Approximate Coding of Digital Contours

SAMBHUNATH BISWAS AND SANKAR K. PAL

Abstract—Two methods for the coding of the discrete contour of binary images are proposed. A set of key pixels (guiding pixels) on the contour is defined for this purpose. The decoding schemes approximate the contour through the key pixels using quadratic Bezier approximation technique. The amount of deviation of the decoded image from the original image has been examined through the objective measures of percentage error and shape compactness. The decoded images are found to be faithful reproductions of the original image. A set of cleaning operations is also introduced as an intermediate step before final reproduction. The bit requirements and the compression ratios are also found to be improved significantly as compared to the contour run length coding and discrete line segment coding techniques.

I. Introduction

Image coding is a technique that represents an image, or the information contained in it, with fewer bits. Its objective is to compress the data for reducing its transmission and storage costs while preserving its information.

The extent to which information is to be preserved depends on the problem in hand. For example, data compression applications are motivated by the need to reduce storage requirements where it is important to employ encoding techniques that allow perfect reconstruction (i.e., no loss of information) of the images from their coded form. In image transmission applications, e.g., transmission of space probe pictures for human interpretation, interest lies in achieving maximum reduction in the quantity of data transmitted subject to the constraint that a reasonable amount of fidelity be preserved. Here the main emphasis is given on reducing the amount of data transmitted, and the encoding technique need not be error-free as long as the resulting images are acceptable (depending on some criteria) for machine or visual analysis. Similar is the case for feature extraction problems where the most important consideration is to reduce the data while preserving just the information necessary to allow a machine to discriminate different items (regions) of interest in an image.

Various techniques, such as spatial domain methods, transform coding, hybrid coding, interframe coding, etc., where both exact (error-free) and approximate (faithful replica) coding algorithms for binary and gray-tone images have been formulated, are avail-

Manuscript received January 15, 1988; revised August 10, 1988.

The authors are with the Electronics and Communication Sciences Unit, Indian Statistical Institute, 203 Barrackpore Trunk Road, Calcutta 700 035, India.

IEEE Log Number 8824409.

0018-9472/88/1100-1056$01.00 ©1989 IEEE

able in [1]–[3]. Approximate coding of gray-tone contour for extraction of its primitive (lines and arcs of different degrees of curvature) using fuzzy sets, is described by Pal *et al.* [4], [5].

Bezier approximation technique [6], [7], which uses Bernstein polynomials as the blending function, provides a successful way to approximate an arc (not having any inflexion point) from a set of at least three control points. The approximation scheme is simple and useful for its axis-independence property. It is also found to be computationally efficient.

The present work describes two algorithms for approximate coding of binary images based on the Bezier approximation technique. First, a contour is decomposed into a set of arcs and line segments. For this, a set of key pixels is defined on the contour. In the first method vertices of Bezier characteristic triangles corresponding to an arc are coded. The second method, on the other hand, replaces a point by an intercept, thus further reducing the bit requirement. The regeneration technique involves Bresenham's algorithm [8] in addition to Bezier's method. During the regeneration process, key pixels are considered to be the guiding pixels, and their locations are, therefore, in no way disturbed. To preserve them and to maintain the connectivity property, some intermediate operations (e.g., deletion and shifting of undesirable pixels generated by Bezier approximation, and insertion of new pixels) are introduced to achieve a more faithful reproduction.

The effectiveness of the algorithms is compared with two existing algorithms based on contour run-length coding (CRLC) [9] and discrete line segment coding (DLSC) [10]. The compression ratios of the proposed methods are found to be significantly improved without much affecting the quality when a set of images is considered as input. The difference in area between the input and output versions, keeping the location of the key pixels the same, and the compactness are also computed to provide a measure of error of the techniques.

II. Bezier Approximation Technique

Bernstein Polynomials

The Bernstein polynomial approximation of degree m to an arbitrary function $F: [0, 1] \to R$ is defined as

$$B_m[f(t)] = \sum_{i=0}^{m} f(i/m)\phi_{im}(t)$$

where the weighting functions ϕ_{im} are, for fixed t, the discrete binomial probability density functions for a fixed probability

$$\phi_{im}(t) = \binom{m}{i} t^i (1-t)^{m-i}, \qquad i = 0, 1, \cdots, m \qquad (1)$$

where

$$\binom{m}{i} = \frac{m!}{(m-i)!i!}.$$

The remarkable characteristics of the Bernstein polynomials are the extent to which they mimic the principal features of the primitive function f and the fact that the Bernstein approximation is always at least as smooth as the primitive function f where "smooth" refers to the numbers of undulations, the total variation, etc.

Bezier Curves

This class of curves was first proposed by Bezier [6], [7]. The parametric form of the curves is

$$X = P_x(t) \qquad (2a)$$

$$Y = P_y(t). \qquad (2b)$$

Let $(x_0, y_0), (x_1, y_1) \cdots (x_m, y_m)$ be $(m+1)$ ordered points in a plane. The Bezier curve associated with the polygon through the aforementioned points is the vector-valued Bernstein polynomial and is given by

$$P_x(t) = \sum_{i=0}^{m} \phi_{im}(t) x_i \qquad (3a)$$

$$P_y(t) = \sum_{i=0}^{m} \phi_{im}(t) y_i \qquad (3b)$$

where $\phi_{im}(t)$ are the binomial probability density function of (1).

In the vector form, (3) becomes

$$P(t) = \begin{pmatrix} P_x(t) \\ P_y(t) \end{pmatrix} \text{ and } v_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$$

so that

$$P(t) = \sum_{i=0}^{m} \phi_{im}(t) v_i. \qquad (4)$$

The points $v_0, v_1, \cdots, v_m$ are known as the guiding points or the control points.

From (4) it is seen that

$$P(0) = v_0 \text{ and } P(1) = v_m.$$

Thus the range of t significantly extends from 0 to 1. The derivative of $P(t)$ is

$$P'(t) = -m(1-t)^{m-1} v_0 + \sum_{i=1}^{m-1} \binom{m}{i} \{ i t^{i-1}(1-t)^{m-i} - (m-i) t^i (1-t)^{m-i-1} \} v_i + m t^{m-1} v_m.$$

Now $P'(0) = m(v_1 - v_0)$ and $P'(1) = m(v_m - v_{m-1})$. Thus the Taylor series expansion near zero is

$$P(t) = P(0) + tP'(0) + \text{higher order terms of } t$$
$$\approx v_0(1 - mt) + \cdots$$

and an expansion near one is

$$P(t) = P(1) - (1-t)P'(1) + \text{higher order terms of } (1-t)$$
$$\approx v_m\{1 - m(1-t)\} + m(1-t)v_{m-1}.$$

It is now clear that as $t \to 0$, the Bezier polynomial lies on the line joining v_0 and v_1, and for $t \to 1$ on the line joining v_{m-1} and v_m. This means that these lines are tangents to the curve at v_0 and v_m.

Also, since $\sum_{i=0}^{m} \phi_{im}(t) = 1$, the Bezier curve lies inside the convex hull of the control points. For cubic Bezier curves, $m = 3$. The control polygon then consists of four control vertices v_0, v_1, v_2, v_3, and the corresponding Bezier curve is

$$P(t) = (1-t)^3 v_0 + 3t(1-t)^2 v_1 + 3t^2(1-t)v_2 + t^3 v_3. \qquad (5)$$

Though the cubic Bezier curve is widely used in computer graphics [11], we have used here its quadratic version to speed up the procedure.

For a quadratic Bezier curve, $m = 2$ and the control polygon always consists of three points. The Bernstein polynomials in this case are

$$\phi_{02} = (1-t)^2 = 1 - 2t + t^2$$
$$\phi_{12} = 2(1-t)t = 2t - 2t^2$$
$$\phi_{22} = t^2.$$

In the polynomial form the Bezier curve is

$$P(t) = t^2(v_1 + v_2 - 2v_1) + t(2v_1 - 2v_0) + v_0. \qquad (6)$$

This is a second-degree polynomial and can be computed much faster than in Horner's process [11].

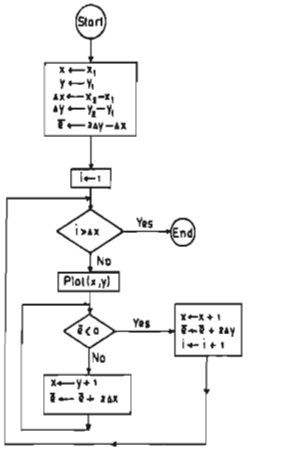

Fig. 1. Flow chart for Bresenham's algorithm for generating straight line in first octant.

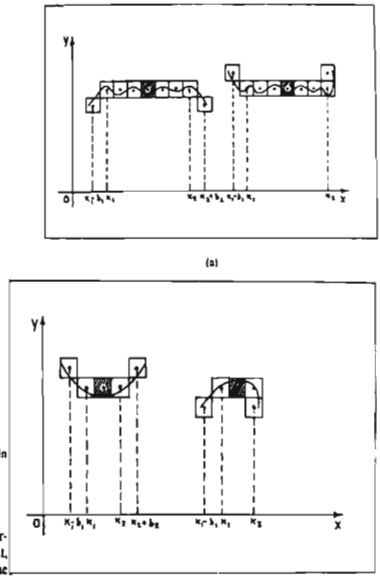

Fig. 2. Possible behavior of $f_c(x)$ when $f(x)$ is constant. (a) Considering local maxima/minima of $f_c(x)$. (b) Considering global maximum, minimum of $f_c(x)$. ■ denotes the position of key pixel.

Bresenham Algorithm [8]

The underlying concept of the Bresenham algorithm for generating the points for a straight line segment restricted to an octant, given its two end points, lies in checking the proximity of the actual line to the grid location. Let (x_1, y_1) and (x_2, y_2) be the two points through which a discrete straight line segment is required. For this, the intercept of the line segment with the line at $x = x_1 + 1, x_1 + 2, \cdots, x_2$ are considered. If the intercept with the line at $x = x_1 + 1$ is closer to the line at $y = y_1 + 1$, then the point $(x_1 + 1, y_1 + 1)$ better approximates the line segment in question than the point $(x_1 + 1, y)$. This means if the intercept is greater than or equal to half the distance between $(x_1 + 1, y)$ and $(x_1 + 1, y_1 + 1)$, then the point $(x_1 + 1, y_1 + 1)$ is selected for approximation; otherwise, the point $(x_1 + 1, y)$ is selected. Next, the intercept of the line segment with the line at $x = x_1 + 2$ is considered, and the same logic is applied for the selection of points.

Now instead of finding the intercept, an error term e is used for the selection purpose. Initially, $e = -1/2$, and the initial point (x_1, y_1) is selected. The slope of the line, $\Delta y/\Delta x$, is added to e, and the sign of the current value of $e = e + \Delta y/\Delta x$ is tested. If it is negative, then the point is selected along the horizontal line, i.e., x is incremented by one and y remains the same. The error term is then updated by adding the slope to it. However, if the error term is positive (or two) then the point is selected along the vertical line, i.e., both x and y are incremented by one. The error term is updated by decreasing it by one.

For integer calculation, e is initialized to

$$\bar{e} = 2\Delta y - \Delta x \text{ because } 2\Delta y - \Delta x = 2e\Delta x = \bar{e} \text{ (say)}.$$

The details of the algorithm for the first octant are given in the flow chart as shown in Fig. 1.

III. Key Pixels and Contour Approximation

A. Key Pixels

In the analytic plane the contour of an object exhibits sharp maxima and minima, and these points can be detected almost accurately without much difficulty. However, when the contour is digitized in a two-dimensional array space of $M \times N$ points or pels or pixels, the sharpness in the curvature of the contour is destroyed due to the information loss inherent in the process of digitization. The error is known as the digitization error. Consequently, it becomes rather difficult and complicated to estimate the points of maxima and minima. An approximate solution to this problem is to define a set of pixels, we call key pixels, which are close to the points of maxima and minima.

For example, consider a function $f(x)$ in the discrete plane. When $f(x)$ is constant in an interval $[k_1, k_2]$, the corresponding analytic function $f_c(x)$ may exhibit local maxima and minima (or a global maximum or minimum) anywhere within the interval as shown in Figs. 2(a) and (b).

If we get pixels either directly connected or outward-corner connected to the end pixels of the interval $[k_1, k_2]$ such that both the values of $f(x)$ at these pixels are either greater or smaller than its value in the interval, then we assume a maximum or minimum point to exist at the midpoint of the interval, i.e., at $x = (k_1 + k_2)/2$ if $(k_1 + k_2)$ is even and at $x = (k_1 + k_2 + 1)/2$ if $(k_1 + k_2)$ is odd. Let us consider this point or pixel in the discrete plane to be a key pixel. Another example for the existence of a key pixel is depicted in Fig. 3 for which $f(x)$ is not constant over an interval.

B. Definition

A function $f(x)$, constant in $[k_1, k_2]$, in the discrete plane is said to have a key pixel P at $x = c$ (where $c = (k_1 + k_2)/2$

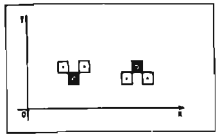

Fig. 3. Position of key pixel when $K_1 = K_2 = C$.

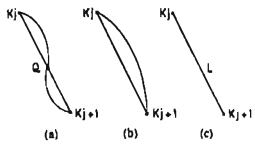

Fig. 4. Types of GE. (a) Arc with inflexion point. (b) Arc. (c) Straight line.

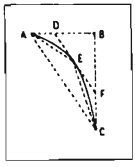

Fig. 5. Bezier characteristic triangles for arc AEC.

or $(k_1 + k_2 + 1)/2$ corresponding to even and odd values of $(k_1 + k_2)$) provided $\delta_1, \delta_2 \in \{0,1\}$ exist such that in both the intervals $[(k_1 - \delta_1), k_1]$ and $[k_2, (k_2 + \delta_2)]$

$$\text{either } f(c) > f(x)$$

$$\text{or } f(c) < f(x).$$

When $k_1 = k_2 = c$, the definition is applicable for Fig. 3 where $\delta_1 = \delta_2 = 1$.

Note here that the foregoing definition corresponds to Figs. 2 and 3 where key pixels lie on a horizontal sequence of pixels for the interval $[k_1, k_2]$ of x. Similarly, key pixels can also be defined for a vertical sequence of pixels for the interval $[k_1, k_2]$ of y.

C. Contour Approximation

Let $k_1, k_2, \cdots, k_p$ be P key pixels on a contour. The segment (the geometrical entity, GE) between two key pixels can then be classified as either an arc or a straight line. If the distance of each pixel from the line joining the two key pixels is less than a prespecified value, say, δ, then the segment is considered to be a straight line (Fig. 4(c)); otherwise, it is an arc. The arc may again be of two types, with all the pixels either lying on both sides (Fig. 4(a)) or lying on the same side (Fig. 4(b)) of the line joining the key pixels. Let us denote the GE in Fig. 4(c) by L (line) and that in Fig. 4(b) by CC (curve). It is, therefore, seen that the GE in Fig. 4(a) is nothing but a combination of two CC's meeting at a point Q (point of inflexion). Therefore, the key pixels on the contour of a two-tone picture can be used to decompose the contour into two types of GE's, namely, arcs and lines.

Let us now consider Fig. 5, where the curve CC in Fig. 4(b) is first of all enclosed within a right triangle ABC where AC (the line joining k_j and k_{j+1}) is the hypotenuse and AB and BC are the horizontal and vertical lines, respectively. It is proved in Appendix I that the arc CC will always be confined within a right triangle ABC. A line DF is then drawn which is parallel to the hypotenuse AC and passes through the pixel E of maximum displacement with respect to AC. Thus the subtriangles ADE and CFE so constructed may be taken as the characteristic triangles to approximate the curve CC by the quadratic Bezier approximation technique. The preservation of the information of Bezier characteristic triangles with the help of key pixels forms the basis of the underlying concept of the proposed coding schemes.

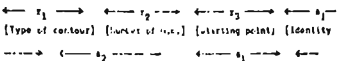

Fig. 6. Bit pattern.

IV. Coding Schemes

Two methods for coding are proposed. In Method 1, only two points (namely, E and C) are stored to preserve the characteristic triangles corresponding to an arc when its starting point A is known beforehand. The points D or F need not be stored because they can automatically be obtained from the aforementioned points. For example, D is the point of intersection of the horizontal line through A and the line through E and parallel to AC. Note that the end point of a GE is the starting point of its following GE.

In Method 2, a further reduction in bit requirement is made by considering only one point along with a length, instead of storing two points for representing an arc. For example, point C along with the length AD is stored here instead of the points C and E (as in Method 1). Since the length AD is stored instead of the point E, it is also desirable to store the sense of the arc (whether it lies on the left or right side of the line AC). This enables one to draw the line DF, and its midpoint is then considered for E during regeneration procedure.

Regarding a straight line, it is obvious that only one point needs to be stored when the starting point is known. The algorithm for key pixel extraction is shown in Appendix II.

A. Bit Requirement for Method 1

Let there be p different contours in a binary image of size $M \times N$ where $M = 2^m$ and $N = 2^n$. The contours may be of two types: either closed or open. If n_c and n_o are respectively, the number of key pixels (including the end pixels for open contour) and the points of inflexion on a contour, then the number of arcs and straight lines (segments) is $(n_o + n_c - 1)$. Of them, let n_s be the number of straight-line segments. For a closed contour the initial key pixel is the same as the final key pixel.

The codeword s of a GE is variable in length. s consists of two subwords s_1 and s_2. s_1 always represents an identity (arc or line) of the GE, while s_2 denotes its description. When the GE is an arc, s_2 gives the vertices of the characteristic triangle (for example, A, E, C in Fig. 5). For a straight-line segment, s_2 indicates the end points of the line segment. It is obvious that the current end point is always the starting point of the succeeding GE. The bit pattern representing a contour is, therefore, of the nature given in Fig. 6.

Types of contours (open or closed) can be represented by a single bit. In the worst case, all the GE's may be straight-line segments, and the number of key pixels may be $= MN$. Thus it needs $(m + n)$ bits to represent the total number of GE's in a contour. The identity of a GE (arc or line) can be represented by a single bit.

Given a starting point of the contour, we need two points for describing an arc and one point for a straight line. Each point can be represented by $(m + n)$ bits.

Therefore, for describing an open contour consisting of $((n_k + n_i - 1) - n_s)$ arcs and n_s lines, we need

$$T_o = (m+n) + 2(m+n)\cdot((n_k + n_i - 1) - n_s) + (m+n)n_s$$

bits where the first term corresponds to the starting point. For a closed contour the amount of bits required is $T_c = T_o - (m+n)$ since the last key pixel (end point) corresponds to the starting point.

From Fig. 6 it is, therefore, seen that T_o (or T_c) gives the bit requirement for s_2s (including r_3) only. The total requirement, considering the remaining entities of Fig. 6, will be

$$B_{\text{total}} = \alpha + \beta + \gamma + \delta$$

where

α requirement corresponding to $r_1 = 1$,
β requirement corresponding to $r_2 = (m+n)$,
γ requirement corresponding to $s_1s = (n_k + n_i - 1)$,

and $\delta = T_o$ or T_c.

B. Bit Requirement for Method 2

An arc is coded here by a point and an intercept (along AB or BC as in Fig. 5) instead of two points as in the case of Method 1. The intercept can be coded by either m or n bits. Now to reconstruct the arc during the regeneration procedure we need the information regarding its sense, i.e., whether it lies on the left or right side of the line joining two key pixels. This requires one bit. The values of α, β, and γ are same as in Method 1.

Let there be h horizontal intercepts in $(n_k + n_i - 1 - n_s) = (\theta,$ say) arcs. The number of bit reductions as compared to Method 1 is then

$$\begin{aligned} T &= 2(m+n)\theta - [(m+n)\theta + (\theta - h)n + hm + \theta] \\ &= (m+n)\theta - (\theta - h)n - hm - \theta \\ &= m(\theta - h) + hn - \theta. \end{aligned}$$

For a square image, $M = N$,

$$T = \theta(m-1).$$

Therefore, the compression ratio (as defined by the total number of pixels in the image/number of bits required to represent the image) for Method 2 is greater than that of Method 1.

V. Decoding

The coded binary string output corresponding to Methods 1 and 2, is shown in Fig. 7(a) and (b). $(m+n)$ indicates the word length for the number of GE's, whereas $(m)+(n)$ denotes the coordinate of a point.

Decoding of the string of Fig. 7(a) is based on the following notations. First, the bit (l_1) indicates the type of contour (i.e., $l_1 = 0$ for open and 1 for closed). The next sequence l_2 of $(m+n)$ bits represents the number of GE's present in the contour. The first m bits of the sequence l_3 denote the value of the ordinate, whereas the remaining n bits give the value of the abscissa of the starting point. Similarly, the coordinate of the first key pixel is given by the sequence l_4. Bit l_5 tells whether the GE between the points represented by l_3 and l_4 is an arc or a straight line. $l_5 = 0$ for a line and 1 for an arc. If there is an arc, then the following sequence l_6 is considered to indicate the point E (as in Fig. 5). Otherwise, the sequence l_6 will be absent.

As soon as an arc or line is reconstructed, the preceding key pixel point becomes the new starting point. The point designated by the following sequence l_4 then represents the new key pixel for further reconstruction.

The procedure for decoding continues until the number of GE's, as represented by the sequence l_2, is exhausted. After that, a new contour is started with the first bit as l_1.

For Method 2 (Fig. 7(b)), the sequences l_1–l_5 are the same as in Method 1. If $l_5 = 1$ (i.e., showing the presence of an arc), the sequence l_6 then denotes the sense of the arc and l_7, the value of the intercepts (AD as in Fig. 5). $l_5 = 0$ implies the absence of both l_6 and l_7.

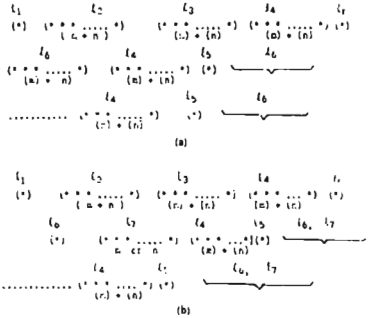

Fig. 7. Coded binary string output of (a) Method 1, (b) Method 2.

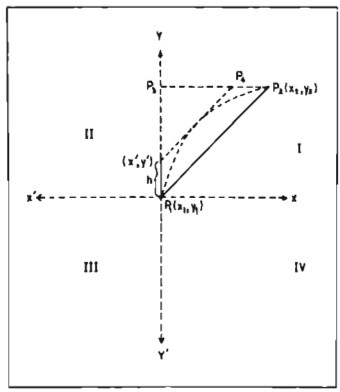

Fig. 8. Detection of Bezier characteristic triangles for Method 2.

VI. Regeneration Technique

During the decoding procedure, if the GE between two key pixels is found to be a straight line, then it is generated by the Bresenham algorithm as mentioned in Section II. If the GE is an arc, the Bezier characteristic triangles are, first of all, constructed to generate its quadratic approximation.

In Method 2, since the intercept along the horizontal or vertical line is considered for further data reduction, we need to find the coordinates of the end point of the intercept to extract the vertices of the Bezier characteristic triangles. The coordinate detection of the end point of the intercept is given in the following.

Consider (x_1, y_1) and (x_2, y_2) to be the initial and final points of an arc as shown in Fig. 8. Let us now imagine a set of mutually perpendicular reference axes placed at the point (x_1, y_1).

TABLE I
DIFFERENCE TABLE FOR RECURSIVE COMPUTATION OF POINTS FOR BEZIER CURVE

t	y	Δy	$\Delta^2 y$
0	c	$aq^2 + bq$	$2aq^2$
q	$aq^2 + bq + c$	$3aq^2 + bq$	$2aq^2$
$2q$	$4aq^2 + 2bq + c$	$5aq^2 + bq$	$2aq^2$
$3q$	$9aq^2 + 3bq + c$	$7aq^2 + bq$	
$4q$	$16aq^2 + 4bq + c$		

Also, let h be the value of the intercept and (X', Y') be the coordinate of the end point of the intercept.

Since an arc may lie either in the left (clockwise) or in the right (anticlockwise) side of the line joining (x_1, y_1) and (x_2, y_2), X' and Y' may have the values

$$\text{either} \begin{cases} X' = x_1 \\ Y' = y_1 + h \end{cases} \quad \text{or} \quad \begin{cases} X' = x_1 + h \\ Y' = y_1 \end{cases}$$

corresponding to the two possible senses of the arc in quadrant I where $x_2 > x_1$ and $y_2 > y_1$.

Similarly, for the other quadrants, where $x_2 < x_1$ and $y_2 > y_1$ (quadrant II), $x_2 < x_1$ and $y_2 < y_1$ (quadrant III), and $x_2 > x_1$ and $y_2 < y_1$ (quadrant IV), we have

$$\begin{aligned} X' &= x_1 - h & X' &= x_1 \\ &\text{or} & &\\ Y' &= y_1 & Y' &= y_1 + h \\ X' &= x_1 & X' &= x_1 - h \\ &\text{or} & &\\ Y' &= y_1 - h & Y' &= y_1 \\ X' &= x_1 + h & X' &= x_1 \\ &\text{or} & &\\ Y' &= y_1 & Y' &= y_1 - h \end{aligned}$$

corresponding to the two possible senses.

Having determined the point (X', Y'), the next task is to construct the line passing through (X', Y') and parallel to P_1P_2 so that it meets the line P_2P_3 at some point P_4. The midpoint of this line, together with the pair of points $((X', Y'), (X_1, Y_1))$ and $((X_2, Y_2), P_4)$, then constitutes the Bezier characteristic triangles for the arc.

A. Recursive Computation Algorithm

An algorithm for computing the values of the second-order Bezier approximation curve using the forward difference scheme is described next. Let

$$y = at^2 + bt + c$$

be a polynomial representation of (6), where the constant parameters a, b, and c are determined by the vertices of the Bezier characteristic triangle.

Suppose a number of points (values of y) on the arc are to be evaluated for equispaced value of the independent variable t. The usual Newton's method of evaluating the polynomial results in multiplications and does not make use of the previously computed values to compute new values.

Assume that the parameter t ranges from 0 to 1. Let the incremental value be q. Then the corresponding y values will be c, $aq^2 + bq + c$, $4aq^2 + 2bq + c$, $9aq^2 + 3bq + c, \cdots$. Let us now form the difference table as shown in Table I.

Observe that

$$\Delta^2 y_j = 2aq^2$$

and

$$y_{j+2} + 2y_{j+1} + y_j = 2aq^2, \qquad \text{for all } j \geq 0.$$

Fig. 9. Deletion of pixels. (a) In absence of key pixel. (b) In presence of key pixel.

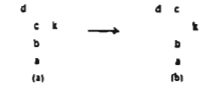

Fig. 10. Shifting of pixels. (a) Contour before shifting. (b) Contour after shifting.

This leads to the recurrence formula $y_2 = 2y_1 - y_0 + 2aq^2$ that involves just three additions to get the next value from two preceding values at hand. Thus one does not need to store all the points on the curve.

VII. IMPLEMENTATION STRATEGIES

After coding a single-pixel-width contour input, the regeneration algorithm as described before is used to decode and it results in its approximated version (output). During regeneration, only the outer contour is traced using Freeman's chain code (clockwise sense), assuring the positions of the key pixels on it. In other words, key pixels are considered to be the guiding pixels (being important for preserving the input shape) during reproduction.

Note that due to the approximation scheme, sometimes the following undesirable situations may arise:

(1) the regenerated contour may not have single-pixel width;
(2) the key pixel may become an interior pixel of the contour.

To overcome these situations, we trace the contours from the ordered regenerated data set, considering the following operations.

A. Deletion of Pixels

During the contour tracing, if a pixel on the contour finds more than one neighbor in its eight-neighborhood domain, then the exterior pixel on the contour is kept while deleting the rest (pixels on the interior contour). However, if there is a key pixel falling in such neighborhood, the key pixel is then retained as the contour pixel and the rest are deleted. This enables us to keep the key pixel always on the contour, thus improving approximation of the input. Fig. 9(a) and (b) depicts the situation. Considering "c" to be the current pixel and "p" the previous pixel, the contour (clockwise) is "a" for the situation as shown in Fig. 9(a), but if the situation is as in Fig. 9(b), the next pixel on the contour is then k (the key pixel).

B. Shifting of Pixels

Suppose a GE is generated, and a key pixel is reached. Now during the generation of a following GE, its first data point may put the preceding key pixel on the interior contour. For example, consider Fig. 10(a). Here abk is a part of the GE already generated. Now generating the next GE $kcd\cdots$, the first move from k to c makes the key pixel (k) lie on an interior contour.

In such cases the data point c is shifted as shown in Fig. 10(b). This preserves connectedness of the pixel c with both the GE's and also ensures single-pixel width of the contour.

C. Undesirable Loop

Sometimes in the vicinity of key pixels an undesirable loop (contour with a single pixel hole) may appear due to the approximated generation procedure. For example, consider Fig. 11. Here

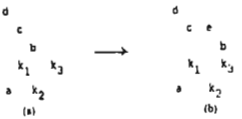

Fig. 11. Undesirable loop. (a) Before cleaning. (b) After cleaning.

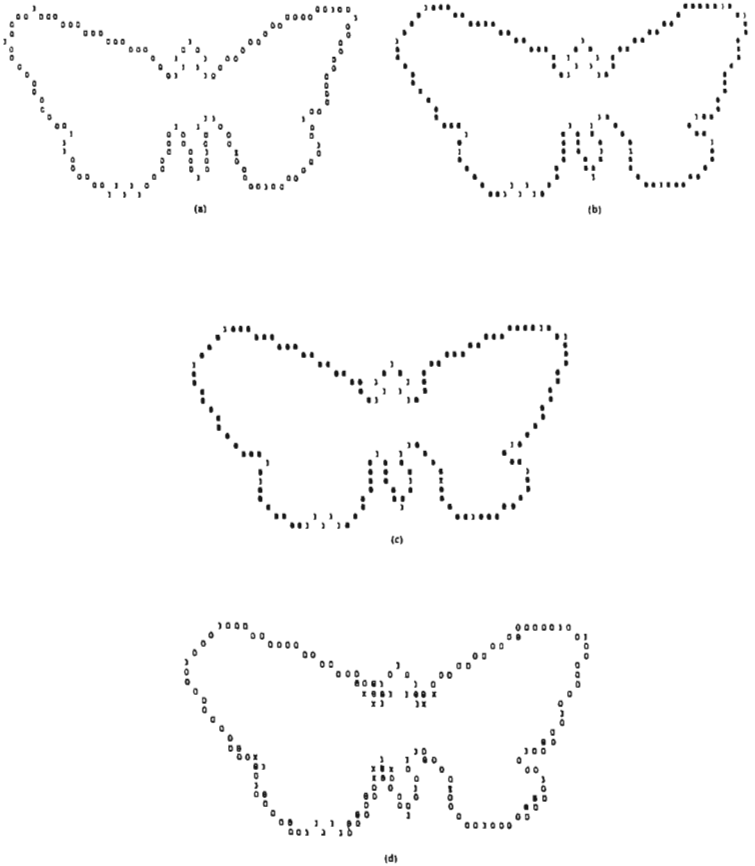

Fig. 12. (a) Butterfly input. (b) Regenerated version by Method 1. (c) Regenerated version by Method 2. (d) Regenerated version by Method 1 before cleaning.

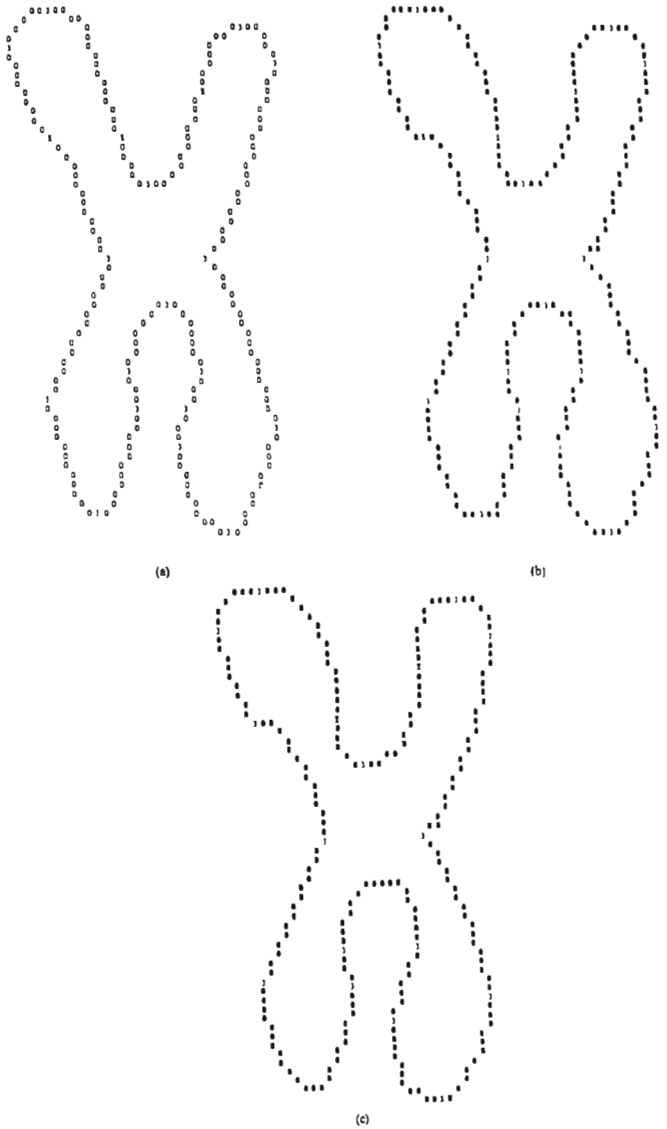

Fig. 13. (a) Chromosome input. (b) Regenerated version by Method 1. (c) Regenerated version by Method 2

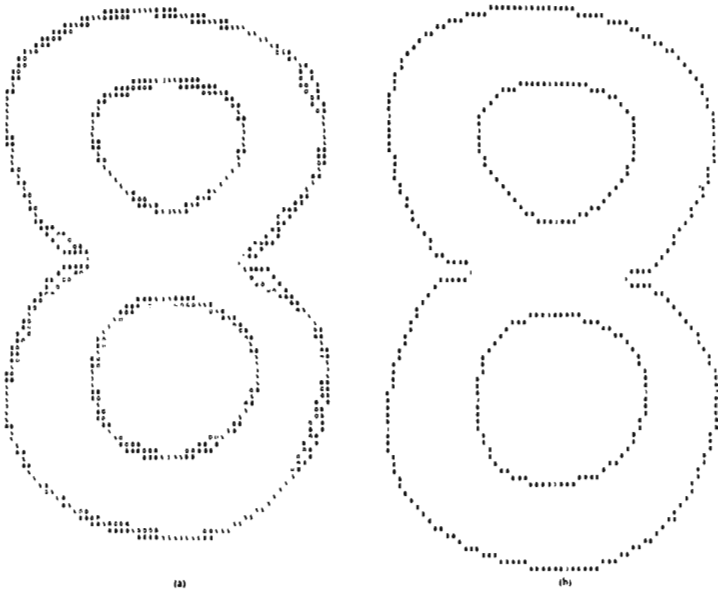

Fig. 14. (a) Numeral 8 input and its regenerated output by Method 1 together. (b) Regenerated version by Method 2.

TABLE II
BIT REQUIREMENT

Figure	Bit Requirement				δ_{ot} (percent) (Relative to CRLC)		δ_{o3} (percent) (Relative to DLSC)	
	CRLC	DLSC	Method 1	Method 2	Method 1	Method 2	Method 1	Method 2
Butterfly	799	531	435	394	183.67	202.79	122.06	134.77
Chromosome	1175	603	452	372	259.95	315.86	133.40	162.09
Numeral "8"	2085	1169	474	386	439.87	540.15	246.62	302.84

GE's $ak_1k_2k_3$ are already generated. The next move from k_3 to b creates an undesirable loop having a single-pixel hole.

To overcome this situation, the pixel b is shifted along with an insertion of a new pixel e (as shown in Fig. 11(b)). Since the shifting of b alone loses the connectivity property between k_3 and the subsequent pixels, it necessitates insertion of a new pixel whose location is governed by the concept of a minimum connected path.

VIII. Results and Discussion

Figs. 12(a), 13(a), and 14(a) show the digital contours of three different figures, namely, a butterfly, a chromosome and the numeral 8, which were considered as input to the proposed coding schemes. The key pixels and the points of inflexion as detected on the input patterns are marked by "3" and "X," respectively.

In Fig. 14(a) the input image contour is represented by the pixels marked "5" and "O" along with "3," denoting its key pixels. The output for Methods 1 and 2 corresponding to the butterfly and chromosome images are shown in Figs. 12(b) and (c) and 13(b) and (c), respectively. The output version for Method 1 corresponding to the numeral 8 contour is marked by "5" and "●" in Fig. 14(a), superimposing on its input. This superimposed diagram facilitates examination of the visual proximity between the input and output versions. Note in this connection that the positions of key pixels in both the input and output remain unaltered. Fig. 14(b) illustrates the output corresponding to Method 2.

For coding the input patterns, the number of GE's for the butterfly, chromosome, and numeral 8 was found to be 27, 19, and 16, respectively. Out of these figures the number of arcs was 9, 16, and 16. The contour of numeral 8, as expected, has a minimum number of GE's and has no straight line.

As a typical illustration, the effectiveness of the cleaning operations (Section VII) performed on the generated points is demonstrated only for the butterfly image. Fig. 12(d) shows such an intermediate state for Method 1 before producing its final

TABLE III
ERROR IN REGENERATION

Figure	Percent Error in Area		Compactness		
			Original Figure	Generated Figure	
	Method 1	Method 2		Method 1	Method 2
Butterfly	8.63	10.07	0.024635	0.025393	0.025551
Chromosome	6.8	6.28	0.016061	0.016672	0.016359
Numeral "8"	2.92	3.49	0.014728	0.014589	0.014860

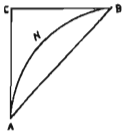

Fig. 15. Arc with its associated right triangle.

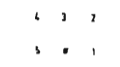

Fig. 16. Directional codes with respect to .

decoded output. Here, θ denotes a pixel deleted and X corresponds to the position where a pixel is inserted to keep connectivity.

To study the efficiency of the coding schemes, the bit requirements for different input images and their relative compression ratios are compared with those in the CRLC and DLSC methods. It is shown in Table II that in both the methods of the proposed techniques the bit requirements are significantly lower. It is obvious that the numeral 8 image, consisting of a few large arcs, provides only a higher compression ratio. The butterfly image, on the other hand, has the largest number of GE's and thus provides the lowest compression ratio.

As the coding schemes are approximate, the regenerated image deviates from its original version. To observe the deviation of regenerated image quality through an objective measure, we have calculated the error in the area and the shape compactness. For the calculation of the area and the perimeter of the contour, we have used the technique proposed by Kulpa [12]. Since the key pixels always are on the contour and the generated arcs are between them and restricted by the respective Bezier characteristic triangles, the maximum error for an arc is the area of its pair of Bezier characteristic triangles. Also, for the previous constraint the shape compactness can provide a good measure of the distortion introduced into the decoded images. Table III shows both the percentage error and the compactness of figures associated with the methods. It is thus seen that the decoded image in each case is a faithful reproduction of its input version. Here too, the butterfly/numeral 8 contours having largest/smallest number of GE's incurred the highest/lowest percent of error in their regeneration. Furthermore, since the regeneration procedure uses the quadratic Bezier approximation technique, the decoded image display is very fast.

Finally, note that the algorithm needs only two points/one point to store an arc/straight line. The sensitivity of the algorithm in the presence of noise depends only on the number of GE's changed. Again, an increase in GE may not always result in an increase in the overall bit requirement. Suppose a total number of GE's consisting of ten arcs and 12 lines is changed to five arcs and 20 lines due to the addition of noise on the input. Here the number of GE's is increased by three, but the overall bit requirement is decreased.

APPENDIX I

Propositon 1: In the discrete plane all the pixels on an arc between two key pixels remain always on or inside a right triangle with the line joining the key pixels as the hypotenuse. The other two sides of the right triangle are the horizontal and the vertical lines through the key pixels.

Proof: When the key pixel is on the horizontal line at $x = c$, it follows from the definition of key pixel that

$$\text{either } f(c) > f(x)$$

$$\text{or } f(c) < f(x)$$

in both the intervals $[(k_1 - \delta_1), K_1]$ and $[K_2, (k_2 + \delta_2)]$, where $f(x)$ is constant in $[K_1, K_2]$ and $\delta_1, \delta_2 \in \{0, 1\}$.

Thus 1) the pixels at K_1 and K_2 are either corner connected or direct connected or its combination to the neighboring pixels outside the interval $[K_1, K_2]$; or 2) when $K_1 = K_2 = C$, the key pixel will have at least one corner connection to its neighboring pixels. Similar arguments hold when the key pixel lies on a vertical line.

Let ANB be the arc with A and B being two successive key pixels as shown in Fig. 15. Now a pixel on the arc can go outside the line AC or BC if and only if a sequence of collinear pixels exists such that its end pixels are either corner connected or direct connected or a combination thereof, or a pixel exists which has at least one corner connection with its neighboring pixel. Both of these conditions lead to the existence of another key pixel outside the line AC or BC. This is a contradiction.

APPENDIX II

Algorithm for Extraction of Key Pixels: The following hold:

$\{P_i\}_{i=1}^{n}$ are the contour points in the binary image

and

$\{(x_i, y_i)\}_{i=1}^{n}$ are their position coordinates.

Since for a closed contour there is a possibility of missing the first key pixel, we need to examine a few more points after the starting point is reached to enable us to get the same back

Step 1 Set $i \leftarrow 1$, count $\leftarrow 1$. Find the initial direction code between P_i and P_{i+1} according to Fig. 16. Let it be d_1.

Step 2 Increment $i \leftarrow i + 1$; if $i = n$, go to step 7; otherwise, find the directional code between P_i and P_{i+1}. Let it be d_2.

Step 3 If $d_1 = d_2$, go to step 2; otherwise, if d_1 div $2 = 0$ and d_2 div $2 = 0$ or if $|d_1 - d_2| = 3$ or 5, then return (x_i, y_i).

Step 4 Increment $i \leftarrow i + 1$; if $i = n$, go to step 7; otherwise, find the direction code between P_i and P_{i+1}. Let it be d_3.

Step 5 If $d_2 = d_3$, then count $\leftarrow$ count + 1 and go to step 4; otherwise, if $|d_1 - d_3| = 0$ or 1, then set count $\leftarrow 1$, $d_1 \leftarrow d_3$ and go to step 2; else do step 6.

Step 6 If count div 2 = 0, then return $(x_{i\text{-count}/2}, y_{i\text{-count}/2})$; otherwise, return $(x_{i\text{-count div 2}}, y_{i\text{-count div 2}})$.

Step 7 Stop.

ACKNOWLEDGMENT

The authors would like to thank Prof. D. Dutta Majumder for his interest in this work, and J. Gupta and S. Chakraborti for rendering the secretarial help.

REFERENCES

[1] *Proc. IEEE*, Special issue on computer graphics, vol. 68, 1980.

[2] M. P. Ekstrom, Ed., *Digital Image Processing Techniques*. New York: Academic, 1984.

[3] T. S. Huang, "Coding of two-tone images," *IEEE Trans. Commun.*, vol. COM-25, pp. 1406–1424, 1977.

[4] S. K. Pal, R. A. King, and A. A. Hashim, "Image description and primitive extraction using fuzzy sets," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-13, no. 1, pp. 94–100, 1983.

[5] S. K. Pal and D. Dutta Majumder, *Fuzzy Mathematical Approach to Pattern Recognition*. New York: Wiley, 1986.

[6] P. E. Bezier, "Mathematical and practical possibilities of UNISURF," in *Computer Aided Geometric Design*, R. E. Barnhill and R. F. Riesenfeld, Eds. New York: Academic, 1974.

[7] B. A. Barsky, "A description and evaluation of various 3-D models," in *Computer Graphics (Theory and Applications*, T. L. Kunii, Ed. New York: Springer-Verlag, 1983.

[8] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Syst. J.*, vol. 4, no. 1, pp. 25–30, 1965.

[9] T. H. Morrin, "Chain link compression of arbitrary black white images," *Comput. Graphics, Image Process.*, vol. 5, pp. 172–189, 1976.

[10] B. B. Chaudhuri and M. K. Kundu, "Digital line segment coding: A new efficient contour coding scheme," *Inst. Elec. Eng., Proc.* pt E, vol. 131, no. 4, pp. 143–147, 1984.

[11] T. Pavlidis, *Algorithms for Graphics and Image Processing*. New York: Springer-Verlag, 1982.

[12] Z. Kulpa, "Area and perimeter measurement of blobs in discrete binary pictures," *Comput. Graphics, Image Process.*, vol. 6, pp. 434–451, 1977.