Outline Coding of Typographical Characters

William F. Schreiber

1. Abstract

Recently, Compugraphic Corporation settled a suit brought by Information International, Inc., for infringement of US Patent Re. 30,679. Several years previously, it was rumored that Merganthaler had settled a similar suit. This patent, a reissue of USP 4,029,947, of which G.W.Evans and R.L.Caswell are the inventors, was originally assigned to Rockwell International, and was conveyed to III when the latter purchased Rockwell's typesetting division. It is known in the US as the "Rockwell" patent. The patent describes a character generation system for typographic information encoded by means of approximations to character outlines using segments of straight lines and circular arcs.

Since many current typographical coding systems are based on outline approximation and coding, there is a great deal of interest in the consequences of this lawsuit for the typesetting industry. The purposes of this paper are to describe the Rockwell system and to present a history of outline coding. The paper does not discuss the merits of any case, past or future. In particular, it does not attempt to analyze just what aspects of outline coding are covered by the patent, nor does it discuss the possible invalidity of the claims by reason of anticipation or obviousness. I have relied throughout on information gained from my own research in this field, and from public sources. No material unavailable to the public was used in the preparation of this paper. The opinions expressed are those of the author alone.

2. Brief History of the Rockwell Patent

On 11 May 1973, separate applications were filed by Evans and Caswell with similar specifications but different claims. One was to cover the basic concept (Caswell) and the other the specific implementation (at first, Evans, but later, both authors). The joint application issued as USP 4,029,947 in 1977, but the sole Caswell application was finally abandoned in 1982. On 14 June 1979, an application was filed to reissue the joint patent, and this become Reissue 30,679 in 1981. (Rockwell sued Eltra, the owner of Merganthaler, immediately after the reissue.) There are 47 claims in the original patent and 99 in the reissue, which includes nearly all the first 47. The file history is exceedingly complicated, and in some places rather murky. A feature is the large number of examiners who were involved, and the many times all the claims were rejected.

3. Patent Law

It is obviously presumptuous for a person who is not a patent attorney, such as the author, to summarize the patent law. However, some understanding of its provisions is necessary in order to follow the rationale of this paper. I have therefore attempted to give a highly simplified summary in what follows. The reader should get competent advice before proceeding on the basis of this outline.

E15-387 Massachusetts Institute of Technology, Cambridge, Mass. 02139

The patent system of the US is provided for in Article 1, Section 8 of the Constitution and is defined in Title 35 of the United States Code. It grants the inventor a 17-year period of exclusive rights to make, sell, and use an invention in return for teaching the public how to use it. The rationale is that this arrangement, as an alternative to secrecy, will tend to promote the development of "science and useful arts" and therefore will be of benefit to the public. To be patentable, an invention must be "new and useful" and must not be "obvious" to a "person having ordinary skill in the art" (whom I shall call a "poska") at the time of the invention. All these term have complicated legal definitions. Briefly, "new" means not known to the public, for example by an article published more than one year before the filing date. "Useful" generally means that the invention must work, at least in a primitive manner. "Obvious" is a matter of judgment, and in addition depends on the state of the art at the time of the invention. A "poska" must be defined in each case, but is typically a person with a certain education and a few years of relevant experience.

To obtain a patent, the inventor files an application consisting of a specification, or description of the invention, together with a series of claims setting forth the precise boundaries of what is desired to be protected. There is no standard form for the specification, and indeed, they vary tremendously. Usually, there is a statement of the background of the invention, a history of relevant work ("prior art") in the field, a brief summary of the main ideas, a set of drawings, and a detailed description of at least one embodiment of the invention. The embodiment is required to disclose the best way to practice the invention. Both methods and apparatus may be patented, but not "ideas." For example, the *idea* of compressing data by a code would not of itself be patentable.

Unlike the specification, there are many rules about how claims must be written, most of these having developed in practice in order to make the meaning as definite as possible. It takes a great deal of experience to read claims accurately. Whatever is claimed must be supported in the specification.

After the application is filed, it usually is passed back and forth between the inventor and the patent examiner a number of times, over a period of several years, as they negotiate as to what is to be covered in the claims. Very often, all the claims are rejected on the first office action on the basis of obviousness and/or anticipation by prior art. The inventor then can argue why his claims are valid, in the process usually modifying and sometimes rewriting them entirely. The specification can also be modified for clarity (but usually is not) but not by adding "new matter." The inventor can speak to the examiner on the telephone and may also have an interview if the examiner agrees. In such cases, a (nonverbatim) summary is placed in the file. All the communications between inventor and examiner constitute the "file wrapper" or file history, which is available to the public after issuance of the patent.

The inventor or his patent attorney normally makes a search before filing an application, but there is no legal obligation to do so. By law, the applicant *is* required to tell the Patent and Trademark Office of any relevant prior art discovered up to the date of issue. The examiner always looks for prior art, mainly by searching the files of the PTO, which consist mostly of patents. It is the author's opinion that many patents are issued improperly because the

examiners cannot really be experts (i.e., they do not, of their own knowledge, know a great deal about current research) in the various fields that each of them covers. This would seem to be important in fields in which there is a great deal of academic research, which is often not patented. In addition, it would seem very difficult to make a judgment about obviousness without personal experience in the particular field.

The inventor wants claims as broad as possible, but the examiner tries to make them as narrow as possible to avoid anticipation by prior art that turns up during the examination. The meaning of any allowed claim, for purposes of determining infringement, must take into account the specification and the file wrapper. Any statement made by the inventor in order to convince the examiner to allow a claim effectively modifies its meaning, and can be used by the defendant in an infringement proceeding. For this reason, in many cases it is impossible to interpret a claim accurately simply by reading the patent.

To infringe literally, a product or process must incorporate *every* element listed in the claim. Therefore, the shorter the claim and the fewer the limiting elements, the stronger it is. Literal infringement occurs where the suspect system accomplishes the same result in exactly the same way. Under the "doctrine of equivalence," infringement also occurs when the differences are of a nonessential kind that might be made by a poska as a matter of judgment or engineering choice. Equivalence is a two-edged sword, in that a claim can be deemed invalid, not only for literal anticipation, but also if the prior art is equivalent in the same sense as is used to charge infringement. In cases where some elements of a claim are found in one prior art reference and some in another, the combination must be obvious for invalidity. Where the various elements are found uncombined in a single reference, it is not clear whether the anticipation is literal or one must rely on equivalence. All these complexities show why patent problems sometimes end up in litigation and why it is difficult to predict the outcome.

4. Outline Coding

The importance of outlines in conveying visual meaning in reproductions, even though they do not usually appear in nature, has evidently been known since prehistoric times. The earliest cave drawings, such as those at Lascaux, are of this nature. Possibly this is related to some fundamental property of human vision, since both children and primitive people tend first to produce outline drawings. Only after instruction does the beginning "artist" learn to fill in the areas inside boundaries to make a more literal representation of nature.

4.1 Nontypographical Applications

The need to represent and/or generate curved lines occurs in many fields such as ship and airplane design.[1] Numerically controlled machine tools must guide a cutting tool along a predetermined 3-dimensional path. In the development that began in the MIT Servomechanisms Lab in 1949, these paths were approximated by a series of short straight lines, the data for which was

[1]The Ikarus typesetting system is said to have developed from such design problems.

digitally stored on punched paper tape. [1] Calculated paths were checked out by driving pen plotters and cathode-ray tube displays before attempting to cut metal. Sometimes the tools were demonstrated by cutting typographical characters in metal. [2] The same display equipment was later used for some of the earliest work in computer graphics, where the movement of the electron beam across the crt screen to create a desired graphic image is analogous to moving a tool to cut a desired shape in a block of metal. [3]

Contours and the need to code them also occur in continuous-tone images. In my own work in this field, beginning in the early fifties, images are represented by a 2-dimensional low-frequency component plus outlines from which the high-frequency component can be reconstructed. [4] The transmission of outline information is an inherent part of these systems. In this connection, considerable work was devoted to outline coding by my students at MIT from about 1960 to 1970. [5] Fig. 1 shows the letter "B" coded by Pan [6] as a series of straight line segments, simply to demonstrate his algorithm, which was not directly primarily at characters. This work was summarized in a 1968 paper by myself and two colleagues, in which an entire hierarchy of successively more sophisticated and more efficient codes are discussed, including approximating contours by higher-order curves. A numerical example of the application of such a code to Roman characters is given. [7]

Optical character recognition (OCR) is another field that has seen the use of outline coding. In Rabinow's 1962 system, [8] outlines are traced during scanning as a pair of intersections of character outline segments with scan lines. Each pair begins at a local maximum and ends at a local minimum, as shown in Fig. 2. This procedure has relevance to later type-coding systems because each closed contour is divided into a number of contour sections, each one of which is a single-valued function of the variable that extends perpendicular to the scan direction. That means that each contour section starts on one scan line and is found on all successive scan lines, once only, until it ends on some last scan line. Characterization of outlines for OCR purposes in terms of local extrema was later pursued by a number of PhD students in our MIT group, including Marston [9] and Seitz. [10]

Outline coding has been exploited extensively in facsimile, beginning with work by Laemmel [11] and Elias, [12] and later applied to weather map transmission by Huang. [13] These systems exploit the 2-dimensional nature of images outlines by locating edge points on one scan line relative to corresponding points on the preceding line. A system based on the same concepts is now the CCITT international standard for digital facsimile coding. [14]

It should be pointed out that Laemmel's report, although it seems to have escaped the attention of many designers of type-coding systems, was widely circulated among those doing bandwidth-reduction research at the time, and is still fairly well known in that group. It quoted Elias's thesis. Laemmel discusses the savings that can be made when the edge positions on one line are predicted from those on the previous one or more lines. Elias discusses predicting edge point locations using both first (straight line) and second (circular arc) differences. When there is a run of equal differences on successive lines, it is at least implied that this group of differences can be transmitted in

toto, which amounts to fitting line segments to the outline.

4.2 Binary Images

Images in general have a continuous range of grey values. In the particular case of images having only two levels, such as black and white, the outlines (boundaries) of the black areas permit exact reproduction using a very simple transformation. This is so obvious that it often is not mentioned. For example, in a typography book from the 16th century, shown in Fig. 3, the generation of characters from straight-line segments and circular arcs is described in some detail.[2]

4.3 Digital Images

In much modern equipment, images are spatially digitized to permit processing by computers and other digital equipment. The more samples (picture elements, pels, or pixels) per unit area, the more the images appear to be spatially continuous. For high quality typesetting, sampling densities of 1000 per inch or more are common, with each character being defined in a grid typically between 256 and 2048 square. Lower resolutions are often encountered in office equipment and word-processing systems, but the principles of operation of typesetting systems do not depend on their resolution.

Considering the wide range of type shapes and sizes needed for typical printing jobs, an enormous amount of on-line data storage would be required if each character were to be represented in its "natural" form, using one binary digit (bit) for each pel of, say, a 1024^2 grid. This is undesirable, not only because of the cost of storage, but of the time taken to retrieve the data while creating the character images at high speed. For this reason, a great deal of effort has gone into data compression systems. The methods that have been used at various times have been influenced principally by the cost and availability of digital logic, memory circuits, and computer systems. The ingenuity demonstrated in developing new typesetting systems has been more in choosing the appropriate components rather than in devising new coding methods, since all of the coding principles used so far have been known for many years. This is not to say that system designers always know the history of character coding - in the course of investigating this subject I found a surprising number of different individuals who believed they were the first to think of coding characters by their outlines!

4.4 One-Dimensional Coding of Binary Digital Images

Historically,[3] 1-d coding preceded 2-d coding, even though it exploits less of the redundancy in the image, because it is easier, and, in particular, requires less storage. Since there are fewer edge points[4] per line than samples,

---

[2]This reference was called to my attention by Kenneth Brown.

[3]This coding is used, for example, in the weaving instructions for oriental rugs, and, more recently, was used in the 20's for transatlantic facsimile coding.

[4]Edge points are defined as points at which the image changes from black to white or white to black. These can also be thought of as points where the original continuous outline intersects the scan line.

it is better to use just the positions of such edge points, which is readily recognized as a kind of outline coding. Note that in 1-d systems, the edge points, although they define the character outline completely, are not necessarily contiguous. Moreover, as shown in Fig. 4, if the coding is done along columns rather than rows, a different set of edge points is found, even though the same image is described. It turns out to be more efficient to code the distance between edge points along each line, i.e., the "runs" of black and white, rather than their absolute position. There is an extensive literature on run-length coding. [15] Both fixed and variable length codes are used, with the most efficient systems using a form of truncated Shannon-Fano or Huffman codes.

Run-length coded data is particularly easy to decode "on the fly" as the image is being displayed. All that is required is count clock pulses during the scan and to compare this count with the next edge point, switching between white and black each time an edge point is encountered. Old as this scheme is, it is described in detail, with a straight face and without reference to the prior art, in many of the patents cited here.

4.5 Two-Dimensional Coding

Since there is about as much vertical as horizontal redundancy in images, one-dimensional codes cannot be as efficient as 2-d codes. Some of the latter are extensions of 1-d codes in which edge points on one line are located relative to the corresponding edge point of the previous line. In the CCITT code referred to above, if the distance between edge points on successive lines is below a threshold, it is assumed that they correspond and an incremental code is used. If the distance is greater, the current point is assumed not to correspond and it is located relative to the preceding edge point on the same line.

A system of this kind that formed the basis for a commercial typesetting system is that of Manber. [16] This is a constant-length 1-d run-length coding system with some features that take advantage of line-to-line redundancy. The basic idea is that edge points inherited from the previous line may be replaced, continued for a given number of lines, or incremented by a certain amount. Strings of line-to-line increments of the same sign may omit the sign. When the same edge position is found on a string of successive lines, this is coded effectively as a single straight-line segment perpendicular to the scan lines. Claim 8 seems to indicate fitting sloped lines to the outline, as well, in that any previous data, such as an increment, which is really a slope, may be repeated any number of times.

In Manber's system, the code consists of data (edge positions, increments thereof, the number of lines an item is to be repeated, etc.) plus commands that tell how the data is to be used. Each code group is 4 bits in length, one bit being used to indicate whether it is data or instructions.

Other codes are thoroughly 2-dimensional in that they trace each contour from start to finish. One way to do this is by chain codes. For such codes, the outline is a contiguous set of points. Therefore each outline point must lie in

just one of 8 directions from the previous point, requiring at the most 3 bits.[5] This procedure requires keeping the entire image in quickly accessible storage, and in addition, does not exploit much of the redundancy of the string of direction codes. For example, if a section of outline is actually straight, many successive codes words are identical, a situation that always indicates that more efficient coding is possible.

4.5.1 Spline Coding of Contours

Splines are polynomials used as interpolation functions. [17] Linear, quadratic, and cubic splines are most common. Depending on the degree of the spline and the number of data points to be fit, one or more derivatives may be made continuous at the joins (sometimes called knots) of the successive spline segments, resulting in a very smooth interpolation. In contour coding, splines are used primarily because it may take less data to define the spline segment than the contour segment to which it is fit. Furthermore, spline fitting lends itself to "lossy" coding, which may increase the efficiency without significantly degrading the characters.

In lossy spline fitting, the starting point of a contour is found and a spline begun, using some type of error criterion. The spline is then extended point by point until the criterion is exceeded, at which point a new spline is started. The data recorded are the parameters of each spline segment. An example of this is given by Pan [18] for straight lines. Spline fitting was also developed at Bell Laboratories, specifically for typography. Mathews et al give the codes for 3 fonts of characters approximated in this manner. [19] Mathews and McDonald [20] and Frank [21] discuss a typographical coding system in which straight lines and parabolas are used to approximate character outlines.

The most sophisticated spline coding system that I know of was done by my student Coueignoux, using straight lines, circular arcs, and cubic splines. [22] This system was specifically designed as a model of a practical typesetting system. Coding and decoding were done in the course of raster scanning, with no more than two lines of data held in memory at one time. All contour portions were one-way curves, as in Rabinow. [23] Output was to a raster display device. Very high typographical quality was achieved at about 200 bits/character. No patent was applied for because, at the time, both Coueignoux and myself felt that the work was mainly an implementation of the proposals in the 1968 paper. [24] In retrospect, it appears that he could have gotten significant claims to certain aspects of the implementation, which is actually very clever.

4.5.2 Utah Computer Graphics Developments

A series of developments that began with Ivan Sutherland's *Sketchpad* work at MIT [25] and was continued later at the University of Utah and the Evans and Sutherland Computer Corporation had, as its purpose, the generation of realistic images on crt's from computer data. The general idea is to represent the image by a series of overlapping polygons, each one of a defined

[5] Since all directions are not equally probable, a more compact code is usually possible. [5]

brightness. Obviously, typographical characters are a special case of such images, in which there are only two brightnesses and in which the character outlines are approximated by a series of straight-line segments, often called vectors. The general idea of the system was first given in Wylie et al [26] and the complete system was the subject of the Erdahl and Evans[6] patent. [27] In the method as disclosed in the patent, the display is on a raster-scanned crt. The basic information consists of outlines approximated by a series of straight line segments, the latter defined by their starting points, slopes, and lengths ("extents") in terms of number of scan lines in which each segment is active. The active boundary definitions for each scan line are kept in a list, sorted in order of appearance along the line. A computation cycle is performed for each scan line, in which new boundary segments are added as needed, expired ones removed, edge points updated, and video data for each line generated by suitably changing a register each time an edge is encountered, a standard method of decoding run-length coded data.

An important feature of the Utah system, especially as potentially prior art relative to the Rockwell patent, is the character code. Input is in the form of 108-bit words, which may be either "instructions" or "data." The latter are used primarily for image sections having grey scale and are not relevant to the typesetting application. The instructions, proper, contain the starting point, slope, and extent of each vector, together with (irrelevant here) intensity gradients, all properly sorted so as to facilitate decoding.

4.5.3 The Seaco Typesetter

In two successive issues of the *Seybold Report* in 1972, a typesetter made, demonstrated, and apparently sold by the short-lived Seaco Corporation was described. [28] This system was never the subject of a technical article or patent application, so that complete details are not readily accessible to the public. In a seminar held by by a Boston law firm[7] further information was given, evidently obtained from individuals involved in the company. In any event, it is clear that this was another machine using outlines approximated by straight-line segments, or vectors. The vectors were specified by their starting and ending points. An active vector list was maintained, with the contours encountered on each scan line sorted in order of appearance. Video was generated by a standard run-length decoder.

The organization of the character code was quite sophisticated (similar to Coueignoux) resulting in a very high decoding rate with a small minicomputer, aided by a high-speed fixed-head disk. Outlines were divided into one-way curves (outline sections), and the points along each section (end points of the vectors) listed in order. A "list of lists" was kept, which pointed to the starting memory locations of the outline sections, in the order needed for decoding. The active vector list was precomputed, and the points in the character at which the list changed were part of the character code.

---

[6]Not the same Evans as in Evans and Caswell!

[7]Cesari and McKenna, 1984.

Unlike most other systems, substantial information was given as to how the characters were encoded to achieve good quality. In fact, there seems to have been no effort made by Seaco to keep any aspect of its system secret.

5. The "Rockwell" System

The Evans and Caswell patent, which presumably was the basis of the MGD *Metroset* typesetter, appears to claim only character generation, i.e., decoding. However, the decoding method clearly depends on the encoding method. The character code is given, and I shall therefore describe the entire system. What is not described in the patent is the precise means by which real characters are encoded. In any system of approximation, the typographical quality obtained with a given amount of data depends very much on the details of the encoding process, and not only on the structure of the code. It is quite possible for manufacturers to keep these details secret, selling only coded fonts and typesetters. Since the patent law requires disclosure of the best method of practicing the invention, it is not surprising that such details have generally not been patented, but have been held as trade secrets.

Each character is coded in a "normalized quad," typically 1024x1024, regardless of the size to be output or the scanning density of the output device. The outlines are approximated by straight-line segments and circular arcs, the data for which is stored as a series of "instructions." The outlines are divided into sections, in each of which y, the distance along each vertical scan line, is a single-valued function of x, the column count in the quad. One computation cycle is performed per column of the quad, during which the y values of each active contour are updated. Each contour is identified by number and each instruction contains an identification of the appertaining contour, the y starting point, the slope (and curvature, if used) of each new contour, the number of computation cycles to the next set of instructions, and the instruction type. The main types of instructions are:

BOLP: beginning of outline pair
EOLP: end of outline pair
CK: curvature increment
CM: slope increment
EC: end of character
CDY: long straight line

In the decoding process, instructions are retrieved in groups by keeping track of the number of computation cycles, each one corresponding to one column of the master quad. An active contour list is maintained, adding to it new contour pairs signaled by BOLP and deleting pairs signaled by EOLP. The new instructions may also adjust the slope and curvature of continuing contours, which also have their y values updated. Run-length coded "stroking" data is generated to produce video by a standard run-length decoder. Provision is made for scaling characters to the desired size in terms of horizontal and vertical scale factors that take account of the scan density of the display crt and the resolution of the master quad. In the scan direction, scaling is performed by adjusting the clock associated with the run-length decoder, effectively eliminating pels in the vertical directions. In the horizontal direction, strokes are deleted as required. These processes are now called decimation, i.e., the

scaling is down only, by elimination of image data.[8]

6. Analysis

We can see from the foregoing that the Rockwell typesetting system, as described in the specification of the patent, shares many features in common with prior coding and typesetting systems. These include:

Digital raster display
Encoding of characters in a master grid
Representing characters by outlines
Approximating outlines by straight lines and circular arcs
Dividing outlines into single-valued functions
Dividing coded data into "instructions"
Ordering instructions as needed for decoding
Identifying instructions as to type
Retrieving instructions as needed
Starting new outlines in pairs
Use of active segment list
Use of computation cycles to update active vector list and y values
Run-length coding/decoding to generate video

On the other hand, there are several characteristics of the Rockwell system that appear to be unique. One is the identification of instructions as to the appertaining contour. Of course, correct decoding requires this knowledge, but all the other systems that I know about simply keep the instructions in the correct order, so that the contour identification is implicit. Another apparently unique element is the manner of indicating "extent," the number of scan lines over which a segment or "boundary definition" is active. Referring to Fig. 5, where we show such a segment, it is clear that a straightforward way to specify the entire segment is by giving the coordinates of the end points. (If it is a circular arc, the radius of curvature is also needed.) One could also give the first point and the horizontal and vertical lengths, called delta x and delta y, or some other combination of 4 of these 6 numbers. (In some systems, the slope, m = (delta x)/(delta y) is used.) All of the other systems discussed here use this technique. In the Rockwell system, segments start with one instruction and end with another. If there are intervening instructions, the extent of any one segment is found by some simple algebraic operations on the "number of computation cycles" entry. This is a unique and seemingly excessively complicated way of transmitting a piece of information that is, of course, required, but which can be conveyed, and is conveyed in all the other systems that I have examined, in a simpler manner.

Another unique feature is the method of scaling, down only, by decimation. In the Bell Laboratories system, the MIT machine tool system, Seaco, and Coueignoux, scaling is up or down by means of scaling the segment (vector or curved segment) data. (Manber and Utah do not discuss scaling.) It is hard to see what advantage the Rockwell method of scaling has, beyond

[8]The word decimation comes from the Latin, and originally meant the "scaling" of captured armies by killing every tenth man. It is an old method.

conceptual simplicity. Especially for small characters, a lot of computation is done to generate lines of data that are then deleted. In addition, one would think that the quality would be lower. The patent does not discuss up-scaling.[9]

No lawyer or technical expert can safely predict the outcome of litigation, especially when a jury of laymen must decide a case involving complicated technology. The meaning of the claims in the patent have never been adjudicated, although they appear to speak only of character generating methods and apparatus. One would think that a decoder for one particular encoder would not cover a decoder for some other encoder, but this may not be a safe assumption.

A more fundamental question goes to the original intent of the founding fathers and the more recent authors of the enabling legislation that defines the patent system. If they intended to protect nonobvious advances in the practical arts, then straightforward engineering solutions to a given problem should not be protected. For example, if I ask a class of MIT students to design a system to decode contour-coded characters, and I define the code, most of them would be able to do that.[10] Out of 10 or 20 students, I would expect at least a few rather clever solutions, which could be built and which would work. Are these solutions worthy of patenting? If they were, would the issued patents cover the solutions the other students produced?

One can think of a different legal scheme in which infringement would occur if the offending product operated on the same general principles, or was otherwise similar to any prior commercial device on which some company had invested some minimum amount of money. This might prevent "rip-offs" of various products, such as occur frequently in certain fields such as appliance and clothing manufacture. It certainly is not the sytem we have now, where only patented "inventions" are protected. Of course, such a system would have its own problems of adjudication. Alternatively, the copyright laws can be used to prevent outright copying, but this is not very helpful in technical products.

9. Conclusion

In this paper, I have discussed outline coding, giving some of the history and describing some of the principal systems that have been developed for typography and other applications, including the Rockwell system. While giving the cautions appropriate to any statement about the outcome of potential lawsuits, especially those involving juries, my own conclusion is that much of the Rockwell system can be found in the prior art. It therefore ought not to cover outline coding in general, but only the particular kind of coding and decoding described in the specification and particularly spelled out in the claims. While it appears to be a valid engineering solution to a practical problem, it does not appear to represent any conceptual advance over the

[9]An analogous up-scaling method would be to repeat strokes as needed, but apparently that was not contemplated in the Rockwell system.

[10]I actually performed such an experiment. I asked a student to write such a program, using references no more recent that 1973. He had no trouble at all in doing so, first in Basic, and later in "C."

multiplicity of other systems that have been developed for the same purpose.

References
(Filing dates shown for patents)

1. W. Pease, "An Automatic Machine Tool," *Scientific American*, 187,3, 9/52, pp 101-115.
J.Forrester et al, "Numerical Control Servo-System," USP 3,069,608, 8/52.
2. MIT Servomechanisms Lab, Open House Demonstration 5/3/52.
3. J.Ward, "Automatic Programming of Numerically Controlled Machine Tools," Report 6873-FR-3, MIT Electronic Systems Lab., 1/15/60. See p 24-25.
4. W.Schreiber et al, "Synthetic Highs, an Experimental TV Bandwidth Reduction System," SMPTE J., 1959, pp 525 ff.
5. D.Graham, "Image Transmission by Two-Dimensional Contour Coding," 1966; G. Walpert, "Image Bandwidth Compression by Detection and Coding of Contours," 1970, both Ph.D. Theses, MIT Dept. of Elec. Eng.
6. J.Pan, "Reduction of Information Redundancy in Pictures," Sc.D. Thesis, MIT Dept. of Elec. Eng., 9/62.
7. W.Schreiber et al, "Contour Coding of Images," IEEE Wescon Conv. Record, 8/68.
8. J.Rabinow, "Developments in Character Recognition Machines," in Fischer et al, *Optical Character Recognition*, Spartan Press, 1962.
9. G.Marston, "Recognition of Characters Transmitted by Picturephone," Sc.D. Thesis, MIT Dept. of Elec. Eng., 6/71.
10. C.Seitz, "An Opaque Scanner for Reading Machine Research," M.S. Thesis, MIT Dept. of Elec. Eng., 1/67
11. A.Laemmel, "Coding Processes for Bandwith Reduction in Picture Transmission," Report R-246-51, PIB 187, Polytechnic Institute of Brooklyn, 8/51. (Available from OTS, US Dept. of Commerce. #135119)
12. P.Elias, "Predictive Coding," Ph.D. Thesis, Harvard Univ., 5/50.
13. T.Huang, "Run-Length Coding and its Extensions," in Huang et al, eds, *Picture Bandwidth Compression*, Gordon and Breach, 1972., pp 231 ff. (Report of a meeting at MIT in 1968)
14. R.Hunter et al, "International Digital Facsimile Coding Standards," Proc. IEEE, 68,7,7/80, pp 854-867.
15. R.Arps, "Bibliography of Digital Graphic Image Compression and Quality," IEEE Trans. Information Theory, 1/74, pp120-122.
16. S.Manber, "Pattern Generator," USP 3,471,848, 8/15/86.
17. J.Ahlberg et al, *The Theory of Splines and Their Applications*, Academic Press, 1967.
18. J.Pan, op. cit. Ref. 6.
19. M.Mathews, et al, "Three Fonts of Computer-drawn Letters," J. Typographical Research, 1,4,10/67, pp 345-56.
20. M.Mathews, et al, "Generation of Graphic Arts Images," USP 3,422,419, 10/19/65.
21. A.Frank, "Parametric Font and Image Definition and Generation," Proc., Fall Joint Computer Conference, 1971, pp 135-144.
22. P.Coueignoux, "Compression of Type Faces by Contour Coding," M.S. Thesis, MIT Dept. of Elec. Eng., 1/73.

23. J.Rabinow, op. cit., Ref. 8.
24. W.Schreiber, op. cit, Ref. 7.
25. I.Sutherland, "Sketchpad: A Man-Machine Graphical Communication System," Tech. Rep. 296, MIT Lincoln Lab., 1/63.
26. C.Wylie, et al, "Half-tone perspective drawings by computer," Proc., Fall Joint Computer Conf., 1967, pp 49-58.
27. A.Erdahl, et al, "Electronically Generated Perspective Images," USP 3,665,408, 5/26/70.
28. *Seybold Report*, 1, 12, 2/14/72; 1, 13, 2/28/72.

Appendix 1. A Decoding System for Outline-Coded Characters

In this section, I describe a program that I wrote to familiarize myself with the problems of decoding characters that had been outline coded.[11] To make the problem as simple as possible, the outlines were approximated by a series of straight lines (vectors), specified by the coordinates of their end points. This is not a very efficient code, but it is an easy one to decode. Furthermore, the entire character code can consist simply of the list of vectors, although it is helpful if an "end-of-character" code is also added, as well as the number of vectors and the number of scan lines in the character. It is also convenient if the vectors are sorted in order of starting x value, where x is the line number, and, for a given x, in order of starting y value. If they are not provided in this order, it is easy to sort them by any standard method. If circular arcs were to be used in addition to vectors, a third number would be required for each segment, and the decoding would be somewhat more complicated, but still perfectly straightforward.

In decoding systems designed to produce output on a raster display device, the required output is a list of edge points for each scan line, commonly called "stroking data." The main part of the program produces the stroking data, line by line, from the list of vectors. Using the convention that each character is completely surrounded by white, each line starts with a white sample. As the display scans out (strokes) the line, sample by sample, the output is made to alternate between white and black at each edge point. This, of course, is a conventional run-length decoder.

The first step is to create the active vector list for the first scan line, and then to update it as each line is generated, immediately going to the next line if there are no active vectors. (Alternatively, the active vector list can be part of the character code, which makes the code less efficient but the decoding faster.) Vectors are added to the list when their starting x value equals the present x value, and deleted when their ending x value is less than the present x value. In order to avoid sorting the active vectors each time the list is changed, I found it more convenient to place new vectors into the first empty space in the list provided, and to maintain a pointer list to show the order in which they must be used.

[11] It takes very little experience with such programs to realize that the key to decoding efficiency is to minimize sorting. While sorting is a standard operation in either software or hardware, it is very time-consuming.

The final step on each scan line is to update the continuing vectors. This is done using the expression

$$y = y_s + m(x - x_s),$$

where m, the slope, is

$$m = (y_e - y_s)/(x_e - x_s).$$

The updated y values of the active vectors, accessed in order, provide the run-length coded, or "stroking" data for each line.

The program also includes scaling, both up and down, by scaling the vector data. Since the coordinates of the vectors are given in absolute values referred to a corner of the character grid, and it is desired to maintain the baseline during scaling, the scaling is done so as to keep the left edge of the baseline in constant position.

Some complications are introduced by scaling. Since it was desired to display the output on the screen of an IBM PC as well as the associated printer, the characters are limited to 80 samples in height, including descenders. About six characters were hand-coded in a grid 60 samples above the baseline and 20 below, and of any width. All vector coordinates are integers, but after scaling to an accuracy of .1 sample, they may not be. In addition, when down-scaling, vectors may become less than 1 sample in length. To handle these cases correctly, the procedure mentioned above was introduced, in which new vectors are introduced in one step and expired vectors eliminated in a second step, before updating and output. Very short vectors are thus added and deleted without ever being used.

Another complication of down-scaling occurs when stroke widths become less than one sample. In that case, peculiar looking characters result. When up-scaled characters exceed the 80-sample allowance, they are truncated. Thus there are practical limits to the degree of scaling, as might be expected.

As can readily be seen from this example, outline-coded characters can readily be decoded to produce output on a raster scanner using entirely conventional data-processing techniques. It may well be possible to devise a decoder that has patentable inventive features, but in that case one would think that the patent ought to protect that particular novel and advantageous decoding method or apparatus, and not decoding in general.

Appendix 2. More Efficient Codes

The end-point code used in the example of Appendix 1 provides for a very simple decoder, so much so that if more efficient codes are used, it may be a good idea to transform them into the end-point code as an intermediate step in the decoding process. What makes that code so convenient is that the position of each vector is unambiguously specified by the code for that vector only, regardless of their order. Some saving is possible by using relative rather than absolute coordinates, in which case the ordering of vectors becomes essential. Another saving comes from noting that, in most cases, the starting

point of one vector is the ending point of another. Thus for vectors along a single contour, an ordered list of end points, together with an end-of contour code, is a more compact description. (For decoding in scanline order, the contours should be "one-way," or single-valued functions of x.) This is the method used in the numerically controlled machine tool work, in the Seaco typesetter, and in Coueignoux's thesis. It is also the method used in connected-dot puzzles, as in Fig. 6, or in giving instructions for the path of ships and airplanes. As an idea, it is hardly novel.

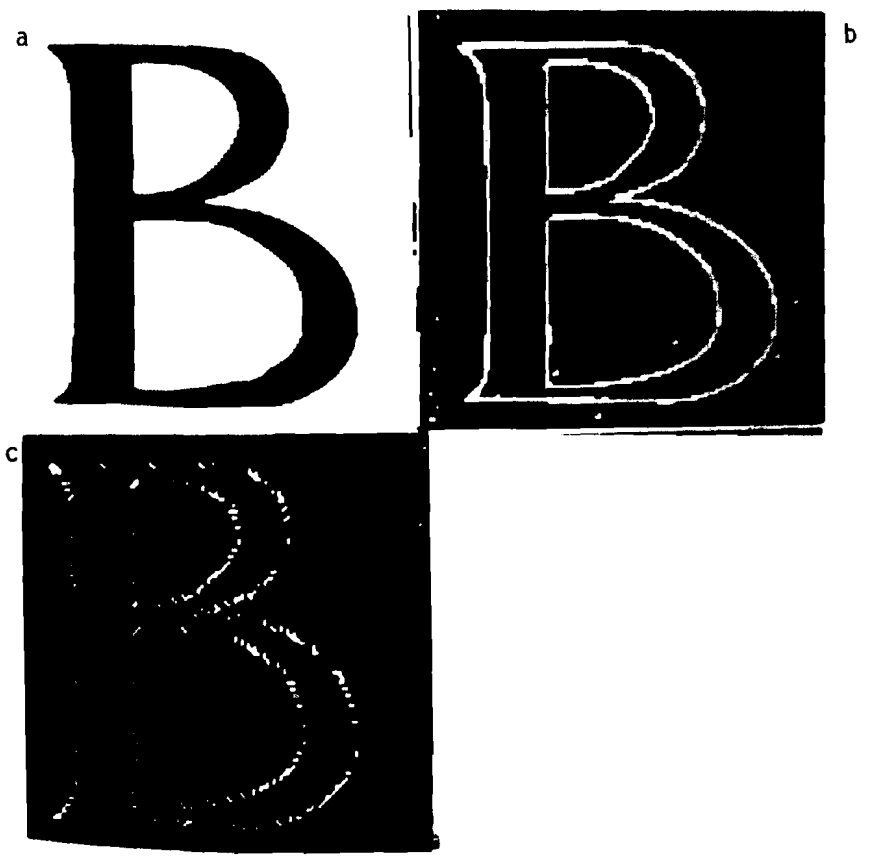


Fig. 1 *Coding of the Letter B by Outlines, from Pan's thesis, Ref. 6.* The original character, on a 128x128 grid, is shown in (a). The outline points are extracted and shown in (b). In (c), the endpoints of the straight line segments, automatically fitted to the extracted outline, are shown.

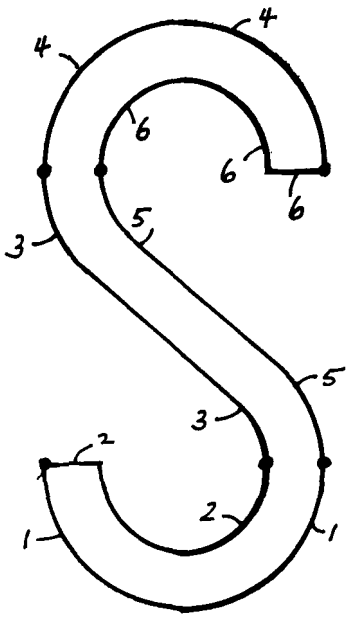


Fig. 2 *Outline of the Letter S Divided into "One-Way" Curves.* The numbers on the outlines show the order in which the outline sections, each of which is a single-valued function of the vertical coordinate, would be encountered in vertical scanning from bottom to top. This division has the effect that each section intersects any one scan line just once.

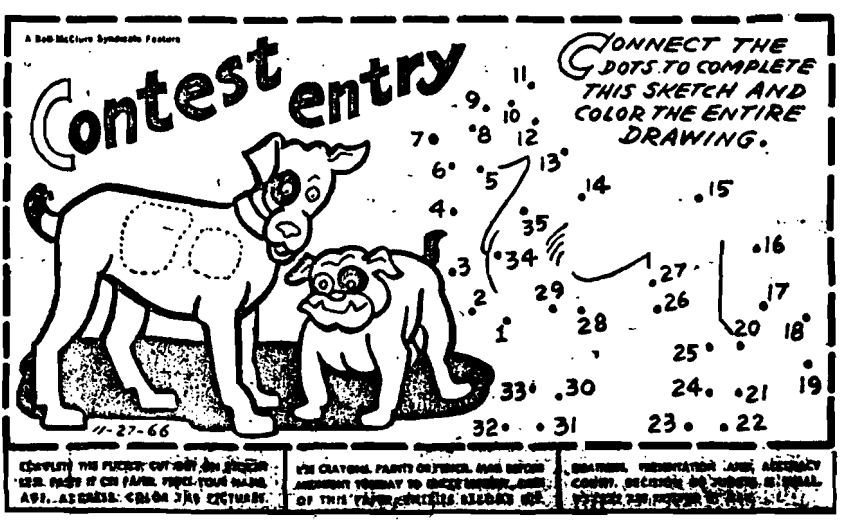
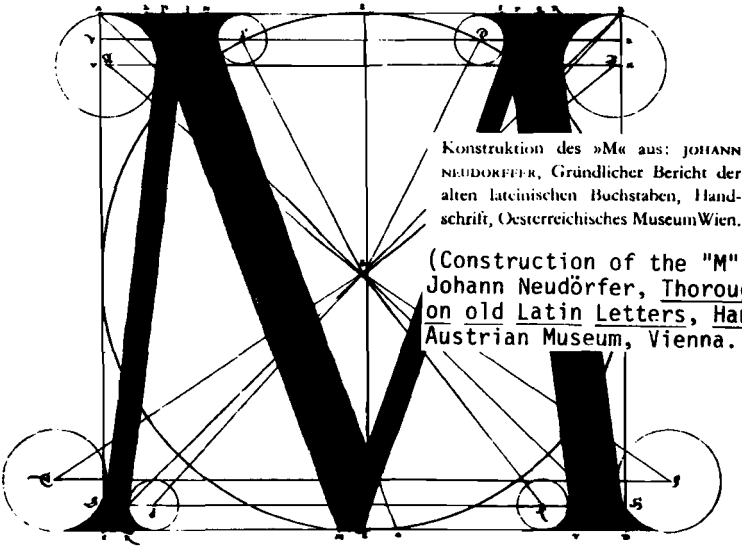


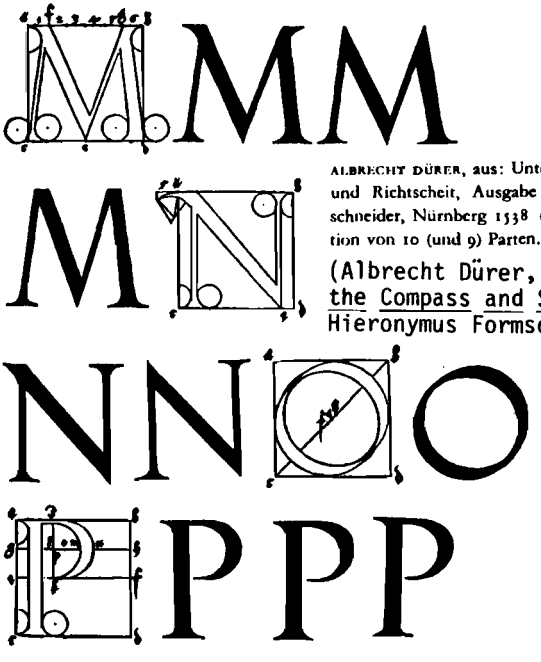
Fig. 6 *Connected-Dot "Puzzle."* In this example from the *Boston Sunday Globe* of 27 November 1966, the figure of an animal can be drawn by connecting the points in the order given. Of course, it is no puzzle at all, being intended for small children. It shows very clearly that one well known code for specifying arbitrary outlines comprises the coordinates of the end points of the approximating vectors together with instructions as to the order of joining.



3b

Konstruktion des »M« aus: JOHANN NEUDÖRFFER, Gründlicher Bericht der alten lateinischen Buchstaben, Handschrift, Oesterreichisches MuseumWien.

(Construction of the "M" from: Johann Neudörfer, Thorough Report on old Latin Letters, Handwriting, Austrian Museum, Vienna.)



3a

ALBRECHT DÜRER, aus: Unterweisung mit dem Zirkel und Richtscheit, Ausgabe von Hieronymus Formschneider, Nürnberg 1538 (verkleinert). Konstruktion von 10 (und 9) Parten.

(Albrecht Dürer, from: Instruction with the Compass and Straight Edge, Edited by Hieronymus Formschneider, Nürnberg 1538

3c

Floor Plan

The Metropolitan
Museum of Art

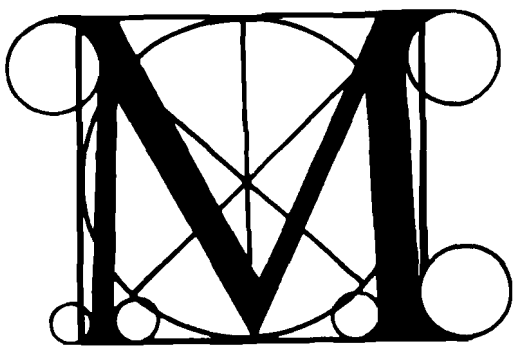


Fig. 3 *Early Description of Character Outlines by Circular Arcs and Straight Lines.* a. Constructions by Albrecht Durer, before 1538. b. Constructions by Johann Neudorfer, 16th or 17th century. c. Contemporary use of Neudorfer's construction by the Metropolitan Museum of Art, New York.

Fig. 4 *Outline Points Associated with Various Scanning Systems.* a. The letter R, on a 128x128 grid. b. Computer-generated closed outline with contiguous points. c. Outline points for horizontal scanning. d. Outline points for vertical scanning.

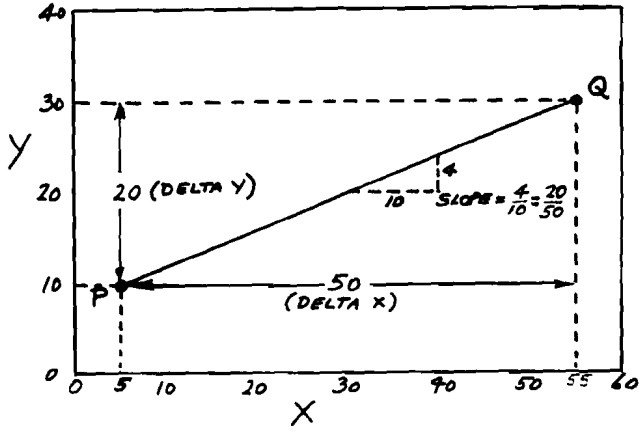


Fig. 5 *Various Ways to Specify a Vector.* The line segment from P to Q can be defined by the absolute coordinate of the end points: 5, 10, 55, 30; by the coordinates of P and the relative coordinates (deltas) of Q: 5, 10, 50, 20; by the coordinates of P, one delta, and the slope: 5, 10, 50, .4; and by several other combinations. All of these can be found in any text on analytic geometry.