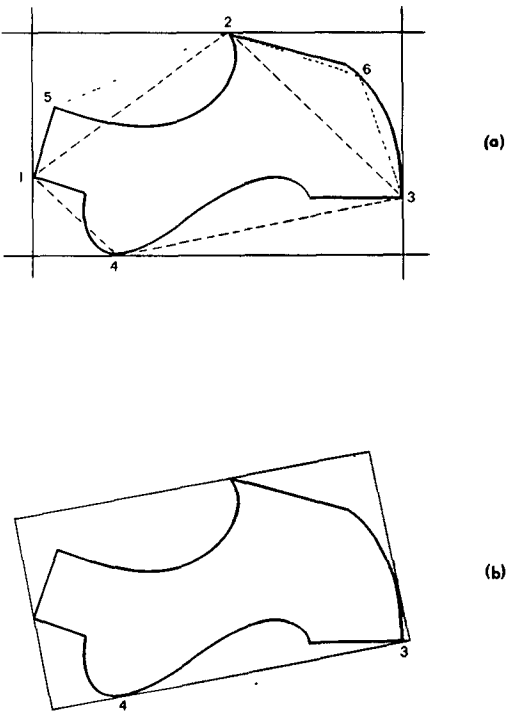
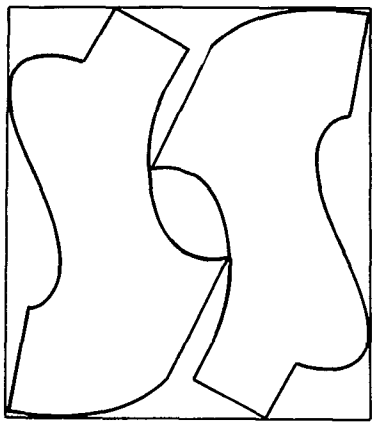


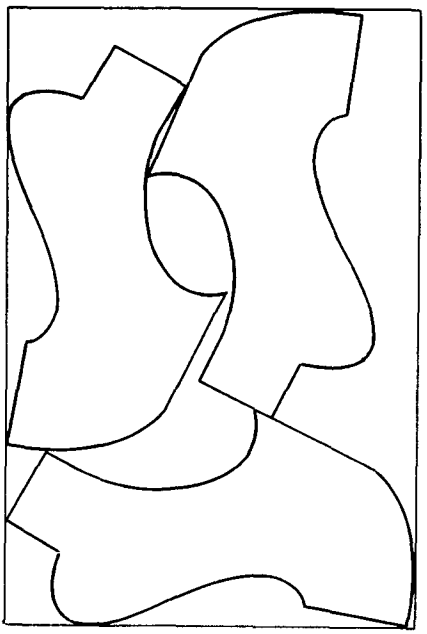
FIG 25 Determination of minimum-area rectangle

To obtain minimum-area encasing rectangles for two or more pieces, one has the additional problem of placing the pieces in the best relative positions. The approach taken is first to represent the piece contours by means of polygonal approximations [92, 109] and to obtain the best internal packing by trying to fit a "peninsula" of one piece to a "bay" of another. A detailed description of the procedure, however, is beyond the scope of this paper [1, 63]. A module containing two pieces of the type of Figure 25 is shown in Figure 26(a), one containing three, in Figure 26(b).

In the second stage of the procedure, an optimal sheet layout is generated using the set of modules obtained in the first stage. The procedure will initially attempt to achieve a layout entirely with the module that contributes the highest benefit value to the objective function (e.g., the one having the highest space utilization ratio). For left-over space (near the sheet boundaries) small,



(a)



(b)

FIG 26 Two- and three-piece rectangular modules

lower-valued modules will be used, and, where appropriate, some of the large, high-valued modules may be replaced by a larger number of lower-valued ones if this contributes toward achieving a higher overall value for the objective function.

If we ignore the special situation at the sheet boundaries, the most straightforward layout is simply to stack the modules side by side, in rectangular-array fashion, as illustrated in Figure 28(a). This kind of stacking has the important property of being *spatially complete*; that is, the modules can be packed side by side without any intervening space.

It is possible to pack the modules more densely and preserve the spatial completeness property by overlapping modules in a systematic fashion. The nature and amount of overlap is, of course, governed by the template positions within the modules. Figure 27(a) and (b) show two overlap configurations for the single-template module of Figure 25(b). Version (a) represents maximum overlap and would normally be selected. For the two-piece module of Figure 26(a) the optimum overlap position is shown in Figure 27(c).

In Figure 27 the overlap is achieved by taking identical modules and positioning them so that corresponding edges are parallel. Under these conditions the overlap will be *iterable*; that is, each module overlaps one of its neighbors and is itself overlapped by the same amount. An iterated overlap layout is illustrated in Figure 28(b). If two modules of identical dimensions but different internal configurations are overlapped such that corresponding edges are parallel, or if one of two identical modules is rotated a multiple of 90° relative to the other, the result will not be iterable but will still be spatially complete, as shown in Figure 28(c). Finally, modules of different size and different internal configuration can be overlapped with spatial completeness preserved, provided edges are pairwise parallel and the two modules have one edge collinear; this is illustrated in Figure 28(d).

To determine the position of maximum overlap for two modules, one must test the top, right, bottom, and left internal piece profiles of one module respectively against the bottom, left, top, and right internal piece profiles of the other (See Figure 27.) For example, consider right-left profile testing Starting from the line $x = 0$, one computes the difference in the x coordinates between points on the left profile of the right module and the corresponding points on the right profile of the left module for each common y coordinate, as one module is shifted, point by point vertically along the full length

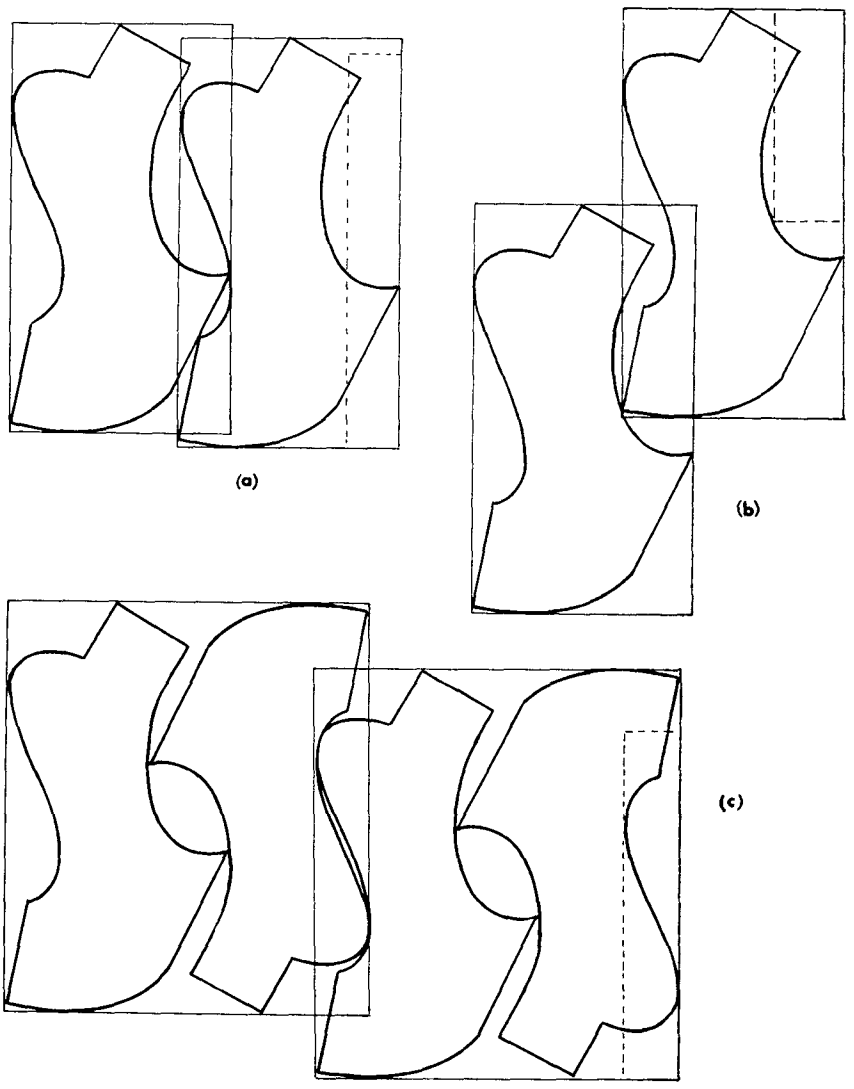


FIG 27 Iterable module overlap.

of the other. For each shift position, the minimum positive x-coordinate difference times the common y-axis range yields the permissible module overlap area for this shift position. The shift position yielding maximum area overlap is the desired one. If the piece profiles are represented in chain-coded form, the computation is quickly performed with the aid of Equations (6) and (7) (p. 76).

The efficiencies realizable in array layouts of the various modules of Figures. 25, 26, and 27 are as follows:

Module Type	*Area*	*Number of Templates*	*Utilization Ratio*
Fig 25 (a)	47.6	1	0 49
(b)	41.1	1	0.57
Fig 26 (a)	68 4	2	0 68
(b)	117	3	0 60
Fig. 27 (a)	63	2	0 74
(b)	68	2	0 69
(c)	119	4	0 78*

The template area is 23 3.
* optimum

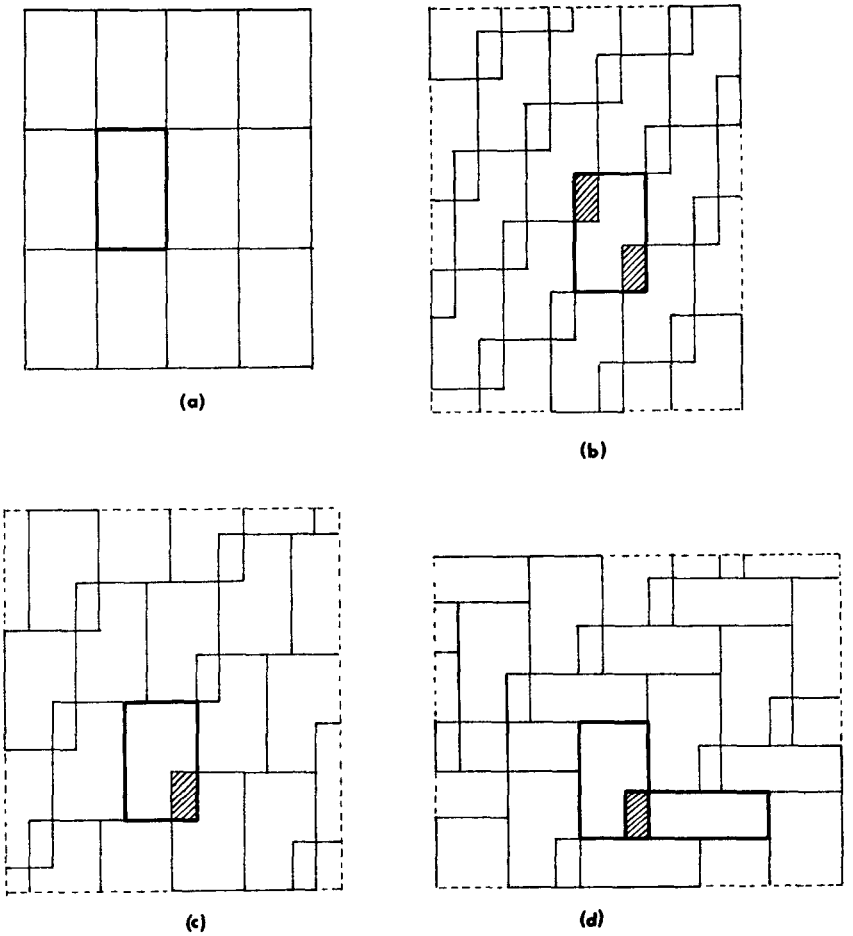


FIG 28. Spatially-complete, module array layouts.

This two-stage procedure, which has been briefly outlined here, will, of course, not result in a true optimum layout. In problems of this kind one is, in any case, rarely able to know whether a possible solution (one that satisfies all constraints) is in fact at or near optimum. An effective procedure must be able to generate a "possible" solution, verify that it satisfies all problem constraints, and then determine whether it is better (with the objective function serving as measure) than previously obtained solutions. The procedure must further be able to derive some indication from each new "best" solution as how to generate a possible still better solution; it will terminate when its ability to generate potentially better solutions becomes exhausted. The procedure will be useful if it yields results that, on average, are superior to those obtainable by other means.

5. CONCLUSION

The objective of this paper has been to provide a tutorial review of techniques for the computer processing of line drawing images. Emphasis was placed on the processing of irregular line structures because of the importance of these in aerial photography, bio-medical imagery, and other imagery drawn from nature. In contrast to the processing of line structures representing man-made objects (e.g., engineering or architectural designs), the processing of natural images in terms of line structure representa-

tions has not thus far been given adequate coverage in the literature. There is much reason to believe that the subject will become increasingly important in the future as computing costs continue to drop and the pressure to replace present manual image interpretation procedures mounts.

A variety of algorithms for processing line structures have been described. The importance of the encoding scheme selected for a particular application was stressed in relation to both efficiency of storage and facility for processing. Three application problems were discussed in some detail to demonstrate the power and usefulness of the techniques available for line structure processing.

As with any survey, some important aspects have been omitted, either for lack of space or by oversight. However, a reader desiring to become familiar with this field should find the paper a useful starting point.

REFERENCES

1. ADAMOWICZ, M., AND A. ALBANO "A two-stage solution of the cutting-stock problem," *Information Processing 71* Proc IFIP Congress 71) Freiman, North-Holland Publishing Co., Amsterdam, 1972, pp 1086–1091.
2. AMIDON, E. L., AND G. S. AKIN "Algorithmic selection of the best method for compressing map data strings," *Comm ACM*, **14**, (12), Dec 1971, pp. 769–774
3. ARCELLI, C., AND S. LEVIALDI "Parallel shrinking in three dimensions," *Comp. Graphics and Image Proc* **1**, (1), pp 21–30, April 1972
4. ARMIT, A. P. "The interactive languages of multipatch and multiobject design systems," *Computer Aided Design*, Autumn 1971, pp 10–15
5. ATTNEAVE, F., AND M. D. ARNOULT "The quantitative study of shape and pattern perception," *Psychol Bulletin*, **53**, pp 453–471, 1956.
6. BAECKER, R. M. "Picture driven animation," *Proc AFIPS*, **34**, SJCC, 1969, pp 273–288
7. BELLMAN, R., AND S. E. DREYFUS *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962
8. BENGTSSON, B. E., AND S. NORDBECK "Construction of isorithms and isorithmic maps by computers," *Nordisk Tidskrift for Informations Behandling*, Copenhagen, Denmark, 1964.
9. BLUM, H. "A transformation for extracting new descriptors of shape," *in Models for the perception of speech and visual form*, W. Wathen-Dunn, ed., M.I.T. Press, Cambridge, Massachusetts, 1967.
10. BOTTING, R. J., AND M. L. V. PITTEWAY Correspondence, comments on letter, by M. F. Partridge on pp. 119–120 of same journal, *Computer Journal*, **11**, 1968, p. 120.
11. BOUKNIGHT, W. J. "A procedure for generation of three-dimensional half-toned computer graphics presentations," *Comm. ACM*, **13**, (9), Sept. 1970, pp. 527–536.
12. BRACCHI, G., AND D. FERRARI "A language for treating geometric patterns in a two-dimensional space," *Comm. ACM*, **14**, (1), Jan. 1971, pp 26–32.
13. BRESENHAM, J. E. "Algorithm for computer control of a digital plotter," *IBM Systems J.*, **4**, pp. 25–30, 1965.
14. BRICE, C. R., AND C. L. FENNEMA "Scene analysis using regions," *Artificial Intelligence*, **1**, (1), 1970, pp. 205–226. (Also SRI Tech. note 17, Artif. Int. Grp., Menlo Park, California)
15. BURKHALTER, R. "An investigation of packing with emphasis on the 2-dimensional pattern cutters problem," Ph.D. dissertation, University of Michigan, Ann Arbor, Mich., 1964.
16. CANTONI, A. "Optimal curve fitting with piecewise linear functions," *IEEE Trans. Comp., C-20*, (1), January 1981, pp. 59–67.
17. CAPON, J. "A probabilistic model for run-length coding of pictures," *IRE Trans. on Information Theory, IT-5*, (4), pp. 157–163, December 1959.
18. CHANG, S. K. "Automated interpretation and editing of fuzzy line drawings," *Proc. AFIPS* 1971 SJCC, **38**, AFIPS Press, Montvale, N.J., pp. 393–339.
19. CHENG, G. C., AND R. S. LEDLEY "A theory of picture digitization and applications," *Pictorial pattern recognition*, Cheng, C. et al., eds., Thompson Book Co., Washington, D.C., 1968, pp 329–352.
20. CHERRY, C. *On human communication*, John Wiley and Sons, New York, 1957.
21. CHRISTENSEN, C., AND E. N. PINSON "Multifunction graphics for a large computer system," *Proc. AFIPS* 1967 FJCC, **31**, AFIPS Press, Montvale, N.J., pp. 697–711.
22. CLOWES, M. B. "Transformational grammars and the organization of pictures," in *Automatic interpretation and classification of images*, Grasselli, A. ed., Academic Press, N.Y., 1969, Ch 2.
23. COURTIEUX, G., G. FAYOLLE, F. POLICNER, AND M. BADEL "An interpreter for the interactive generation and animation of two-dimensional pictures," in *Graphic languages*, F. Nake and A. Rosenfeld, eds., North-Holland Publ. Co., Amsterdam, 1972.
24. DAVIS, C. M. "A study of the land type," Tech. Rept., ORA Project 08055, Dept. of Geography, Univ. of Michigan, Ann Arbor, Michigan, March 1969, AD 685 871 *
25. DIMOND, T. L. "Devices for reading hand-

* Copies of reports identified with an "AD" number may be purchased for a nominal fee from National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22151.

written characters," *Proc 1957 EJCC*, Washington, D.C. pp 232–237, Dec. 9–13, 1957.

26. DINNEEN, G. P "Programming pattern recognition," *Proc. Western Joint Computer Conference*, Los Angeles, pp 94–100, Mar 1955
27 DUDA, R. O. AND P E HART *Pattern classification and scene analysis*, John Wiley Interscience, New York, 1973.
28. DWYER, S. J , C. A. HARLOW, D. A. AUSHERMAN, AND G. S LODWICK "Computer diagnosis of radiographic images", *Proc. SJCC*, **40**, AFIPS Press, Montvale, N J , May 1972, pp 1027–1041.
29. ENCARNACAO, J , AND W. GILOI. "PRADIS—An advanced programming system for 3-D display," *Proc. AFIPS* Spring 1972 SJCC, **40**, AFIPS Press, 210 Summit Ave , Montvale, N J
30 EVANS, D., AND A. VAN DAM. "Data structure programming system," in *Information Processing 68* (Proceedings of IFIP Congress 1968), North-Holland Publishing Co., Amsterdam, 1969.
31. FAIMAN, M., AND J NIEVERGELT *Pertinent concepts in computer graphics*, University of Illinois Press, Urbana, Ill , 1969.
32 FEDER, J. "Languages of encoded line patterns," *Inform. and Control* **13**, (3), pp. 230–244, Sept. 1968.
33. FEDER, J., AND H FREEMAN. "Digital curve matching using a contour correlation algorithm," *IEEE Internat'l Conv. Record*, Pt. 3, pp. 69–85, 1966
34. FEDER, J "Languages, automata and classes of chain-encoded patterns," TR 400–165, New York University, August 1967. Available from U. S. Dept. of Commerce as AD 668 081 *
35 FIRSCHEIN, O., AND M A FISCHLER. "Describing and abstracting pictorial structures," *Pattern Recognition*, 3, (4), Nov. 1971, pp. 421–443
36. FIRSCHEIN, O., AND M A. FISCHLER. "A study in descriptive representation of pictorial data," *Pattern Recognition*, **4**, (4), Dec 1972, pp. 361–377.
37. FORREST, A R. "Curves for computer graphics," in *Pertinent concepts in computer graphics*, ed. by M Faiman and J. Nievergelt, University of Illinois Press, Urbana, Ill , 1969, pp. 31–47
38. FORSEN, G. E "Processing visual data with an automaton eye," *Pictorial pattern recognition*, Cheng, G. et al , eds , Thompson Book Co , Washington, D.C. 1968, pp 471–502
39. FREEMAN, H. "On the encoding of arbitrary geometric configurations," *IRE Trans. EC-10*, (2), pp. 260–268, June 1961.
40. FREEMAN, H. "Techniques for the digital computer analysis of chain-encoded arbitrary plane curves," *Proc. Natl Elect. Conf.* **17**, pp 421–432, Oct. 1961.
41. FREEMAN, H. "A technique for the classification and recognition of geometric patterns," *Proc 3rd. Intl. Congress on Cybernetics* Namur, Belgium, pp. 348–368, 1961.
42. FREEMAN, H "On the digital-computer classification of geometric line patterns," *Proc. Natl Elect Conf.* **18**, pp. 312–234, Chicago, Ill , 1962.
43. FREEMAN, H AND L. GARDER "Apictorial jigsaw puzzles· the computer solution of a problem in pattern recognition," *IEEE Trans. on Elec Comp. EC-13*, (2) April 1964, pp. 118–127
44 FREEMAN, H. AND S P MORSE. "On searching a contour map for a given terrain elevation profile," *J. Franklin Inst , 284*, (1), pp. 1–25, July 1967.
45. FREEMAN, H. "Computer methods for the processing, classifying, and matching of profiles and other irregular curves," in *Computers and Their Potential Applications in Museums*, pp 237–258, Arno Press, New York, 1968
46 FREEMAN, H , AND J. M GLASS "On the quantization of line drawing data," *IEEE Trans. Syst. Science and Cybern , SSC-5*, (1), Jan 1969, pp. 70–79.
47. FREEMAN, H "A review of relevant problems in the processing of line-drawing data," in *Automatic interpretation and classification of images*, Grasselli, A ed , Academic Press, New York, 1969, pp 155–173.
48. FREEMAN, H. "Boundary encoding and processing," *Picture processing and psychopictorics*, B Lipkin and A Rosenfeld, eds., Academic Press, Inc , New York, 1970
49. GILMORE, P C , AND GOMORY, R. F. "Multistage cutting stock problems of two or more dimensions," *Operations Res.*, **13**, January-February 1965, pp 94–120.
50. GLASS, J. "A criterion for the quantization of line-drawing data," doctoral dissertation, New York University, June 1965 Available from U. S. Dept. of Commerce as AD 621 086 *
51. GLASS, J. "Smooth-curve interpolation a generalized spline-fit procedure," *BIT*, **6**, (4), 1966, pp. 277–293.
51 GLUSS, B "A line segment curve-fitting algorithm related to optimal encoding of information," *Inform. and Control*, **5**, (3), pp 261–267, Sept. 1962
53 GRAHAM, D. N. "Image transmission by two-dimensional contour coding," *Proc. IEEE* **55**, (3), March 1967, pp 336–346.
54 GRASSELLI, A , ed *Automatic interpretation and classification of images*, Academic Press, New York, N Y , 1969.
55. GRAY, S. B "Local properties of binary images in two dimensions," *IEEE Trans. Comput* , vol. C-20, no 5, pp. 551–561, May 1971.
56. GREANIAS, E. C., P F. MEAGHER, R. J NORMAN, AND P ESSINGER "The recognition of handwritten numerals by contour analysis, *IBM J. Res and Develop* , **7**, (1), pp. 14–21, Jan. 1963
57. GREANIAS, E. C , C. J. HOPPEL, M. KLOOMOK, AND J. S OSBORNE. "Design of logic for recognition of printed characters by simulation," *IBM J Res and Develop.*, **1**, (1), pp. 8–18, Jan. 1957

58. GRIMSDALE, R L., F H SUMMER, C. J TUNIS, AND T. KILBURN. "A system for the automatic recognition of patterns," *Proc Inst Elec. Engineers*, **106**, Pt. B, (26), pp. 210–221, (England) March 1959.

59. GRONER, G. F. "Real-time recognition of hand-printed text," *Proc. AFIPS* 1966 FJCC, **29**, AFIPS Press.

60. GROSS, M., AND M. NIVAT "A command language for visualization of articulated movements," *Computer and information sciences II*, Julius T Tou, ed , Academic Press, New York, 1967, pp. 281–290.

61. GUZMAN, A. "Decomposition of a visual scene into three-dimensional bodies," *Proc* AFIPS 1968 FJCC, **33**, AFIPS, pp. 291–304.

62 GUZMAN, A "Analysis of curved line drawings using context and global information," *Machine intelligence* **6**, American Elsevier Publ. Co., New York, 1971, pp. 325–375

63 HAIMS, M J., AND H FREEMAN. "A multistage solution of the template-layout problem, "*IEEE Trans Syst Sci & Cyber SSC-6*, (2), April 1970, pp. 145–151

64. HANKLEY, W. J , AND J. T TOU. "Automatic fingerprint interpretation and classification via contextual analysis and topological coding," *Pictorial pattern recognition*, Cheng, G , et al., eds , Thompson Book Co., Washington, D. C., 1968, pp. 411–456

65 HILDITCH, C J "A system of automatic chromosome analysis," in *Automatic interpretation and classification of images*, ed. by A. Grasselli, Academic Press Inc , New York 1969, pp 363–389.

66 HODES, L "Machine processing of line drawings," Rept. 54G-0028, M I T Lincoln Lab, Lexington, Mass , 1961; AD 252 850.*

67 HOLDERMANN, F , AND H. KAZMIERCZAK "Preprocessing of gray-scale pictures," *Comp Graph and Img Proc* **1**, (1), pp. 66–80, April 1972.

68 HUECKEL, M. H. "An operator which locates edges in digitized pictures," *J. ACM*, **18**, (1), 113–125, Jan 1971

69 HUFFMAN, D. A "Impossible objects as nonsense sentences," *Machine intelligence 6*, Meltzer, B., and Michie, D , eds , Am Elsevier Publ. Co , New York, pp 295–323, 1971.

70. JARVIS, C. L "A method for fitting polygons to figure boundary data," *Australian Computer Jour* , **3**, (2), May 1971, pp. 50–54

71 KIRSCH, R A , L. CAHN, C RAY, AND G H. URBAN. "Experiments in processing pictorial information with a digital computer," *Proc EJCC*, Washington, D C., pp 221–229, 1957.

72. KIRSCH, R "Computer interpretation of English text and picture patterns," *IEEE Trans. EC-13*, (4), pp. 363–376, August 1964

73. KNOWLTON, K C "A computer technique for the production of animated movies," *Proc AFIPS* 1964 SJCC, **25**, AFIPS Press, pp 67–87.

74. KNOWLTON, K , AND LEON HARMON "Computer-produced grey scales," *Comp. Graphics and Image Proc* , **1**, (1), April 1972, 1–20.

75. KUHL, F. "Classification and recognition of hand-printed characters," *IEEE Intl Conv. Record*, Pt. 4, 1963, pp. 75–93.

76 KULSRUD, H E. "A general-purpose graphics language," *Comm. ACM*, **11**, (4), pp. 247–254, April 1968.

77. LEEUWENBERG, E. L. J. "A perceptual coding language for visual and auditory patterns," *Am J. Psychology* **84**, (3), pp. 307–350, 1971.

78 LEVI, G , AND U. MONTANARI. "A grey-weighted skeleton," *Inform. and Control*, **17**, pp. 62–91, 1970.

79. LIPKIN, B , AND A. ROSENFELD *Picture processing and psychopictorics*, Academic Press, New York, 1970.

80. LIPKIN, L. E "Resolution, scale change and information distortion," *Picture processing and psychopictorics*, Lipkin, B. and Rosenfeld A , eds , Academic Press, New York, 1970, pp 203–215.

81. LOEB, J "Communication theory of transmission of simple drawings," in *Communication theory*, Jackson, Willis, eds., Butterworths Scientific Publ Co , London, 1953, pp. 317–327.

82. LOUTREL, P. "A solution to the hidden-line problem for computer-drawn polyhedra," *IEEE Trans Comp., C-19*, (2), March 1970, pp. 205–213

83 MACH, R E , AND T. L. GARDNER "Rectification of satellite photography by digital techniques," *IBM J of Res. and Develop., 6*, July 1962

84. MARTELLI, A "Edge detection using heuristic search methods," *Computer graphics and image proc* **1**, (2), August 1972, pp. 169–182

85 MARUYAMA, K. "An approximation method for solving the sofa problem, "Tech. Rept UIUCDS-R-71-489, Dept of Comp Science, Univ. of Ill , Urbana, Ill. 61801, October 1971

86. MATSON, W. L , H. A. MCKINSTRY, G. G. JOHNSON, JR , E. W WHITE, AND R. E MCMILLAN "Computer processing of SEM images by contour analysis," *Patt. Rec* , **2**, (4), Dec. 1970, pp 303–312.

87 MAXWELL, P C. "The perception and description of line drawings by computer," *Comp Graph. and Img. Proc.*, **1**, (1), pp. 31–46, April 1972

88. MERRILL, R. D. "Representation of contours and regions for efficient computer search," *Comm ACM*, **16**, (2), 69–82, February 1973

89 MILLER, W AND A SHAW. "Linguistic Methods in Picture Processing. A survey," *Proc. AFIPS* 1968 FJCC, **33**, Pt I, AFIPS Press, pp. 279–290

90 MONTANARI, U. "Continuous skeletons from digitized images," *J. ACM*, **16**, (14), October 1969, pp 534–549

91 MONTANARI, U. "On limit properties in digitization schemes," *J. ACM*, **17**, (2), April 1970, pp 348–360.

92 MONTANARI, U. "A note on minimal length

polygon approximation," *Comm. ACM*, **13**, 1970, p. 41.

93. MORSE, S. "A mathematical model for the analysis of contour-line data," *J. ACM*, **15**, 205–220, 1968.
94. NAGAO, M "Picture recognition and data structure," in *Graphic languages*, ed by F. Nake and A. Rosenfeld, North-Holland Publishing Co , Amsterdam, 1972, pp. 48–68
95. NARASIMHAN, R "Labeling schemata and syntactic descriptions of pictures," *Inform and Cont* **7**, (2), June 1964, pp 151–179.
96 NARASIMHAN, R "On the description, generation, and recognition of classes of pictures," in *Automatic interpretation and classification of images*, Grasselli, A , ed , Academic Press, N Y , 1969, Ch 1
97 NEWMAN, W M "A system for interactive graphical programming," *Proc SJCC*, pp. 47–54, **32**, 1968, Thompson Book Co.
98 NEWMAN, W. M. "Display procedures," *Comm. ACM*, **14**, (10), pp 651–660, October 1971
99. PALMER, J. A B "An economical method of plotting contours," *Australian Computer J* , **2**, (1), February 1970, pp 27–31
100. PARKS, J. R "A multi-level system of analysis for mixed font and hand-blocked printed characters recognition," in *Automatic interpretation and classification of images*, Grasselli, A , ed , Academic Press, N Y , 1969, pp 295–322.
101. PARTRIDGE, M F. "Algorithm for drawing ellipses or hyperbolae with a digital plotter," *Computer J* , **11**, 1968, pp. 119–120.
102. PFALTZ, J. L., AND A ROSENFELD. "Computer representation of planar regions by their skeletons," *Comm ACM*, **10**, (2), February 1967, pp. 119–125
103 PHILBRICK, O "Shape description with the medial axis transformation," *Pictorial pattern recognition*, Cheng, G., et al , eds., Thompson Book Co , Washington, D C., 1968, pp 395–407.
104 PINGLE, K K., AND J. M TENENBAUM. "An accommodating edge follower," *Proc Second Int'l Joint Conf. on Artif Intell* , British Comp. Society, London, 1971, pp 1–7.
105. PITTEWAY, M L. V. "Algorithm for drawing ellipses or hyperbolae with a digital plotter," *Computer J.*, **10**, (3), 1967, pp. 282–289
106. PREWITT, J M S "Parametric and nonparametric recognition by computer. an application to leukocyte image processing," in *Advances in Computers*, **12**, 1972, Academic Press, New York, pp 285–414.
107. PRINCE, M. D. *Interactive graphics for computer-aided design*, Addison-Wesley Publ. Co , Reading, Mass , 1971.
108. RABINOWITZ, A. "Obtaining 3-D geometric descriptions of polyhedra from sets of their perspective projections," *Proc. Seminar on Pattern Recog. Studies*, New York, June 1969, pp. 55–62, **18**, Soc Photopt Instr. Eng'rs., Redondo Beach, California.
109 RAMER, U. "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Proc* , **1**, (3), November 1972, pp 244–256
110 RAUDSEPS, J. G "Some aspects of the tangent-angle vs arc length representation of contours," Tech Rept 1801-6, Air Force Avionics Lab , Wright-Patterson AFB, Ohio, available from U S. Dept of Commerce as AD 462 877 *
111. REGGIORI, G "Digital computer transformation for irregular line drawings," Tech. Rept 403-22, New York University, April 1972, available from U S Dept of Commerce as AD 745 015
112. ROBERTS, L. G "Machine perception of three-dimensional solids," Ph D Thesis, MIT, Dept. of Elec. Eng'g , February 1963, (MIT Eng'g Library).
113. ROOS, D. AND C L. MILLER. "COGO-90· Engineering user's manual," Rept R64-12, Dept of Civil Eng'g , MIT, Cambridge, Massachusetts, April 1964.
114 ROSENBERG, B "The analysis of convex blobs," *Computer Graphics and Image Processing*, **1**, (2), August 1972, pp 183–192
115 ROSENFELD, A "Picture processing by computer," *Computing Surveys*, **1**, (3), September 1969, pp. 147–176
116 ROSENFELD, A , M THURSTON, AND Y H. LEE. "Edge and curve detection. further experiments," *IEEE Trans Comp* **C-21**, (7) July 1972, pp 677–715.
117 ROSENFELD, A "Figure extraction," in *Automatic interpretation and classification of images*, Grasselli, A , ed , Academic Press, New York 1969, Ch 6
118 ROSENFELD, A *Picture processing by computer*, Academic Press, Inc , New York, 1969.
119. ROSENFELD, A. "Connectivity in digital pictures," *J. ACM*, **17**, pp 146–160, 1970
120. ROSENFELD, A AND M THURSTON. "Edge and curve detection of visual scene analysis," *IEEE Trans Comp* **C-20**, (5), pp 562–569, May 1971
121. ROSENFELD, A "Arcs and curves in digital pictures," *J ACM*, **20**, (1), January 1973, pp. 81–87
122. RUTOVITZ, D "Data structures for operations on digital images," *Pictorial pattern recognition*, Cheng, G , et al , eds , Thompson Book Co , Washington, D.C. 1968
123 RUTTENBERG, K "Digital-computer analysis of arbitrary three-dimensional geometric configurations," Tech Rept 400-69, New York University, available from U S Dept. of Commerce, NTIS, as PB 167 315
124 RUTTENBERG, K "Algorithms for the encoding of three-dimensional geometric figures," Tech. Rept. 400-86, available from U. S Dept of Commerce as AD 434 320 *
125 SAKAI, T , M NAGOA, AND S FUJIBAYASHI. "Line extraction and pattern detection in a photograph," *Pattern Recog* , **1**, (3), March 1969, pp. 233–248
126 SAKAI, T., M NAGAO, AND H MATSUSHIMA. "Extraction of invariant picture sub-struc-

tures by computer," *Comp Graph. and Img. Proc* **1**, (1), pp 81–96, April 1972.

127. SCHREIBER, W. F., T S. HUANG, AND O J. TRETIAK "Contour coding of images," in *Picture bandwidth compression*, Huang, T S. and Tretiak, O. J , ed , Gordon and Breach, publ , New York 1972, pp. 443-448.
128 SCHWINN, P M "A problem-oriented graphic language," *Proc ACM Nat'l Meeting* 1967, ACM, New York, pp. 471-476
129 SELFRIDGE, O "Pattern Recognition and Modern Computers," *Proc Western Joint Computer Conf* , Los Angeles, March 1955
130 SHERMAN, H "A quasi-topological method for the recognition of line patterns," *Proc Int'l Conf on Information Processing*, UNESCO, Paris, 15-20 June 1959, pp 232–238 (paper also available as M I T Lincoln Lab Rept , Group Rept 2G-25-17, May 1959, Lexington, Massachusetts)
131 SIDHU, G S , AND R T BOUTE. "Property encoding applications in binary picture encoding and boundary following," *IEEE Trans Comp* , **C-21**, (11) 1206-1216, November 1972.
132 SKLANSKY, J "Recognition of convex blobs," *Pattern Recognition*, **2**, (1), January 1970, pp 3-10
133. SMITH, D N. "GPL/1 - A PL/I extension for computer graphics," *Proc AFIPS* Spring 1971 SJCC, **38**, AFIPS Press, Montvale, N, J pp 511-528.
134. STALLINGS, W. "Recognition of printed chinese characters by automatic pattern analysis," *Comp Graph and Img Proc* , **1**, (1), pp 47–56, April 1972, (see also. Proc SJCC 40, May 1972)
135 STOCKHAM, T G "Natural image information compression with a quantitative error model," in *Pertinent concepts in computer graphics*, Faiman, M. and Nievergelt, J eds , Univ Illinois Press, Urbana, Illinois, 1969, pp 67-86.
136. STONE, S P , J L LITTLEPAGE, AND B R CLEGG "Second report on the chromosome scanning program at the Lawrence Radiation Laboratory," *Proc Pattern Recognition seminar*, Soc Photo Instr Eng'rs , **18**, June 1969, pp 157-171
137 SUTHERLAND, I E "Sketchpad a man-machine graphical communication system," *Proc SJCC*, **23**, pp 329–346, Spartan Books, Baltimore, Maryland, May 1963
138. TAKASAWA, Y , S MORIGUCHI, AND T. SAKAMAKI "A graphics manipulating language," in *Graphic languages*, ed. by F Nake and A Rosenfeld, North-Holland Publ Co , Amsterdam, 1972, pp. 327-333.
139. TALBOT, P. A , J. W CAR III, R R COULTER JR , AND R C. HUANG. "Animator: an on-line two-dimensional film animation system," *Comm. ACM*, **14**, (4), pp. 251–259, April 1971.
140. TIPPITT, J., et al. *Optical and electro-optical information processing*, M.I.T. Press, Cambridge, Massachusetts, 1965.
141 TOBLER, W. R. "Problems and prospects in geographical photointerpretation," *Pictorial pattern recognition*, Cheng, G , et al, eds., Thompson Book Co., Washington, D.C. 1968, pp. 267–273
142. TRIENDL, E E "Skeletonization of noisy handdrawn symbols using parallel operations," *Pattern Recog.* **2**, (3), Sept. 1970, pp 215–226.
143. UHR, L "Machine perception of printed and handwritten forms by means of procedures for assessing and recognizing gestalts," *Proc ACM Nat'l Meeting 1959*, ACM, New York
144 UHR, L , ed *Pattern recognition*, John Wiley & Sons, Inc. New York, 1966.
145 UNGER, S H. "Pattern Detection and Recognition," *Proc. IRE*, **47**, (10), pp. 1737–1752, October 1959
146 WATTS, T L. "Scanning and measuring photographs of bubble chamber tracks using a computer controlled line segment (PEPR), in *Pictorial pattern recognition*, Cheng, G., et al, eds , Thompson Book Co., Washington, D C. 1968, pp 207–220.
147. WHITE, H. S. "DAPR—digital automatic pattern recognition for bubble chambers," in *Pictorial pattern recognition*, Cheng, G., et al , eds , Thompson Book Co , Washington, D C. 1968, pp 175–198.
148. WHOLEY, J S. "The coding of pictorial data," *IRE Trans on Info Theory*, **IT-7**, (2), pp 99–104, April 1961.
149 WILLIAMS, R. "A general purpose graphical language," in *Graphic Languages*, ed. by F Nake and A Rosenfeld, North-Holland Publishing Co , Amsterdam, 1972, pp 334–353.
150. WINSTON, J. S. "The operational use of meteorological satellite data," *Annals of the New York Acad. of Sciences*, **93**, (19), pp. 775–812, October 1962.
151 WOON, P., AND H FREEMAN. "A procedure for generating visible-line projections of solids bounded by quadric surfaces," in *Information processing 71* (Proc. IFIP Congress 71), North-Holland Publ. Co., Amsterdam, 1972, pp 1120–1125.
152. ZAHN, C T. "Two-dimensional pattern description and recognition via curvature points," SLAC Rept No 70, Stanford Linear Accelerator Center, Stanford Univ., Stanford, California, December 1966