Image Coding Using Modified Bezier–Bernstein Approximation

SAMBHUNATH BISWAS AND SANKAR K. PAL

Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700 035, India

ABSTRACT

A modified version of the Bezier–Bernstein polynomial approximation technique has been developed which gives local control of data points depending on an absolute error criterion. Based on this concept, two algorithms for coding a gray-tone image have been formulated. Error bounds have been developed which are used to approximate gray segments of pixels. These bounds are determined by the desired error in approximation. Effectiveness of the algorithms has been demonstrated on a set of images.

1. INTRODUCTION

Bezier approximation technique [1] which uses the Bernstein polynomial as the blending function is well known in the field of computer graphics for its speed of computation and axis independence property. It has recently been used successfully in contour coding of binary images [2, 3].

The present work is an attempt to investigate an application of the Bezier–Bernstein polynomial in gray-tone image data compression. First of all, we have investigated if the conventional way of approximating an image by the Bezier–Bernstein polynomial provides any advantage from the data compression standpoint. For this, an entire row (or column) of an image has been considered as a single segment for its approximation. From the approximation theorem of Bernstein [4], it is evident that, for a given error term, the order of the polynomial increases with the maximum gray value present in the segment. Therefore, if the maximum gray value in an image is very large, the order of the polynomial becomes large. Consequently, it introduces a large number of control points and the generation then becomes slow. This makes inconvenience in using the conventional way of approximating an image for its compression.

A modified version of the approximation technique is then developed

to serve the purpose. Here, we have emphasized the local control of data points instead of minimizing the global squared-error sum. An absolute error criterion has been chosen to keep the absolute error within a bound. Also, for the sake of data compression, we have chosen a second-order polynomial.

Based on the modified concept of approximation, two algorithms are proposed. The first algorithm uses the error bound to segment rows (columns) into lines and arcs which are coded in the subsequent stage. The second algorithm, on the other hand, considers a row (or column) as a space curve on intensity surface and separates out the small deflection curve segments on the basis of a homogeneity criterion. These segments are then approximated and coded. The performance of the algorithms is tested on a set of input images. Their discriminating features are also provided.

2. SHORTCOMINGS OF THE BERNSTEIN POLYNOMIAL AND ERROR OF APPROXIMATION

The Bernstein polynomial is a powerful tool to approximate a continuous function within any degree of accuracy. It uses the global information while approximating a function, and the order of the polynomial increases with accuracy in approximation. Let us consider the Bernstein polynomial of degree m,

$$B_m(t) = \sum_{i=0}^{m} f\left(\frac{i}{m}\right) \phi_{im}(t) \tag{1}$$

for approximating a function $f(t)$. Here $f(t)$ is defined and finite on the closed interval $[0, 1]$. Also,

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

with $i = 1, 2, \ldots, m$.

It can be shown that the order m of the Bernstein polynomial $B_m(t)$ satisfies the inequality [4]

$$\frac{k}{\epsilon\delta^2} < m \tag{2}$$

in order to have an error of approximation less than ϵ, where k is the maximum value of the approximating function $f(t)$ in the interval $[0, 1]$. δ is a positive number such that, for points $t_1, t_2 \in (0, 1)$,

$$|f(t_1) - f(t_2)| < \frac{\epsilon}{2},$$

whenever $|t_1 - t_2| < \delta$.

Since a gray-level image can be approximated either row wise or column wise, it appears from the above inequality that the order of the approximating polynomial may be different for different rows (or columns) depending on the value of k. Let us consider the case of approximating a 32×32 gray-level image row wise. If a row has its maximum value $k = 32$, then, for $\epsilon = 1$, (i.e., one unit error in gray value), $m > (32 \times 32 \times 32)/31 \times 31 \approx 34.09$, since $\delta = 31/32$. Therefore, for $k = 32$, one can choose m to be equal to 35.

On the other hand, if $k = 1$, then $m \approx 1.06$, i.e., $m = 2$. $k = 1$ means all the gray-level values in a row are the same and are equal to 1. Since in a gray image, it is very likely to have the maximum value anywhere in each row, the order may be as high as the maximum gray level in the image. This makes the method ineffective.

3. PROPOSED APPROXIMATION TECHNIQUE

It is seen in the previous section that, to approximate a gray-tone image row wise (or column wise), the order of the Bernstein polynomial varies from row to row (or column to column), and for a high-resolution image (small δ) with one unit error in approximation (ϵ), this order becomes close to the maximum value present in each row (or column). The large order of the polynomial, in turn, makes the time of approximation also high. Again, the variation of the order of the polynomial from row to row (or column to column) makes the coding scheme complicated.

An attempt is made in this section to describe an approximation scheme keeping the order of the polynomial equal to 2. For this purpose, let us consider the Bezier–Bernstein (B-B) polynomial which incorporates the Bernstein polynomial as the blending function. Since the order is chosen to be 2, the amount of error ϵ, as expected, will be significantly high. Furthermore, unlike the case of two-tone contour coding [2], the straightforward application of a quadratic B-B polynomial to image data is not able to segment a row (or column) for their proper approximation. In order to circumvent this, a modification of the B-B polynomial is proposed here. This leads us to formulate a scheme by which it is possible to obtain any degree of accuracy in approximation.

Given n points, the approximation algorithm requires $(n - 2)$ unique quadratic B-B polynomials for their representation. Unlike the method described in Section 2, the scheme proposed here decomposes a row (column) either into a single gray segment or into a number of segments so as to enable them to be approximated properly. An error bound has been defined which guides the process of segmentation.

3.1. BEZIER-BERNSTEIN POLYNOMIAL

Equation (1), which represents an mth degree Bernstein polynomial for approximating a function $f(t), 0 \leq t \leq 1$, can be written as

$$B_m(t) = \phi_{0m}(t)f(0) + \phi_{1m}(t)\, f\left(\frac{1}{m}\right) + \phi_{2m}(t)f\left(\frac{2}{m}\right) + \cdots + \phi_{mm}(t)f(1).$$

$B_m(t)$ is seen to consider a set of weights $\phi_{im}(t)$ $(0 \leq t \leq 1)$, along with some fixed points of function $f(t)$ in [0, 1] for its approximation. With the choice of some arbitrary points for $f(\frac{1}{m})$, one can determine $B_m(t)$ for each value of t.

Let v_i represent a point in a multidimensional space and that $v_i = f(\frac{i}{m})$. Thus, $B_m(t)$ becomes

$$B_m(t) = \sum_{i=0}^{m} \phi_{im}(t)\, v_i. \tag{3}$$

Equation (3) can be viewed as a vector-valued Bernstein polynomial, and it approximates a polygon with vertices v_i with t in [0, 1]. $B_m(t)$ is thus seen to generate a space curve. Equation (3) is known as an mth degree Bezier–Bernstein (B-B) polynomial. For $m = 2$, the quadratic B-B polynomial is

$$\begin{aligned} B_2(t) &= \sum_{i=0}^{2} \phi_{i2}(t)v_i \\ &= \phi_{02}(t)v_0 + \phi_{12}(t)v_1 + \phi_{22}(t)v_2 \\ &= (1-t)^2 v_0 + 2t(1-t)v_1 + t^2 v_2. \end{aligned} \tag{4}$$

3.2. APPROXIMATION CRITERIA OF $f(t)$

In order to develop an approximation technique, let us first of all formulate the key criteria associated with this technique. Let us assume $(N-2)$ quadratic B-B polynomials for the representation of N data points, such that

$$f(t_i) = B_2^i(t_i), \qquad i = 1, 2, 3, \ldots, N-2,$$

where $B_2^i(t_i)$ is the value of the ith quadratic B-B polynomial at the point t_i and is given by

$$B_2^i(t_i) = (1-t_i)^2 v_0 + 2t_i(1-t_i)v_1^i + t_i^2 v_2. \tag{5}$$

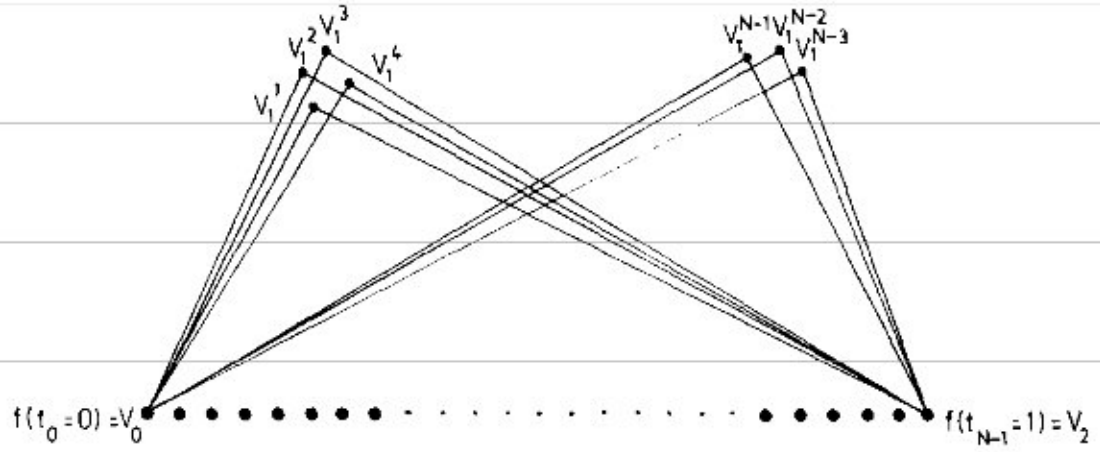

Fig. 1. Supports of approximation due to a sequence of quadratic polynomial.

Let

$$B_2^1(0) = B_2^2(0) = \cdots = B_2^{N-2}(0) = v_0$$

and

$$B_2^1(1) = B_2^2(1) = \cdots = B_2^{N-2}(1) = v_2.$$

In other words, the end supports of all the quadratic B-B polynomials are assumed to be identical. This is shown in Figure 1, where the second supports v_1^i of all the polynomials are shown to be different. From (5), the second support of the ith polynomial is obtained as

$$v_1^i = \frac{B_2^i(t_i) - (1-t_i)^2 v_0 - t_i^2 v_2}{2t_i(1-t_i)}. \tag{6}$$

Let $v_1^i = \overline{v}_1$ when $t_i = \frac{1}{2}$ and let the corresponding B-B polynomial with support v_0, $\overline{v}_1$, and v_2 be $\overline{B}_2(t_i)$. Note that B_2^i (data points) at $t_i = \frac{1}{2}$ may not always be available (e.g., for even number of data points). In this case, we consider two data points in the neighborhood of $t_i = \frac{1}{2}$ ($t_i < \frac{1}{2}$, *and* $t_i > \frac{1}{2}$), to calculate the corresponding v_1^i values and take their average to find $\overline{v}_1^i$. The discrete form of $\overline{B}_2(t_i)$ can be expressed as

$$\overline{B}_2(t_i) = (1-t_i)^2 v_0 + 2t_i(1-t_i)\overline{v}_1 + t_i^2 v_2. \tag{7}$$

So,

$$\left|\overline{B}_2(t_i) - B_2^i(t_i)\right| = \left|\overline{v}_1 - v_1^i\right| \times 2t_i(1-t_i).$$

This expression denotes the absolute difference between the polynomial $\overline{B}_2(t_i)$ and an arbitrary ith quadratic B-B polynomial $B_2^i(t_i)$ at an instant t_i.

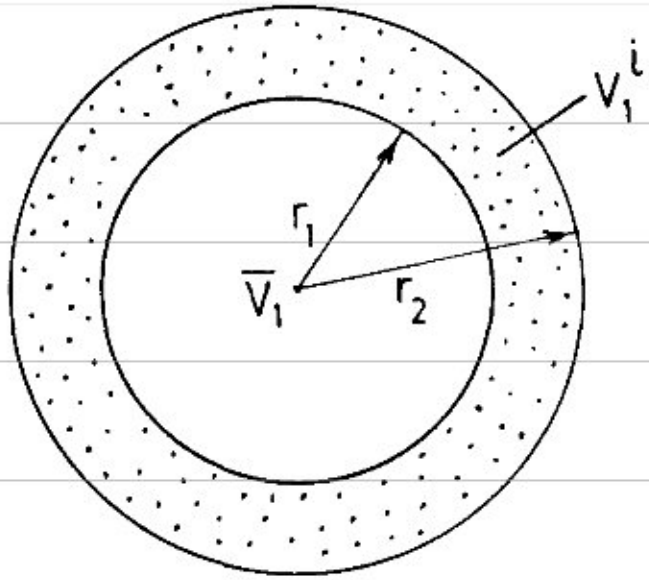

Fig. 2. Annular zone indicates space for $\overline{v}_i$.

inequality (11) tells us that the function $f(t_i) = B_2^i(t_i)$, $i = 1, 2, \ldots, N-2$ can be approximated by $\overline{B}_2(t)$ with an error inequality expressed in (12). v_1^i values thus form an annular ring with center at v_1. The inner and outer radii r_1 and r_2 are, respectively, $(N-1)^2/[2(N-2)]\epsilon_{\min}$ and $2\epsilon_{\max}$. ϵ_l and ϵ_h may lie either outside or on this annular ring. This is shown in Figure 2.

3.3. WORST-CASE APPROXIMATION

It is seen from the previous section that the inequalities (11) and (12) can be used to approximate a gray-tone image row wise (or column wise). During approximation, it may be the case that the inequality (11) does not hold good for all values of i associated with the dimension of the image. Let us consider that the inequality is true for n pixels out of N in each row (or column). Thus the remaining $(N - n + 1)$ pixels can again be approximated over the interval $[0, 1]$. Approximation technique thus may involve decomposition of the entire row (or column) into a number of gray segments. It is to be noted that the inequality (11) is always true for an interval having three pixels irrespective of the inequality (12). This situation is referred as worst-case approximation in the sense of coding because it generates a maximum number of gray segments while doing the approximation. Finally, the end pixel of the row (or column) may remain free. In this case, the same pixel may be considered twice for the worst-case approximation.

TABLE 1
Illustration of Approximation Techniques

Interval	Original Data Points	Approximated Values	Error in Approximation
	24	24.00000000	0.00000000
	27	26.68640137	0.31359863
1	29	28.72960281	0.27039719
	30	30.12960052	0.12960052
	31	30.88640213	0.11359787
	31	31.00000000	0.00000000
	31	31.00000000	0.00000000
	32	31.52640343	0.47359657
2	32	31.88960266	0.11039734
	32	32.08959961	0.08959961
	32	32.12639999	0.12639999
	32	32.00000000	0.00000000
	32	32.00000000	0.00000000
1	31	31.01000023	0.01000023
	31	31.00000000	0.00000000

EXAMPLE. In order to explain the method of approximation, let us consider a sequence of 13 data points. The sequence is 24, 27, 29, 30, 31, 31, 32, 32, 32, 32, 32, 31, 31. Let the maximum and minimum supplied errors ϵ_{max} and ϵ_{min} be, respectively, 1.0 and 0.01. It is seen that the approximation can be done over three intervals. The approximated values in the three intervals, along with the original data points and errors in approximation, are shown in Table 1. It is also seen from the table that the data point at the beginning of one interval is exactly the same as the end point of the previous interval.

The partition of data points into three intervals is controlled by (11). The values of $\overline{v}_1$ for the three intervals are, respectively, 31.52000046, 32.52000046, and 30.52000046. The lower bounds for the absolute value of $(\overline{v}_1 - v_1^i)$ as indicated in (11) are found to be 0.03125, 0.03125, and 0.02 in the three intervals, whereas their upper bounds were found to be the same and equal to 2.0.

4. IMAGE DATA COMPRESSION ALGORITHM

Based on the modified version of the B-B polynomial, we have developed here two algorithms for image data compression. Both the algorithms involve scanning in the horizontal (or vertical) and encode line and arc seg-

ments present in images. Algorithm 1 uses the absolute error criterion of the approximation scheme for a specified error bound while segmenting a row (or a column). Algorithm 2, on the other hand, uses a homogeneous criterion (based on analogy between the small deformation cubic splines and image space curves) to segment rows (columns). These segments are then approximated using the quadratic B-B polynomial and coded suitably.

4.1. ALGORITHM 1

4.1.1. Coding Scheme

An image can be approximated either row wise or column wise. The one which needs fewer number of segments is selected for coding. In the following section, we will explain the bit requirement for the proposed method of coding.

4.1.2. Bit Requirements

Let us consider an image of size $M \times N$ with L number of gray levels. Since there may be a number of gray segments resulting in the process of approximation, each of them can be coded by their corresponding two supports (the starting pixel being known). If the image is coded row wise, then the starting pixels will be the first column, while for column-wise coding, the first row pixels will be the starting pixels.

Since the positional information of the B-B supports is not taken into account for coding, the size of the gray segments becomes important for regeneration of the image. Since the maximum possible size of a segment is N (or M), the bit required for coding a segment is $\log_2 N$(or $\log_2 M$) if coded row wise (or column wise).

It should be noted that each of the gray segments represents a Bezier arc having three supports of approximation, namely, v_0, v_1, v_2 (guiding pixels). Of them, v_1 may not be integer. So we store v_0, the integer part of the data point d_1 (say) at $t = \frac{1}{2}$ of the segment and v_2. The bit required for each of them is $\log_2 L$.

Furthermore, we notice that when $v_0 + v_2 = 2v_2$, the Bezier arc reduces to a straight line segment, and under this situation, we need to store only v_2. In practice, we consider a Bezier arc to be a line segment when $v_0 + v_2 = 2v_1 \pm \delta_1$ is observed, where δ_1 is a small positive integer. Also, if $v_2 = v_0 \pm \delta_1$, then only a single check bit and a sign bit are sufficient to recover the line segment (provided the starting point is known). Two such lines are also merged together to make a single line segment if

$$(v_0)_{\text{1st seg}} + (v_2)_{\text{2nd seg}} = 2(v_1)_{\text{combined seg}} \pm \delta_1.$$

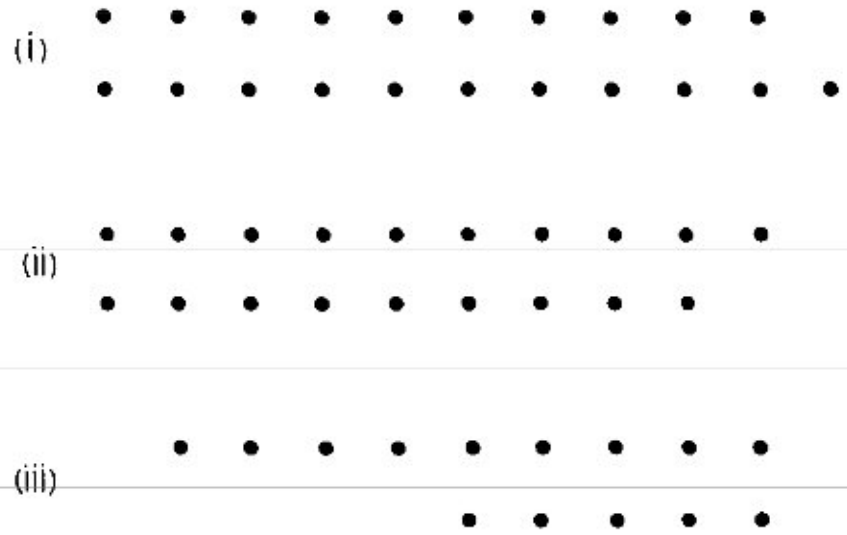

Fig. 3. Coded binary string for Algorithm 1.

Finally, if the end pixel of the row (or column) remains free, then we neither consider it for worst-case approximation nor consider it for coding. During regeneration of this pixel, we simply consider the previous pixel.

Let n_i, r_i be the number of line segments with or without the replaceability condition satisfied, and m_i the number of arcs present in the ith row of the image. If α_i, β_i, and γ_i are the amount of bits required in each of the above respective cases, then,

$$\alpha_i = n_i(\log_2 N \text{ (or } \log_2 M) + 3),$$
$$\beta_i = r_i(\log_2 N \text{ (or } \log_2 M) + 2 + \log_2 L),$$
$$\gamma_i = m_i(\log_2 N \text{ (or } \log_2 M) + 1 + 2\log_2 L).$$

If s is the bits required for the starting pixels, then the total number of bits for an $M \times N$ image for row-wise approximation is

$$T = 1 + s + \sum_{i=1}^{M} \alpha_i + \beta_i + \gamma_i.$$

4.1.3. Decoding

The coded binary string for the image is as shown in Figure 3. Decoding of the string uses the following notations.

The first bit (l_1) denotes the mode of approximation. $l_1 = 0$ for row-wise and 1 for column-wise approximation. The next sequence l_2 of $\log_2 L$ bits represents the gray value of the starting pixel, which is the first entry of the image $M \times N$. The length of l_3 is either $\log_2 N$ or $\log_2 M$, depending

on ($l_1 = 0$ or 1). The bit (l_4) indicates the type of the segment. $l_4 = 0$ for straight line segment and for arc.

If $l_4 = 0$, then the subsequent bit l_5 denotes the replaceability of the end pixel of the line segment. $l_5 = 0$ indicates that the end pixel is replaceable by the beginning pixel of the line segment. l_6 also is 1 bit long and gives the sign for δ_1, and the next sequence in the decoded string is l_{10}. On the other hand, if $l_5 = 1$, then l_6 is absent and the sequence l_7 gives the gray value of the end pixel of the line segment. l_7 has length $\log_2 L$. For $l_4 = 1$ (i.e., the segment is an arc), l_5, l_6, and l_7 are all absent, and l_8, l_9 are the two pixels of the arc. Each of them is $\log_2 L$ bits long. Finally, l_{10} is identical to l_2 for the next row. The same process is then repeated for the entire image.

4.2. ALGORITHM 2

Here each row (column) of pixels has been viewed as a space curve and is segmented depending on the homogeneity among the pixels. Each segment is then approximated by the modified B-B polynomial with a variable error criterion. Since the segments are all homogeneous, approximation for coding can be done with small error. This will, in turn, reduce the smearing significantly. It also makes the approximation faster, and the algorithm becomes parameter independent. Further, unlike algorithm 1, where we considered $\overline{v}_1 = v_1^i$ at $t = 0.5$, the present approximation scheme incorporates

$$\overline{v}_1 = \frac{1}{n} \sum_{i=1}^{n} v_1^i. \tag{13}$$

This, in turn, introduces flexibility in approximating larger segments as compared to algorithm 1.

4.2.1. Small Deformation Space Curve and the Concept of Homogeneity

An image may be considered to be an intensity surface with surface contours representing the space curves along the rows and columns of the image. Note that for any curve Γ, the amount of information contained in it can be represented by its curvature vector $\mathbf{k}$ or by any other related quantity. The curvature vector $\mathbf{k}$ is defined as

$$\mathbf{k} = \frac{\mathbf{dt}}{ds},$$

t being the tangent vector and s being the arc length.

For a curve Γ, with given end points, its bending energy B_e can be written as

$$B_e = \int_\Gamma k^2 \, ds.$$

Here the deformation of the curve is in the direction normal to the axis of the equilibrium position. Therefore, when the x-axis is along the axis of equilibrium position, the deformation may be represented by $z(x)$ and, consequently, we have

$$\begin{aligned} B_e &= \int_\Gamma k^2 \, dx \\ &= \int_\Gamma \frac{[z''(x)]^2}{[1 + (z'(x))^2]^3} \, dx. \end{aligned} \tag{14}$$

For small deformation, $z'(x) \approx 0$ and $B_e \approx \int_\Gamma [z''(x)]^2 \, dx$. Since B_e represents the total energy of the curve, k^2 or $(z'')^2$ represents the energy of the curve at an arbitrary point. Therefore, in an image plane, k^2 will represent the energy of the image space curve at a pixel position.

With the above principle, a curve (a set of pixels along a row or a column) can be considered to be perfectly homogeneous if the bending energy is zero at every pixel position. This is obviously the most stable state of the curve (i.e., without any deformation). Homogeneity decreases with the increase of deformation.

For the purpose of image compression, we are interested in finding the homogeneous segments of pixels in an image, because such segments can be approximated with small amount of error and they do not produce significantly any smearing effect. From the space curve analogy, homogeneous segments of pixels are segments with $z'(x) \approx 0$. However, in practice, it is very difficult to obtain long segments of pixels with zero gradient everywhere. On the other hand, we can find a threshold θ and accept those segments with $z'(x) < \theta$ as the allowable deformation space curves.

In order to determine θ, we consider an analogy between a space curve and a thin elastic physical spline, resting on two simple supports. Without any loss of generality, a physical spline can also be viewed as a thin elastic beam. It is shown in the Appendix that, in order to have a corner-free small deformation cubic spline segment (i.e., homogeneous space curve), one should have

$$\theta < \frac{3}{\cos \alpha},$$

α denoting the direction of tangent vectors at the ends of the segment. The segments of pixels thus obtained with this θ are then approximated by the modified B-B polynomial.

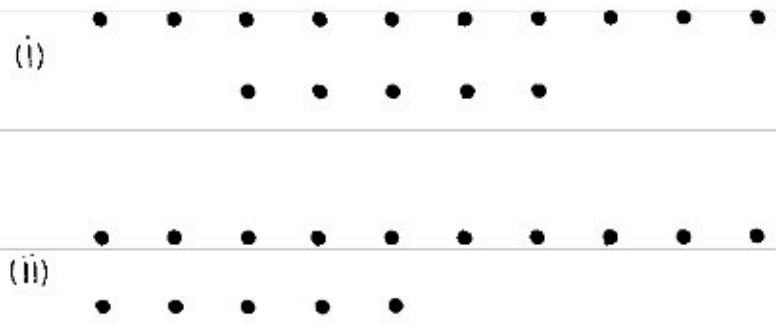

Fig. 4. Typical (connected and nonconnected) homogeneous segments.

4.2.2. Segment Geometry and Bit Requirements

Each row (or column) of pixels in the image is a collection of homogeneous segments. The homogeneous segments obtained in the previous section are either straight line segments or arcs, or a combination of them. Further, the pixel segments in a particular row may or may not be connected to the segments in the just preceding row. Two segments in two successive rows are connected in the x-y projection plane if their end points are connected. This means that only three different connections are possible. The end of a segment s_i with position vector (i, j) in the ith row is connected to a segment s_{i-1} in the preceding row if

$$(i, j) \in \{(i-1, j), (i-1, j+1), (i-1, j-1)\}.$$

Two individually homogeneous segments in two successive rows together are called homogeneous if the column-wise differences between successive pixels are all less than θ. Similar definition holds for two successive homogeneous columns. We consider those homogeneous segments for which their end point difference also is less than θ. With this principle, we get the following four different kinds of segments in a row (or a column):

- connected and homogeneous segment (CH)
- connected and not homogeneous segment (C-NH)
- nonconnected and homogeneous segment (NC-H)
- nonconnected and not homogeneous segment (NC-NH)

Figure 4 shows the nature of typical homogeneous (connected and nonconnected) segments.

Since the approximation scheme is restricted to the scan direction (either horizontal or vertical), a connected vertical line of one or two pixel width will produce a single-pixel- (SP) or two-pixel- (TP) long segment, and they are also taken as valid segments in the coding algorithm. Successive three

segments of one pixel length or a single pixel followed by a two-pixel-long segment, or vice versa, is replaced by an arc segment. In all other cases, single-pixel or two-pixel-long segments are merged to the nearest segment. Note further that a (SP) or a (TP) segment may or may not be connected, and they can also maintain a column-wise difference less than θ in its value from the preceding one. Therefore, for a single-pixel segment from the above nomenclature we get CHSP, C-NHSP, NC-HSP, NC-NHSP type of segments, and for a two-pixel-long segment we get CHTP, C-NHTP, NC-HTP, and NC-NHTP type of segments. With the above geometry of segments, it is clear that a CHSP segment requires minimum bits of information, while NC-NH arc requires maximum bits of information. For two homogeneous arc segments (connected or nonconnected), one arc can always be found from the other provided we know one of the arcs and a difference vector with the differences at known points. This difference vector $\triangle$ can be taken as

$$\triangle = \begin{pmatrix} a_1 - b_1 \\ a_2 - b_2 \\ a_3 - b_3 \end{pmatrix},$$

where a_1, a_2, and a_3 are the known points on an arc, and b_1, b_2, and b_3 are the points on an unknown arc, where the differences are considered.

Figure 4 shows two such arcs. For a CH arc, we therefore need 2 bits for the end connection, 3×3, i.e., 9 bits for the pixel intensity differences, and 3 more bits for its identity, so altogether 14 bits. For a NC-NH arc, the number of bits is $\log_2 N + 3 \log_2 L + 3$ for an $M \times N$ and L-level image, where $\log_2 N$ bits of information are required for the length of the segment, and $3 \log_2 L$ bits for pixel intensities of the three points of the arc. For all other segments, bit requirements are given in Table 2.

4.3. DISCRIMINATING FEATURES OF THE ALGORITHMS

For Algorithm 1:

- Segmentation of pixels in row (or column) does not need any separate algorithm. The approximation scheme itself selects the specific segments.
- The method of approximation depends on the selection of $\epsilon_{\max}$ and $\epsilon_{\min}$. The values of these parameters are the same for all segments in the image. The resulting performance in reconstruction, therefore, is parameter dependent.
- For large $\epsilon_{\max}$, the possibility of the long homogeneous segments of pixels for satisfying the approximation criterion increases. This may

TABLE 2
Bit Requirements for Different Segments

Segment Nature	Bit Requirement	Segment Nature	Bit Requirement
C-H line segments	11	C-NH line segment	$5 + 2\log_L$
C-H arc segment	14	C-NH arc segment	$5 + 3\log_2 L$
C-HSP	7	C-NHSP	$4 + \log_2 L$
C-HTP	10	C-NHTP	$4 + 2\log_L$
NC-H line segment	$3 + \log_2 N$	NC-NH line segment	$3 + \log_2 N + 2\log_2 L$
NC-H arc segment	$12 + \log_2 N$	NC-NH arc segment	$3 + \log_2 N + 3\log_2 L$
NC-HSP	$5 + \log_2 N$	NC-NHSP	$2 + \log_2 N + \log_2 L$
NC-HTP	$8 + \log_2 N$	NC-NHTP	$2 + \log_2 N + 2\log_2 L$

introduce visual disparity between the original and the reconstructed segments. This, in turn, may affect the overall picture quality.

For Algorithm 2:

- A separate algorithm selects only those segments which are homogeneous in some cases. For this, an image has been considered as an intensity surface and the homogeneity concept of pixel segments has been viewed as a small segment space curve on this intensity surface.
- Different homogeneous segments in an image are approximated with different values of $\epsilon_{\max}$, which are determined automatically in the process of approximation. The performance of the algorithm, therefore, does not depend on $\epsilon_{\max}$ as in Algorithm 1.
- Since the approximation is done over the homogeneous segments of pixels, low values of $\epsilon_{\max}$ are able to approximate all the segments. This, in turn, significantly reduces the smearing effect.

5. REGENERATION

Reconstruction of the image during decoding is done using the quadratic B-B polynomial. We use here the recursive computation algorithm based on Newton's forward difference scheme as described in [2]. Let $y = at^2 + bt + c$ be a polynomial representation of (4), where the constant parameters a, b, and c are determined by the three pixels (two end pixels and one mid pixel) of the arc segment. 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 the parameter t ranges from 0 to 1. Let the incremental value be q. Then the corresponding y values will be c, $aq^2 + bq + c$, $4aq^2 + 2bq + c$,

$9aq^2 + 3aq + c, \ldots$. It can be observed from [2] that

$$\Delta^2 y_j = 2aq^2 \quad \text{and} \quad y_{j+2} - 2y_{j+1} + y_j = 2aq^2, \qquad j \geq 0.$$

This leads to the recurrence formula

$$y_2 = 2y_1 - y_0 + 2aq^2 \tag{15}$$

that involves just three additions to get the next value from the two preceding values at hand.

Since the gray segment size is known, the increment q can be obtained from

$$q = \frac{1}{\text{segment size} - 1}.$$

The regenerated gray value y_2 can therefore be determined from (15).

6. RESULTS AND DISCUSSION

An attempt is made here to demonstrate an application of one-dimensional quadratic Bezier–Bernstein polynomial approximation in coding gray-tone images. Drawbacks in using the conventional way of approximation have been examined and a modification is then introduced in order to make it useful in image data compression. Based on the modified concept, two different algorithms have been formulated.

Note that Algorithm 1 may produce smearing for large values of $\epsilon_{\max}$, because with the increase in the value of $\epsilon_{\max}$, the possibility of the long homogeneous segments of pixels for satisfying the approximation criterion increases. As a result, visual disparity may arise. However, this smearing is almost absent in Algorithm 2. Also, the picture quality and the compression ratio are better compared to those in Algorithm 1. It is seen from Table 3 that the compression ratio is of the order of 0.8 bit/pixel.

The approximation technique described here is different from the conventional least-square method of approximation. Instead of minimizing the global squared sum of errors, it controls an absolute maximum error for each data point. It should be noticed in this context that if the pixels of a segment have low intensity variation, then the techniques based on conventional quadratic least-square and the quadratic B-B polynomial approximation will produce the same result. Since the proposed method of approximation controls an absolute local error instead of global sum of errors, it is expected that even for moderate variation of intensity within data points, the proposed method will produce better results. Also, given an error term, the conventional least-square technique does not ensure that

TABLE 3
Results

Algorithm 1				Algorithm 2		
Images	Mode of Approx.	c_{max}	Compression Ratio	Images	Mode of Approx.	Compression Ratio
Fig. 7b	row-wise	6	4.87			
Fig. 7c	,,	10	6.04	Fig. 10a	row-wise	6.33
Fig. 7d	,,	14	7.32			
Fig. 8b	row-wise	4	3.80			
Fig. 8c	,,	6	4.44	Fig. 10b	row-wise	6.58
Fig. 8d	,,	8	4.76			
Fig. 9b	row-wise	4	2.34			
Fig. 9c	,,	5	2.54	Fig. 10c	row-wise	3.80
Fig. 9d	,,	6	2.82			

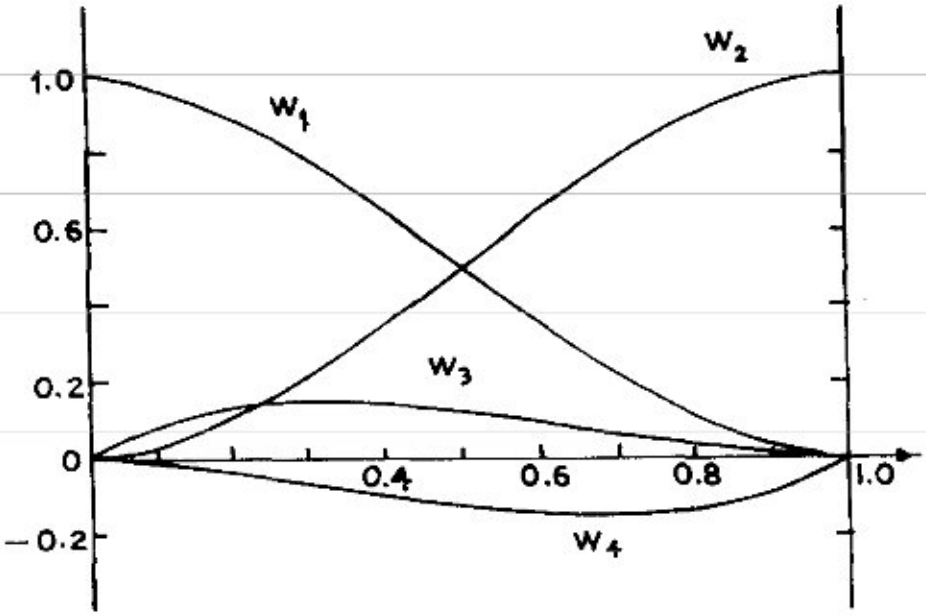

Fig. 5. Behavior of weighting (blending) functions for a cubic spline.

all the data points will satisfy the error criterion, whereas in the proposed method this is not the case. Furthermore, it is not necessary to compute any functional distance (unlike the least-square technique [5]) to justify the goodness of approximation because the error term itself quantifies this.

Note further that our intention here is to demonstrate an application of one-dimensional B-B polynomial in the scan direction for image data compression. Since the algorithms consider scanning in only one direction,

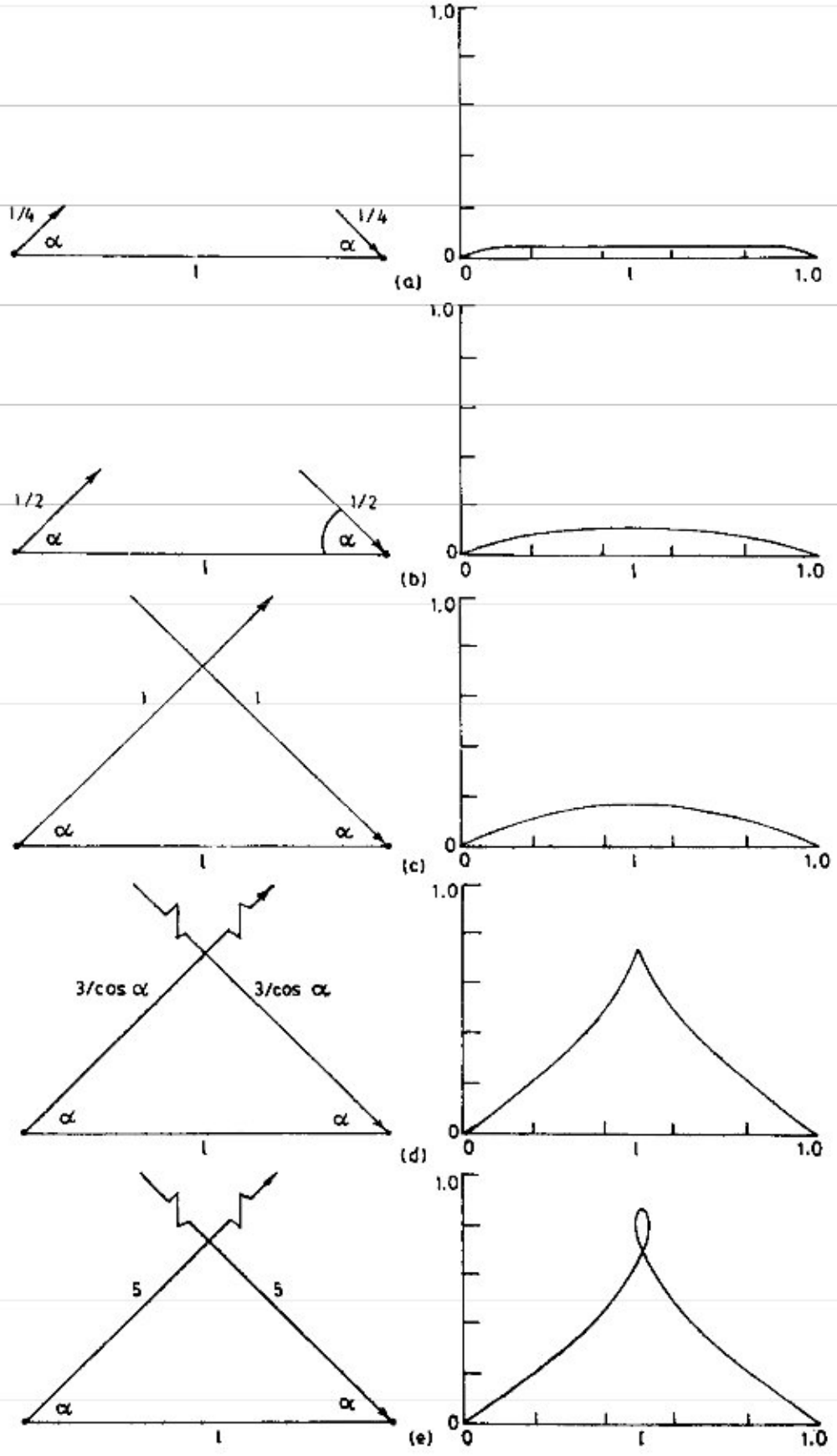

Fig. 6. Effect of tangent vector magnitude on cubic spline segment shape $\alpha = \frac{\pi}{4}$ (a) $\frac{1}{4}$; (b) $\frac{1}{2}$; (c) 1; (d) $\frac{3}{\cos\alpha}$; (e) $\frac{3}{2}$.

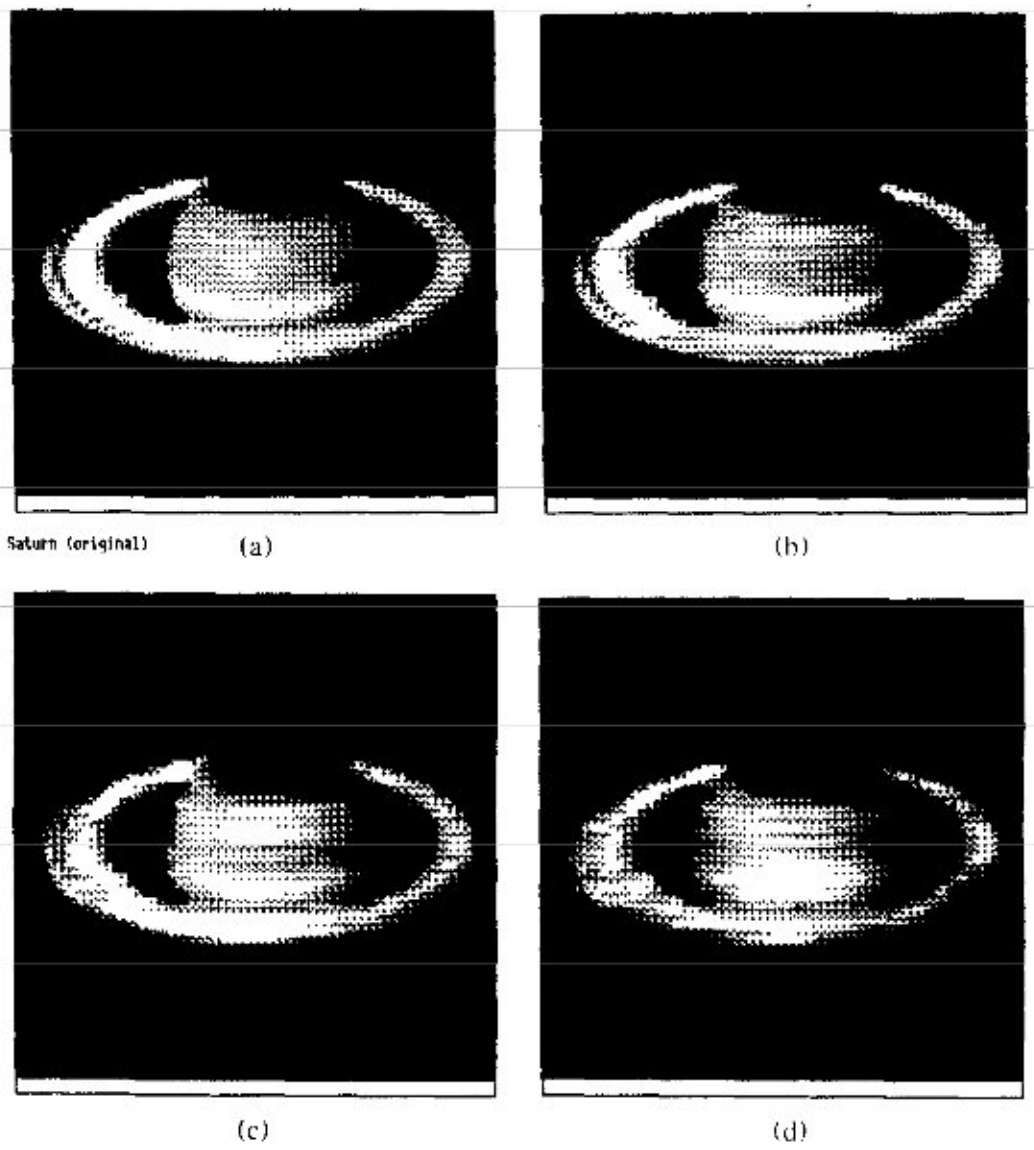

Fig. 7. (a) Input saturn image. (b)–(d) Regenerated output images with (b) $\epsilon_{\max} = 6$, (c) $\epsilon_{\max} = 10$, and (d) $\epsilon_{\max} = 14$ by Algorithm 1.

the scheme is fast and simple in hardware implementation. However, it is needless to mention that the two-dimensional approximation always provides a better compression ratio than the corresponding one-dimensional approximation.

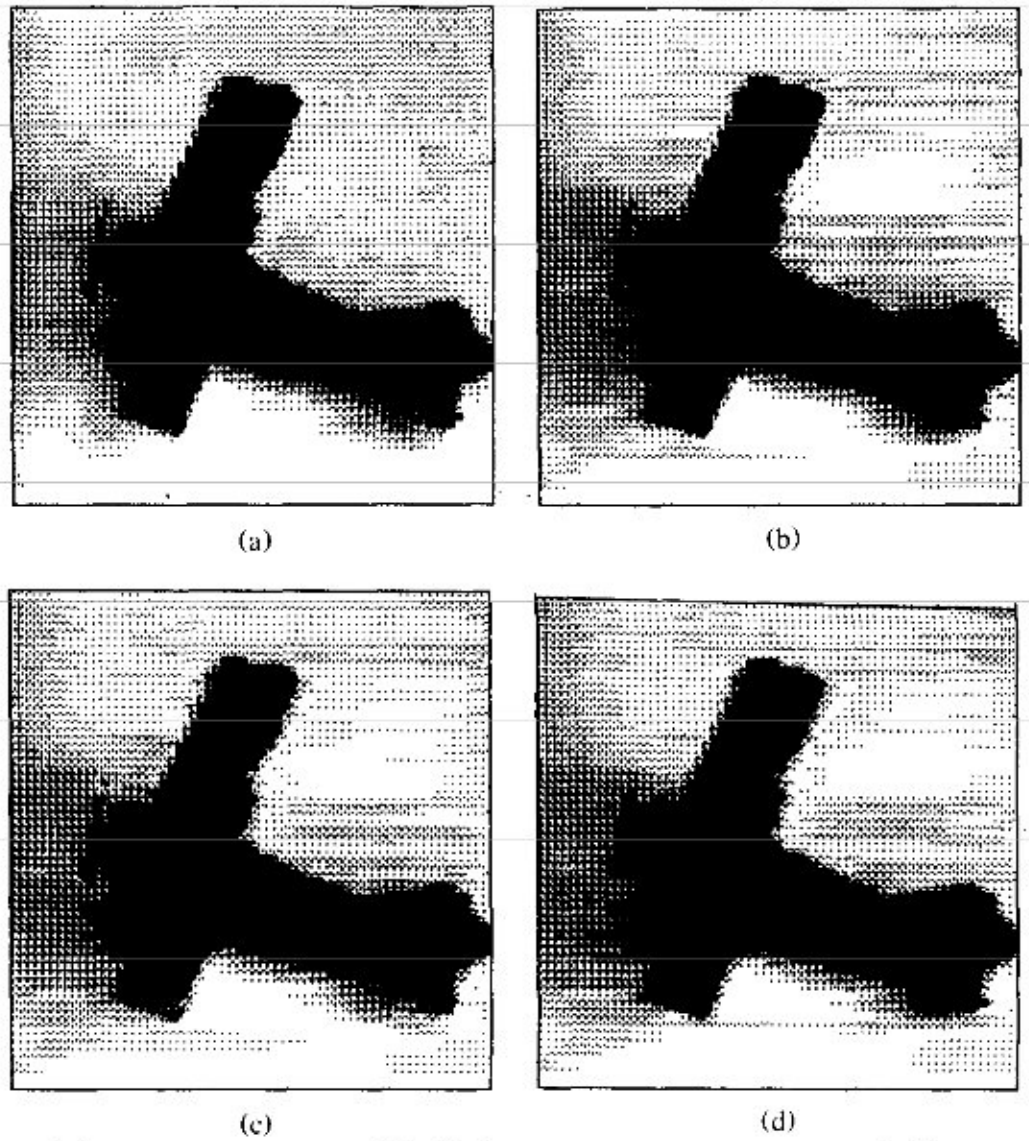

Fig. 8. (a) Input biplane image. (b)–(d) Regenerated output images with (b) $\epsilon_{\max} = 4$, (c) $\epsilon_{\max} = 6$, and (d) $\epsilon_{\max} = 8$ by Algorithm 1.

APPENDIX

We know that for small deformation, the Euler equation for bending moment $M(x)$ of a beam can be written as

$$M(x) = \frac{YI}{R(x)},$$

where $1/R(x) = k(x) \approx z''(x)$ [from (14)]. Y is the Young's modulus depending on the material of the beam, I is the moment of inertia for the cross section of the beam, and $R(x)$ is the radius of curvature of the beam. Assuming simple supports, the bending moment is known to vary linearly

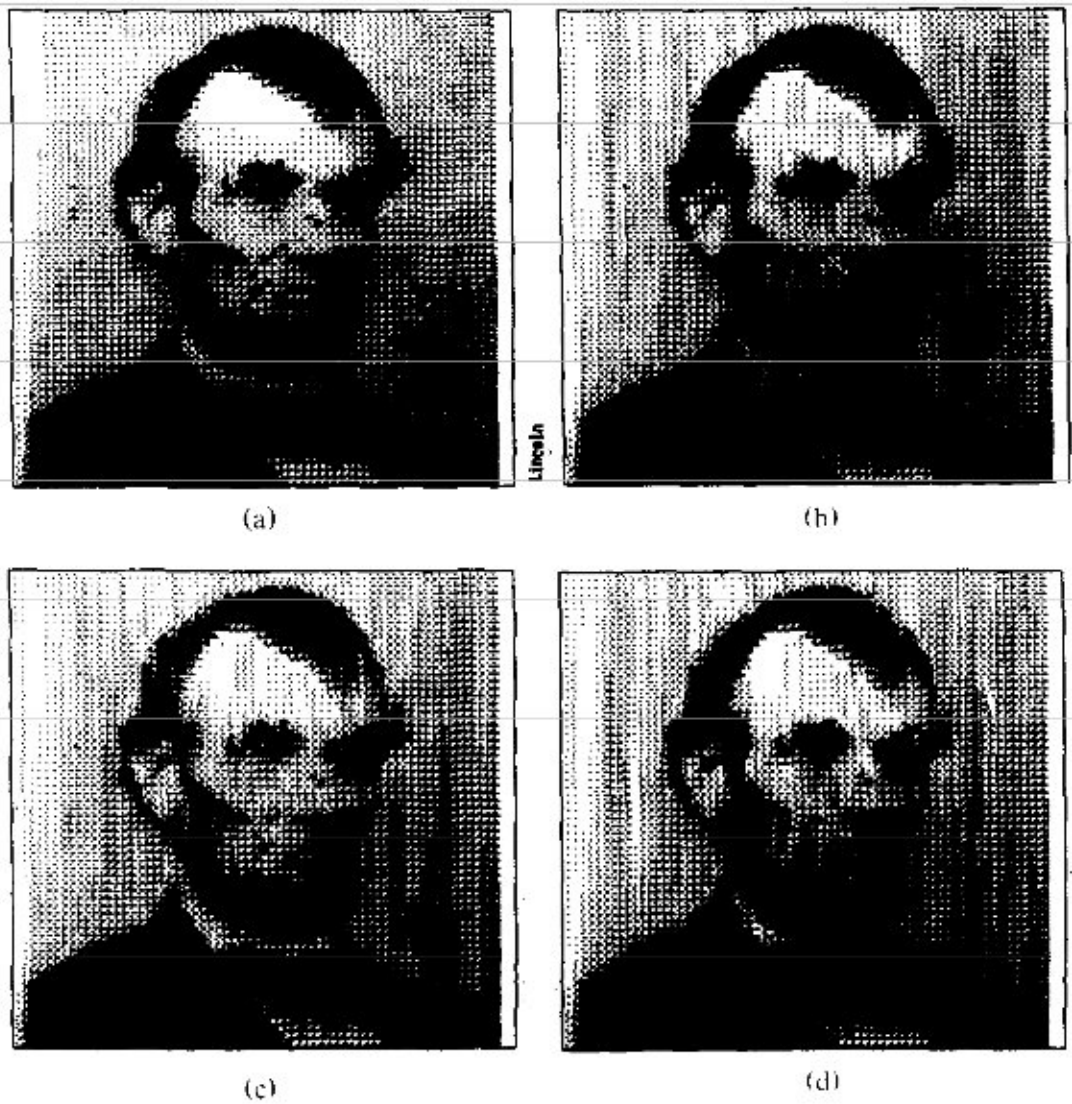

Fig. 9. (a) Input Lincoln image. (b)-(d) Regenerated output images with (b) $\epsilon_{\max} = 4$, (c) $\epsilon_{\max} = 5$, and $\epsilon_{\max} = 6$ by Algorithm 1.

[6], and we therefore put $M(x) = ax + b$. With this, $z''(x) = (ax + b)/YI$, which yields

$$z(x) = \int \left\{ \int \frac{ax + b}{YI} dx \right\} dx$$
$$= a_1 + a_2 x + a_3 x + a_4 x. \tag{16}$$

This equation indicates that a small deformation curve can always be represented by a cubic spline curve. In the image plane, a homogeneous segment of pixels can therefore be viewed as a cubic spline segment, and therefore it can be extracted based on the properties of the cubic spline function. As the axis of the stable position of all such segments may not always correspond

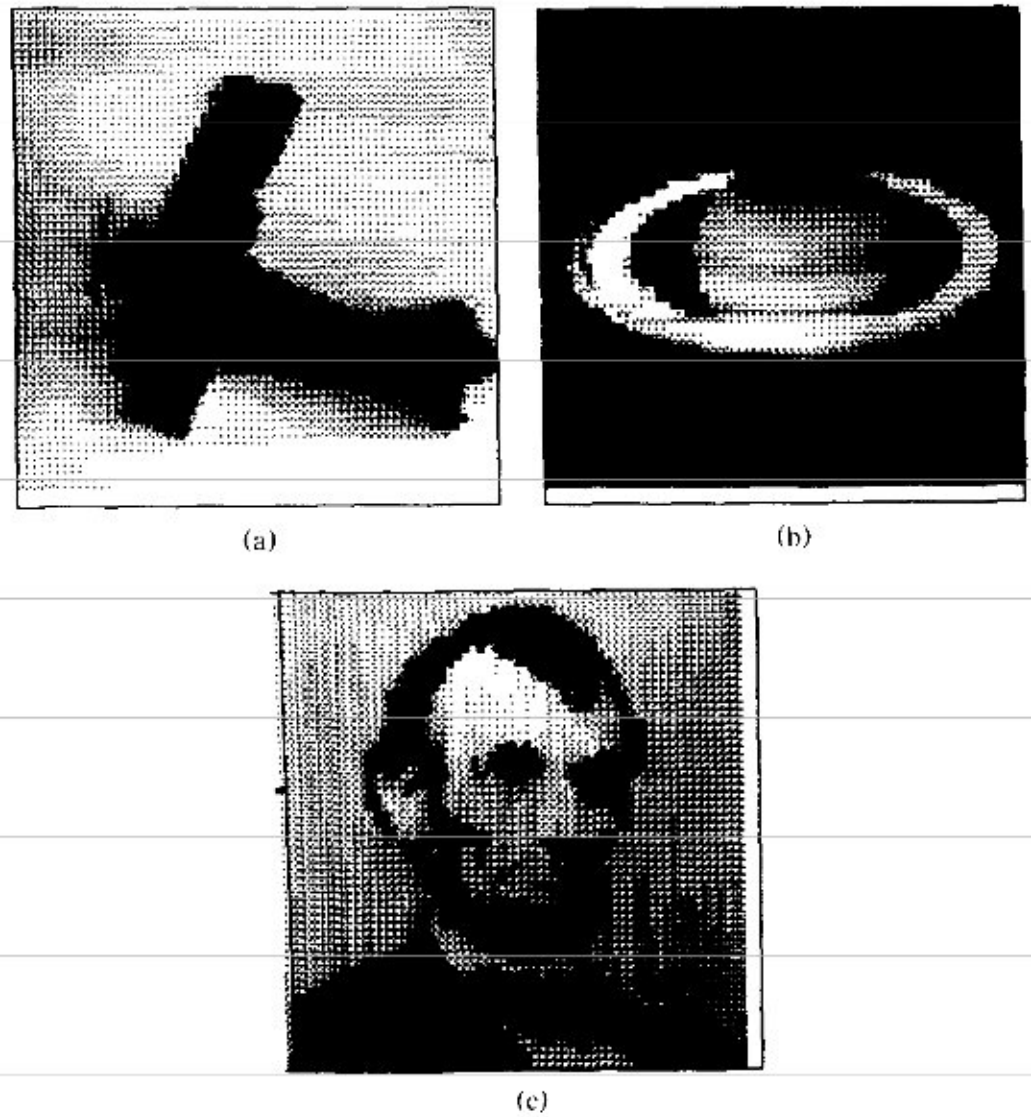

Fig. 10. (a)–(c) Regenerated output images by Algorithm 2.

to the x-axis, it is wise to consider an axis-independent representation. For this, we write

$$z(t) = \sum_{i=1}^{4} B_i t^{i-1}, \qquad t_1 \le t \le t_2,$$

where t_1 and t_2 are the parameter values at the end positions of a segment and B_i's are the coefficients. Using the boundary conditions, this can be written as

$$z(t) = (w_1 \; w_2 \; w_3 \; w_4) \begin{pmatrix} p_1 \\ p_2 \\ p_1' \\ p_2' \end{pmatrix}.$$

p_1, p_2 and p_1', p_2' are the end positions of the curve and their derivatives (tangent vectors) at these positions, respectively, whereas w_i's are the weighting functions.

Figure 5 shows the behavior of these weighting functions. Since the difference in magnitudes of w_1 and w_2 is more than that of w_3 and w_4, we can say that the end position vectors have more influence than the tangent vectors on the value of $z(t)$.

Although the tangent vectors have less influence on the value of $z(t)$, they have a strong impact, as described below, on the smoothness of $z(t)$ [7]. Figure 6 shows a single plane symmetric spline segment with constant tangent vector direction (α_0) and varying magnitudes (represented by the length of the tangent vector). When the magnitude is a small fraction of the chord length l, the curve is convex at the ends and lies inside the triangle formed by the chord and the tangent vectors, as shown in Figure 6(d). With the increase of magnitude, the curve eventually becomes concave at the ends and lies outside the triangle. A corner is developed in the curve when the tangent vector magnitude is $3/\cos\alpha$ [7]. For magnitudes larger than this, a loop is formed. This is shown in Figure 6(e).

From the above behavior, it is evident that a symmetric cubic spline curve is distortion- (corner or loop) free or, in other words, smoothness of such curves is preserved when the tangent vector magnitude is kept below $m_t < 3/\cos\alpha$. This feature of tangent vector magnitude can therefore be used to segment an arbitrary space curve in the image plane.

REFERENCES

1. P. E. Bezier, Mathematical and practical possibilities of unisurf, in *Computer Aided Geometric Design*, E. R and R. F. Risenfeld, Eds., Academic, New York, 1974.
2. S. N. Biswas, S. K. Pal, and D. Dutta Majumder, Binary contour coding using Bezier approximation, *Pattern Recognition Letters* 8:237–249 (1988).
3. S. N. Biswas and S. K. Pal, Approximate coding of digital contours, *IEEE Transactions on System, Man and Cybernetics* 18:1056–1066 (1988).
4. N. Macon, *Numerical Analysis*, Wiley, New York, 1963.
5. M. Kunt, M. Benard, and R. Leonardi, Recent results in high compression image coding, *IEEE Transactions on Circuits and Systems* CAS-34:1306–1336 (1987).
6. A. Higdon, E. Ohlsen, W. Stiles, and J. Weese, *Mechanics of Materials*, Wiley, New York, 1967.
7. D. J. Rogers and J. A. Adams, *Mathematical Elements For Computer Graphics*, McGraw-Hill, New York, 1990.

Received 1 January 1994; revised 1 June 1994