A Growing Bezier Curve for Efficient Curve Approximation

Asif Masood
Department of Computer Science, Military College of Signals
National University of Sciences and Technology, Islamabad, Pakistan
amasood@mcs.edu.pk

Abstract

A curve approximation technique, based on growing quadratic and cubic Bezier, is presented in this paper. The approximating curve starts growing along the given curve till it reaches to the end. It determines the suitable location of control points in its way. Some important features of proposed algorithm, that distinguish it from previous approaches, may include use of standard Bezier curves, efficient computation of approximating curve by exploiting the properties of Bezier curves, efficient method of calculating the approximation error, and incremental growth of curve. Results of proposed algorithm are compared with different approaches in terms of compression ratio, approximation error, and computation efficiency. Proposed algorithm can lead to various applications in computer aided design, computational geometry, and computer vision.

Keywords: Curve Approximation, Growing Bezier, Bernstein Polynomials, Control Points, Approximation Error, Curve subdivis on

Introduction

Curve approximation is an important area of research in CAD and CAGD. These algorithms find a set of data points that can accurately represent the given curve. Two types of data points are the focus of any curve approximation method, namely interpolating and approximating data points. Interpolating points are picked from the original curve and resulting curve is passed through them in order [3]. These points are easy to find and is the target of most curve approximation techniques. A danger with interpolation is that the fitting function may tend to exhibit unwanted fluctuations. This problem is particularly common if the fitting function is a polynomial defined over the entire region of interest, and becomes more severe as the number of data points increases. Piecewise cubic splines are often used to interpolate a large number of data points because they reduce the computational requirements and numerical instabilities that arise with higher degree curves. Plass and Stone [17] used piecewise parametric cubic polynomial for end point interpolation with tangent vector specification. Razdan [20] highlights the use of arc length and curvature characteristics of the given curve to extract the interpolation points. Various other techniques were considered in [7, 8, 18, 19 ,25].

On the other hand, approximating curve is fitted to the general path without necessarily passing through the data points [4]. It involves significantly fewer coefficients than the corresponding interpolants. This approach is much less liable to produce unwanted fluctuations and can provide better approximation results. Proposed technique falls in this category of curve approximation. Itoh and Ohno [12] used least square method to find intermediate control points of approximating cubic Bezier curve and end points were located by computing the intersection of adjoining curves. Sohel et al. [26] proposed shape description algorithm using Bezier curves. They used fixed length coding scheme to find the control points. Masood, et al. [15] proposed curve approximation with quadratic B-splines by analyzing their opening angle plots.

Approximating data points are computationally heavy to determine [23]. Most of the time is consumed in calculating and minimizing the error from original curve e.g. least square fit [12]. Some of the researches [14,21,23] have proposed techniques to minimize this time. Sohel et al. [21] presented enhanced Bezier curve model, with no increase in order of computational complexity, to reduce the distance between the Bezier curve and its control polygon. It was ultimately applied to reduce distortion of approximating curves. Sarfraz and Masood [14] proposed a method to find intermediate control points along the end point tangents. Similarly, Sarfraz, et al. [23] calculated the ratio between two intermediate control points of cubic Bezier and used it to estimate the position of control points. This caused reduction of computation time in subsequent phase of approximation.

In above techniques, although the computation time was considerably reduced, by minimizing the number of iterations involved in error calculation, but still most of the computations were used in calculating and reducing the approximation error. Reduced set of data points without any compromise to the quality of approximation and an efficient process are some of the properties which make this method superior to others techniques. Proposed algorithm is progressively developed in this paper. Some properties of Bezier curves, useful to compute the position of intermediate control points, are described in section 2. Computation of approximating (quadratic and cubic) curve is discussed in section 3. A technique of growing Bezier is explained in section 4. Analysis of results including comparison with other techniques is given in section 5. Finally, section 6 concludes this presentation.

Bezier curves and its properties

Bezier curves are used in computer graphics to produce curves which appear reasonably smooth at all scales. The mathematics of these curves is classical, but it was a

French automobile engineer Pierre Bezier [4] who introduced their use in computer graphics. Bezier curves are simple and efficient to implement and have number of properties [2] which make them highly useful and convenient for curve and surface design [6]. That's why these are widely available in various CAD systems, vector graphics, animations, drawing and painting packages [3]. These are extensively used in curve approximation as well [12, 14, 17, 22, 23, 26].

In general, a Bezier curve section can be fitted to any number of control points. The number of control points to be approximated and their relative position determine the degree of Bezier polynomial. A Bezier curve can be specified with boundary conditions [2], with a characterizing matrix [9], or with blending functions [9]. For general Bezier curves, the blending function specification is the most convenient. For a given n+1 control point positions $P_k = (x_k,y_k)$, with k varying from 0 to n. These coordinate points can be blended to produce the curve $C(u)$, which describes the path of an approximating Bezier polynomial between P_0 and P_n. It can be given as:

$$C(u) = \sum_{k=0}^{n} P_k B_{kn}(u), \qquad 0 \le u \le 1 \tag{1}$$

The Bezier blending function $B_{kn}(u)$ are the Bernstein polynomials [6].

$$B_{kn}(u) = \binom{n}{k} u^k (1-u)^{n-k}, \tag{2}$$

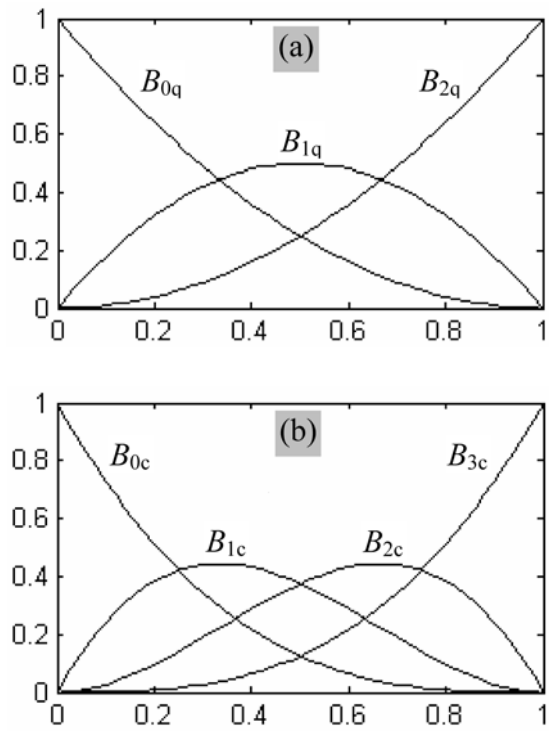

Fig. 1. Plot of blending functions for (a) Quadratic Bezier and (b) Cubic Bezier

As a rule, a Bezier curve is a polynomial of degree one less than the number of control points used. Three points generate a parabola, four points a cubic curve and so forth. Quadratic and cubic splines are used in most applications [2,6,9]. This gives reasonable design flexibility while avoiding the increased calculations needed with higher-order polynomials. Quadratic and cubic Bezier curves are mainly focused in proposed technique as well. The plot of blending functions for a quadratic and cubic curves are given in Fig. 1. Some of their useful properties, which are subsequently exploited to obtain an approximating curve, are discussed below.

- The vector tangent to the Bezier curve at the start (and stop) is parallel to the line connecting the first two (and last two) control points.
- Bernstein polynomials of Bezier curve at any time interval u sum up to 1.

$$\sum_{k=0}^{n} B_{kn}(u) = 1 \tag{3}$$

- A cubic Bezier curve can simply be defined as:

$$C_c(u) = P_0 B_{0c}(u) + P_1 B_{1c}(u) + P_2 B_{2c}(u) + P_3 B_{3c}(u) \tag{4}$$

Where B_{0c}, B_{1c}, B_{2c}, B_{3c} are the Bernstein polynomials which does not depend upon the position of any control point(s). From eq. 4, it can be stated that each polynomial combined with respective control points influence the overall shape of a Bezier curve. If the position of any control point is known, its influence on the Bezier curve may be excluded. For example:

$$P_0 B_{0c}(u) + P_1 B_{1c}(u) + P_2 B_{2c}(u) = C_c(u) - P_3 B_{3c}(u) \tag{5}$$

- Effect of first (P_0) and last (P_n) control points and their respective Bernstein polynomials (B_0 & B_n) can be removed from any Bezier curve.
- The Bezier curve interpolates the first (P_0) and last (P_n) control point. In other words, it starts from the first control point and stops at the last.
- For a cubic Bezier curve at u = 0.5, the Bernstein polynomial $B_{1c} = B_{2c}$.
- For a given quadratic Bezier curve, if the two control points are known position third control point can be calculated by.

$$P_1 = \frac{C_q(u) - P_0 B_{0q}(u) - P_2 B_{2q}(u)}{B_{1q}(u)} \tag{6}$$

Bezier approximation

Spline curves are defined by the set of given control points. In other words, control points are given and a spline curve is determined. This can be done very efficiently (eq. 1) but

it can become very challenging if this procedure is reversed *i.e.* finding control points from a given curve. This is very important area of research and is the main objective of any curve approximation technique [20]. Calculated control points from a given curve are generally referred as data points. A spline curve passing through these data points is an approximated curve.

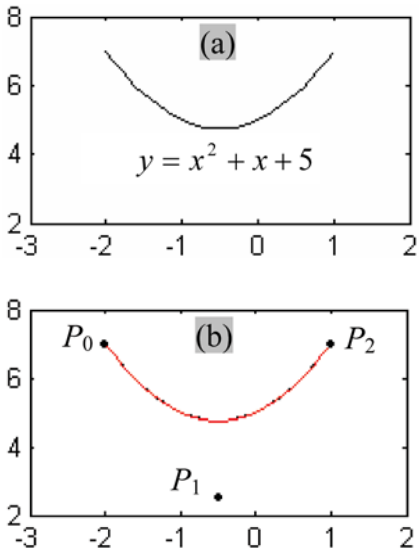

Fig. 2. Approximation with quadratic Bezier (a) Given quadratic curve, (b) Approximated curve (red) over original (black)

Data points of quadratic Bezier

A quadratic Bezier curve is generated with three control points $P_k = (x_k, y_k)$, with k varying from 0 to 2. These coordinate points are blended to produce the quadratic curve $C_q(u)$ with u varying from 0 to 1. It can simply be written as:

$$C_q(u) = P_0 B_{0q}(u) + P_1 B_{1q}(u) + P_2 B_{2q}(u) \qquad 0 \le u \le 1 \tag{7}$$

Where B_{0q}, B_{1q}, and B_{2q} are the respective blending functions of quadratic curve which can be given as:

$$B_{0q}(u) = (1-u)^2$$
$$B_{1q}(u) = 2u(1-u) \tag{8}$$
$$B_{2q}(u) = u^2$$

As shown in eq. 8 that the blending functions are Bernstein polynomials and only the function of interval u. Fig 1(a) shows plot of these blending functions. Here, we need to find three control points (data points) of given quadratic Bezier curve. From eq. 7, two control points P_0 and P_2 are the two end points of quadratic curve which are known. The only unknown in eq. 7 is control point P_1, which can be determined as:

$$S_q(u) = \frac{C_q(u) - P_0 B_{0q}(u) - P_2 B_{2q}(u)}{B_{1q}(u)} \qquad 0 < u < 1 \tag{9}$$

where $S_q(u)$ is the array of points, representing the unknown control point (P_1). This array is called as spread of approximating quadratic Bezier curve. Area of spread is proportional to the difference between two curves (*i.e.* original and computed). Note that the spread $S_q(u)$ is not computed for u = 0 & 1. From $S_q(u)$, position of control point (P_1) is obtained by taking its arithmetic mean. It can be given as:

$$P_1 = mean(S_q(u)) \tag{10}$$

Fig. 2(a) is the plot of quadratic curve $y = x^2 + x + 5$ at $-2 \le x \le 1$ and Fig. 2(b) shows its approximation using above method. Fig 2 shows an accurate approximation using this procedure. However, results can be very different if the given curve is not quadratic. Fig 3(a) shows a cubic Bezier curve taken for approximation using the same method. The spread is shown by green color in Fig. 3(b). Black spot at the center of a green line shows the selected position of P_1. Approximating (red) curve is drawn over the original and one can see that it lacks in accuracy.

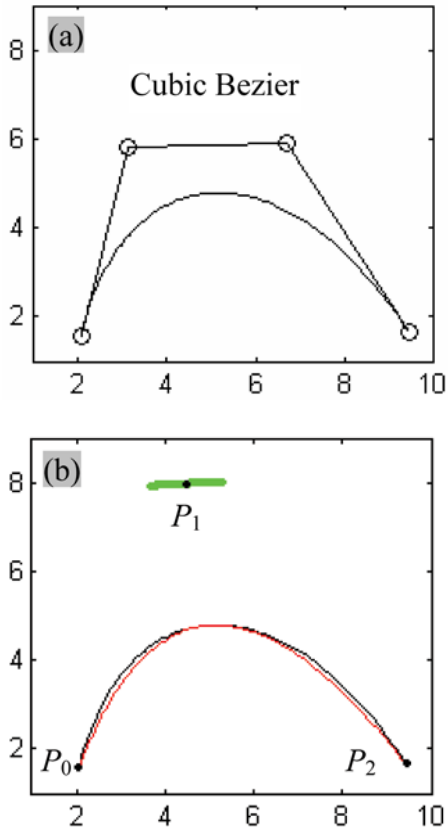

Fig. 3. Approximation with quadratic Bezier. (a) Given cubic Bezier curve, (b) Approximated curve (red) over original (black)

Difference between the two curves (*i.e.* original and computed) is the approximation error. The approximated curve is unacceptable if the maximum error (maximum distance between two curves) error goes beyond specified limits. Researchers [5,8,10,21] have proposed different methods to compute this error. Generally, these methods involve heavy computations. A curve approximation algorithm may need to compute this error again and again, before validating the accuracy of approximation. In other words, error calculation takes the major part of computation time in curve approximation. In proposed algorithm, spread represents the error of approximation and it can give exact value of error at any point along the curve. The error of approximating quadratic curve may be obtained as:

$$E_q(u) = \left\| S_q(u) - P_1 \right\| \times B_{1q}(u) \qquad (11)$$

$$0 < u < 1$$

The norm represents the Euclidian distance. Maximum error of curve from original can be given as:

$$ME = \max_{u=0}^{1} \{E_q(u)\} \qquad (12)$$

The error (*ME*) beyond specified threshold indicates that the computed curve is not accurate enough. There are two solutions to this problem. First, to use a cubic or other higher polynomial curves and second, to use piecewise approximating curve. Piecewise approximation, with growing Bezier, is covered in section 4. Using cubic Bezier for approximation is a better choice than quadratic as it can approximate more flexible curves with lesser data points and without much increase to computation complexity. Approximation with cubic Bezier can be employed using similar methodology (as shown in section 3.2). Other higher polynomials are not used to avoided heavy calculations.

Data points of cubic Bezier

A cubic Bezier curve is defined by four control points $P_k = (x_k, y_k)$, with k varying from 0 to 3. The cubic curve $C_c(u)$, can be written as:

$$C_c(u) = P_0 B_{0c}(u) + P_1 B_{1c}(u) + P_2 B_{2c}(u) + P_3 B_{3c}(u) \qquad (13)$$

$$0 \le u \le 1$$

Where B_{0c}, B_{1c}, B_{2c} and B_{3c} are the respective blending functions of cubic Bezier, which can be obtained as:

$$B_{0c}(u) = (1-u)^3$$
$$B_{1c}(u) = 3u(1-u)^2$$
$$B_{2c}(u) = 3u^2(1-u)$$
$$B_{3c}(u) = u^3 \qquad (14)$$

Fig. 1(b) shows plot of these blending functions. Position of control points does not cause any effect to the blending functions. Therefore, if the position of any control point is known its effect from the overall curve may be eliminated. As we know that end points of a cubic Bezier curve are the two control points (P_0 and P_3). Thus, we can remove effect of P_0 and P_3 from a given curve as:

$$P_1 B_{1c}(u) + P_2 B_{2c}(u) = C'(u) \qquad (15)$$

where

$$C'(u) = C_c(u) - P_0 B_{0c}(u) - P_3 B_{3c}(u) \qquad (16)$$

By observing the plot of blending function (Fig. 1(b)), we can conclude that $B_1 = B_2$ at u=0.5. It can be stated as:

$$B_{0.5} = B_{1c}(0.5) = B_{2c}(0.5) \qquad (17)$$

From eq. 15 and eq. 17

$$P_1 + P_2 = C'' \qquad (18)$$

where

$$C'' = \frac{C'(0.5)}{B_{0.5}} \qquad (19)$$

By simultaneously solving the eq. 15 and eq. 18, we have:

$$\left.\begin{aligned} S_{1c}(u) &= \frac{C'(u) - B_{2c}(u) \times C''}{B_{1c}(u) - B_{2c}(u)} \\ S_{2c}(u) &= C'' - S_{1c}(u) \end{aligned}\right\} \text{ at } 0 < u < 1 \qquad (20)$$

Where S_{1c} and S_{2c} are the two spreads (array of points), which represent the control point P_1 and P_2 respectively. Note that the point P_1 and P_2 are ignored (not computed) for u = 0, 0.5, & 1, as it will produce zero on the denominator. The position of P_1 and P_2 is obtained by taking the mean. It is given as:

$$P_1 = mean(S_{1c}(u))$$
$$P_2 = mean(S_{2c}(u)) \qquad (21)$$

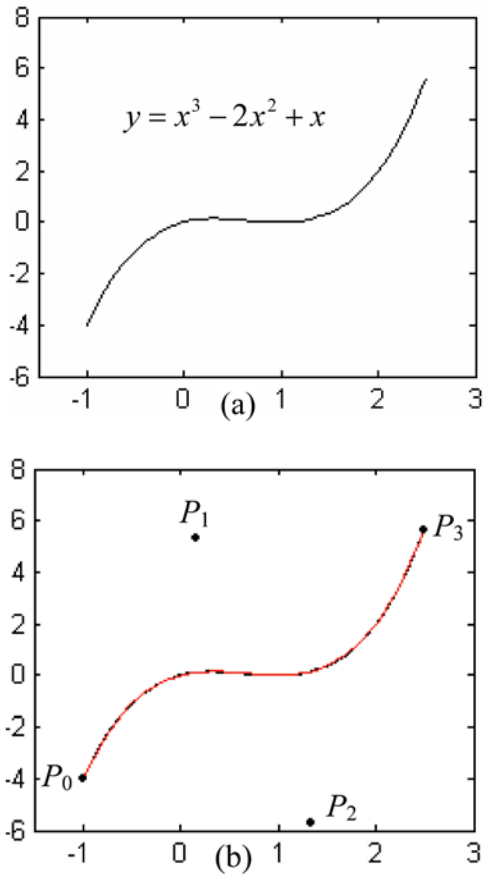

Fig. 4. Curve fitting to a cubic curve. (a) Given cubic curve. (b) Fitted cubic Bezier (red) over original (black)

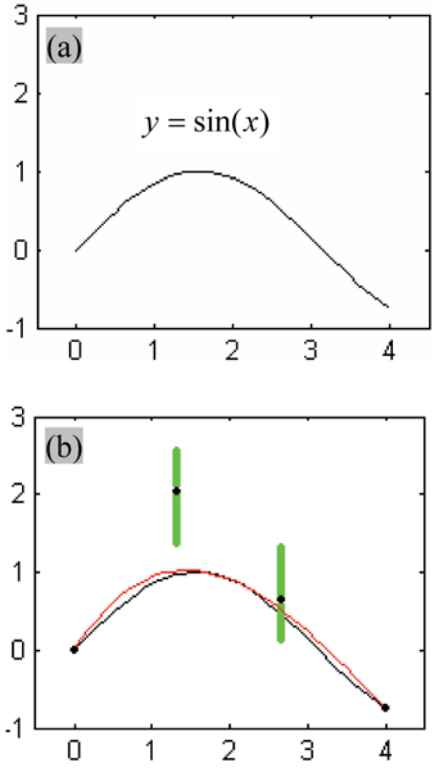

Fig. 5. Approximation with cubic Bezier. (a) Given sin(x) curve, (b) Approximated curve (red) over original (black)

This procedure results is accurate approximation if the given curve is cubic. This can be seen in Fig 4. Fig. 4(a) shows the given cubic curve $y = x^3 - 2x^2 + x$ at $-1 \le x \le 2.5$ and Fig. 3(b) shows the fitted curve (red) drawn over original (black). Respective control points of computed curve are also shown in this figure. Accuracy of computed curve may deteriorate if the given curve is not a cubic. For example, sin-curve $y = \sin(x)$ at $0 \le x \le 4$, shown in Fig. 5(a). Its approximation is shown in Fig. 5(b). Again, the approximation error between two curves can be calculated from the spread S_{1c} and S_{2c}. This can be given as:

$$E_c(u) = abs\left(\begin{pmatrix}\left(\|S_{1c}(u) - P_1\| \times B_{1c}(u)\right) - \\ \left(\|S_{2c}(u) - P_1\| \times B_{2c}(u)\right)\end{pmatrix}\right) \tag{22}$$

The maximum error between two curves is used to validate the approximated curve. Computed curve is accepted only if the maximum error (*ME*) does not go beyond specified threshold error. Maximum error is obtained as:

$$ME = \max_{u=0}^{1}\{E_c(u)\} \tag{23}$$

A growing Bezier curve

Curve approximation method, proposed in section 3.1 & 3.2, does not guarantee an accurate representation of any given curve. Sometimes, it may not be possible to compute an accurate (quadratic or cubic) curve for complex and higher order curves. Highly recommended solution is to implement it through piecewise approximation. In other words, to subdivide the given curve into sections and to approximate each section separately. One needs to handle many issues in this implementation like finding position of subdivision points, maintaining required continuity between each section and computational efficiency. Proposed growing Bezier technique handles all such issues automatically.

A growing Bezier method using cubic Bezier approximation is explained in this section. Same method can be implemented for quadratic Bezier as well. In this method, initially a small section (may be 20 points) of given curve is approximated with cubic Bezier and the maximum error (*ME*) between two curves is checked. If the error is below threshold then few more points of curve are added to that section and so on. The process will continue till error goes beyond threshold. When it reaches to the specified threshold level, one curve section at this point will complete and its data points are added to the list of data points. Next section of curve will start expanding from this point onward. This process will continue till the given curve is exhausted completely. Piecewise cubic Bezier curve through recorded list of data points would be an approximating curve.

Error threshold means the maximum distance (*ME*) allowed at any point along the two curves (*i.e.* original and

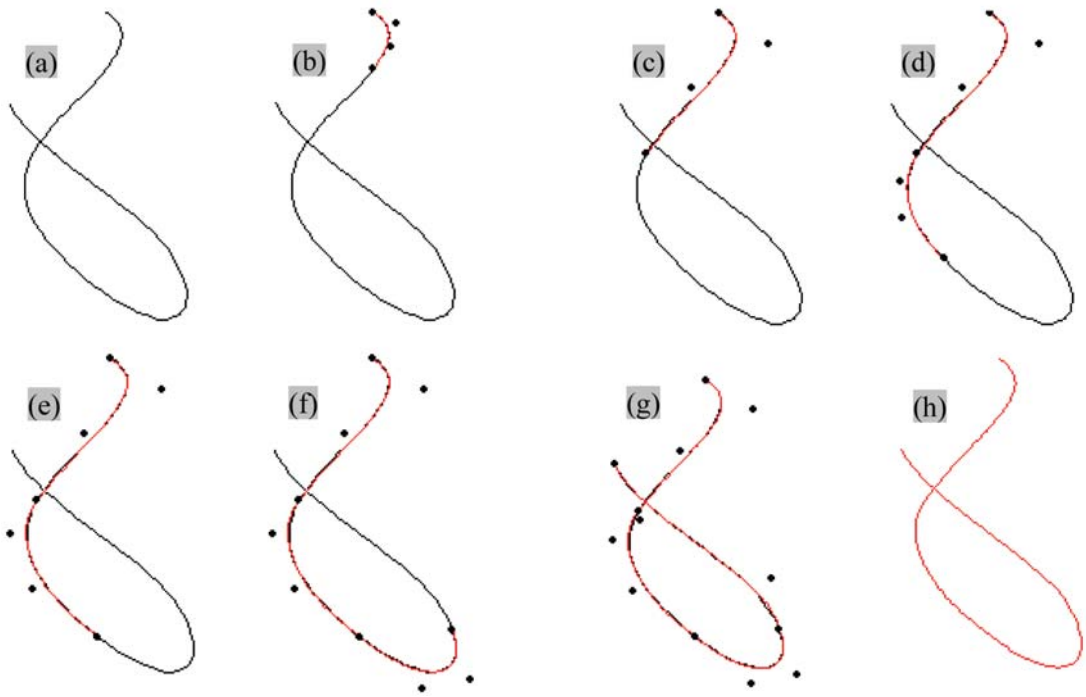

Fig. 6. Different stages of growing Bezier (red) over original curve (black)

computed). Specification of error depends upon the size, resolution, and smoothness/noise (eg. jaggies) of given curve. Different steps of growing Bezier are shown in Fig. 6. Error threshold for this curve was 0.1. Fig 6(a) is the original curve and its approximation consists of four cubic Bezier sections. Fig. 6(b) is the snapshot before reaching to the end of first curve section and Fig. 6(c) shows end of first curve section. The Bezier curve continues to grow in next section. Fig 6(d) is the snapshot between section 1 and 2, and Fig. 6(e) is the end of second section. Similarly Fig. 6(f) and 6(g) shows end of third and fourth section. The computed curve is shown in Fig. 6(h). It is accurate, smooth and data points are well expanded.

In proposed algorithm, speed of growth of approximating curve is proportional to the difference of *ME* from specified threshold level. Growth increment is higher for large differences and vice versa. After each iteration, expected length (in terms of curve points) of approximating quadratic or cubic Bezier is computed by ratio and proportion. The growth increment for next iteration is taken at 70% of the expected length. Computation of expected length becomes more accurate with the growth of curve. This method enables the fasted growth in start and it slows down near destination. Proposed methodology of controlling the growth (of approximating Bezier curve) makes the algorithm highly efficient.

Results analysis

A curve approximation produces curves of given shape which are size effective and can be very useful in many applications like vector graphics, computer drawing/painting, font designing and animations. Efficiently finding a set of data points (control points) without loosing quality of approximation is the main task of research in this area. The proposed approximation technique, by growing Bezier, is very simple and efficient method of approximating the higher degree polynomial and complex curves. It has been compared with different algorithms on the basis of below parameters.

- *Number of Segments.* The boundary is segmented if approximated curve goes beyond specified error threshold. Increase in segmentation means less compression and more data points. Least number of segments is desired.
- *Number of Control Points.* This parameter is important to compare the segmentation of approximating quadratic and cubic curves. Each quadratic and cubic segment consists of 3 and 4 control points respectively.
- *Integral Square Error (ISE).* This error is used to assess the total distortion/error caused by the approximating curve. It is defined as:

$$ISE = \sum_{i=1}^{n} e_i^2 \tag{24}$$

Where e_i is the Euclidian distance of ith point.

- *Maximum Error (ME).* It is the maximum distance of computed curve from original, measured as Euclidian distance. Integral square error gives the total distortion and does not reflect the sudden increase in error at small portion of curve. Such deviations are covered in this parameter. It is given as:

$$ME = \max_{i=1}^{n}\{e_i\} \tag{25}$$

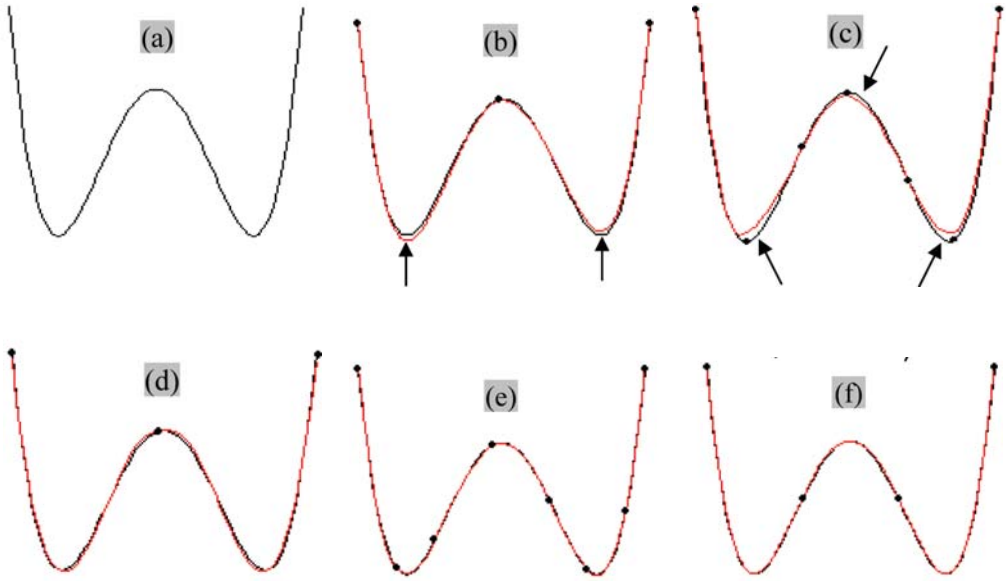

Fig. 7. Comparison of results. (a) Original curve, (b) Approximation with [27], (c) Approximation with [28], (d) Approximation with [29], (e) Approximation with proposed algorithm (growing quadratic Bezier), and (f) Approximation with proposed algorithm (growing cubic Bezier)

- *Computation Time.* Amount of time taken by the machine highly depends upon the implementation technique and speed of processor. However, some general conclusions can still be drawn. The algorithms were run for at least ten times on 1.7 GHz computer and the average time was taken.

Fig. 7 shows the approximation results with different algorithms [27,28,29]. Following curve was taken for testing the results.

$$y = \frac{x^4}{4} - 2x^2 + 4 \qquad (26)$$

at $-3 \le x \le 3$

Plot of eq. 26 is shown in Fig. 7(a). Quantitative comparison of results, with different algorithms, is shown in Table 1. Approximating curve in Fig. 7(b) is computed using the algorithm of Sarfraz et al. [27]. This algorithm [27] computes the approximating piecewise cubic Bezier curve. It is based on search of intermediate control points (P_1 & P_2) along the tangents at the start and end of curve. If error between two curves is beyond specified threshold, it is divided into two from the maximum error point. The curve (Fig. 7(b)) computed by this algorithm [27] consists of two segments *i.e.* maximum compression when compared to other results. On the other hand, approximation error of [27] (*ISE* & *ME*) is higher than all algorithms except [28]. Some deviations (marked with arrows) from original curve may be observed is Fig. 7(b). Computation time is another disadvantage of this algorithm [27], which is higher than all algorithms except [29].

Fig. 7(c) shows the curve approximation result using quadratic B-splines [28]. It uses opening angle plot to determine the position of unknown control points/knots. This algorithm [28] computes the curve in 6 segments (13

Table 1. Comparison of curve approximation results shown in Fig. 7

Algorithm	No. of Segments	No. of Control Points	Integral Square Error	Maximum Error	Computation Time (sec)
Sarfraz et al. [27]	2	7	0.756	0.318	1.237
Masood et al. [28]	6	13	2.391	0.384	0.975
Plass et al. [29]	2	7	0.528	0.257	2.548
Proposed (Quadratic)	7	15	0.393	0.186	0.043
Proposed (Cubic)	3	10	0.406	0.197	0.084

control points). Its approximation error is very high, which can be seen (marked with arrow) in Fig. 7(c). Although its [28] computation time is better than [27] & [29], but does not look very attractive when compared to proposed algorithm. Fig. 7(d) shows the approximation with least square fit using cubic splines. This algorithm [29] produces optimal curve which is the main cause of highest computation time. The optimal curve is expected to produce approximating curve with minimum error (*ISE*). It can be observed that its error is higher than proposed algorithm because the curve (Fig. 7(d)) consists of two segments only. Main drawback of least square fitting is high computation time.

Fig. 7(e) and Fig. 7(f) show the curve approximation with proposed algorithm using growing quadratic and cubic curves respectively. The growing quadratic curve takes least computation time, which looks very attractive when compared to other algorithms. Approximation error with growing quadratic is also lowest. The compression ratio of this technique looks little high. Main reason for high compression ratio is use of quadratic curve which is not very flexible. On the other hand, growing cubic Bezier using same approach produces the approximation in 3 segments. The computation time of cubic Bezier is also better than other algorithms.

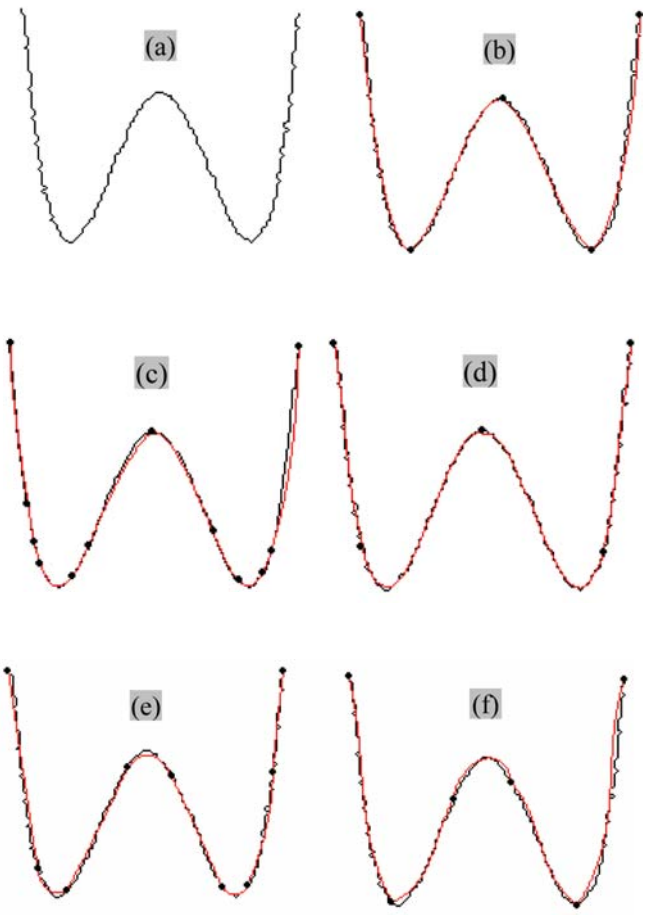

Fig. 8. Comparison of results. (a) Original curve (with noise), (b) Approximation with [27], (c) Approximation with [28], (d) Approximation with [29], (e) Approximation with proposed algorithm (growing quadratic Bezier), and (f) Approximation with proposed algorithm (growing cubic Bezier)

Another set of results are shown in Fig. 8 and Table 2. Fig. 8(a) shows the curve taken for approximation. This curve was obtained by introducing some noise on the curve of Fig 7(a). The noise was obtained by decreasing the resolution and passing it through a scanner (300dpi) once. Addition of noise produces irregularities (jaggies) along the curve. Now each point along the curve is represented in terms of pixels. Thus, error (*ISE* & *ME*) shown in Table 2 is also in pixels. An error threshold at 5 pixels was used for computing the curve by different algorithms. In other words, *ME* value in Table 2 cannot go beyond 5. Fig. 8(b) shows the approximation with algorithm [27]. It computes the curve with 4 segments and its approximation error is less than proposed algorithm using cubic Bezier. High computation time is the main disadvantage of this algorithm [27].

Fig. 8(c) shows the approximation using algorithm [28], which employs quadratic B-splines. Approximating curve consist of 11 segments whereas proposed algorithm computes the quadratic curve in 8 segments and with lesser integral square error. Also the computation time of algorithm [28] is much higher than proposed algorithm. Fig. 8(d) shows optimal approximating curve with 4 segments using least square fit. All comparative results of this algorithm [29] look better than others except computation time, which is the main drawback of least square fit. This drawback was observed in Table 1 as well. Fig. 8(e) & 8(f) shows the approximation results with proposed algorithm. Although the computed curves with proposed algorithm lacks in accuracy but low computation time is its biggest advantage.

Some curve approximation results by proposed algorithm are shown Fig. 9, 10, & 11. A cubic B-spline generated with 10 control points is shown in Fig. 9(a). Approximation with growing quadratic and cubic Bezier is shown in Fig 9(b) & 9(c) respectively. Growing Bezier algorithm reconstruct the curve with 11 quadratic and 7 cubic Bezier sections. It can be seen that the number of segments are relatively lesser for low curvature areas of curve given curve. Thus, the algorithm automatically takes care of curvature characteristic of without studying it explicitly.

Fig. 10 shows approximation of cardioid curve $r = 1 - \cos\theta$ with $0 \le \theta \le 2\pi$. Its Cartesian coordinates can be given as; $x = r\cos\theta$ & $y = r\sin\theta$. The curve is shown in Fig. 10(a). Approximation with growing quadratic Bezier is shown in Fig. 10 (b). It consists of 8 curve sections and 16 data points in total. Similarly, approximation with growing cubic Bezier (Fig. 10(c)) consists of 5 curve sections and 15 data points in total. Left half of cardioid curve is relatively smoother and results in lesser number of data points. Fig. 11 shows the approximation of sin curve, which is given as:

Table 2. Comparison of curve approximation results shown in Fig. 8

Algorithm	No. of Segments	No. of Control Points	Integral Square Error	Maximum Error	Computation Time (sec)
Sarfraz, et al. [27]	2	7	0.756	0.318	1.237
Masood, et al. [28]	6	13	2.391	0.384	0.975
Plass, et al. [29]	2	7	0.528	0.257	2.548
Proposed (Quadratic)	7	15	0.393	0.186	0.043
Proposed (Cubic)	3	10	0.406	0.197	0.084

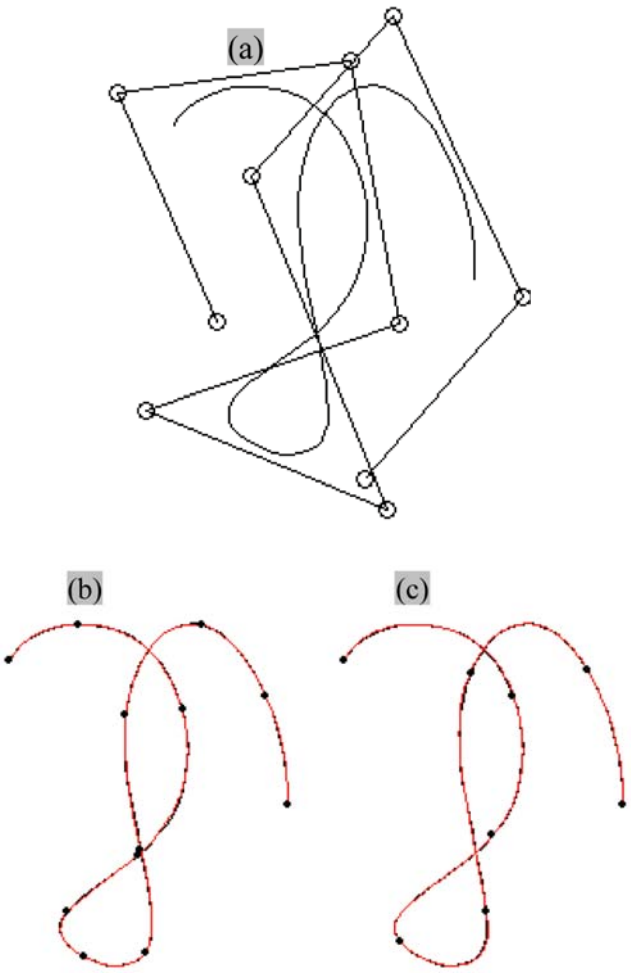

Fig. 9. Approximation with growing Bezier. (a) Given cubic B-spline generated with 10 control points, (b) Approximation with growing quadratic Bezier, and (c) Approximation with growing cubic Bezier

$$y = |\sin(x)| \quad \text{at} \quad -1.8 \le x \le 7.8 \tag{27}$$

Plot of curve using eq. 27 is shown in Fig. 11(a). Absolute value of sin function creates corners in the curve. Proposed algorithm tends to smooth the abrupt change in curvature (like corners). The algorithm needs minor changes to handles such curves. In proposed algorithm, growth of Bezier curve stops when the maximum error reaches the specified error threshold and next piece of approximating curve starts growing from there. Another condition (for breaking the segment) is imposed to handle the corners *i.e.* growth of Bezier curve will stop if sudden increase in error is detected. Fig. 11 shows the result after this modification. This methodology detects the corner points successfully and improves quality of approximation. Fig. 11(b) and 11(c) shows approximation with growing quadratic and cubic curve.

Some important features of proposed algorithm which distinguish it from other curve approximation techniques are summarized below.

- *Computational efficiency:* Most of the proposed curve approximation techniques waste lot of time in curvature analysis, calculating the approximation error and bringing curve closer to original iteratively [12, 14, 17, 22, 23]. The proposed method can calculate the exact value of error directly from the spread. This property is very useful for efficient calculation of unknown control points and subdivision points.
- *Incremental growth:* The growth of approximating Bezier curve is not constant rather it depends upon the difference between current value of error and specified error threshold. In other words, it is very fast in the start and slows down as the growing curve reaches close to it destination. This feature enables quick determination of next subdivision point.
- *Curve subdivision:* Curve subdivision is needed if the algorithm is unable to produce the approximation within specified error limitations. Most curve approximation algorithms [11,12,15,16] select the maximum error point for subdivision of curve into two. Calculation of maximum error points is a time consuming process and subdivision from this point is not optimal. The growing Bezier, in proposed algorithm, is designed to look for the maximum length of curve that can be approximated within specified error limits. However, the algorithm may leave the last segment very small. In such situations, decrease in the specified threshold will

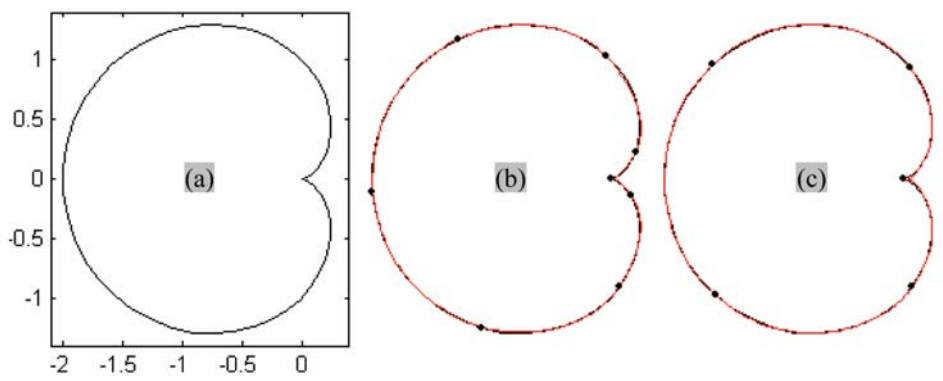

Fig. 10. Approximation with growing Bezier. (a) Given cardioid curve. (b) Approximation with growing quadratic Bezier. (c) Approximation with growing cubic Bezier

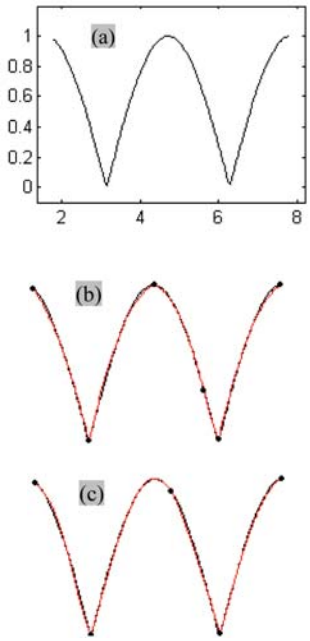

Fig. 11. Approximation with growing Bezier. (a) Given sin curve. (b) Approximation with growing quadratic Bezier. (c) Approximation with growing cubic Bezier

reduce the length of all sections hence increasing the length of last section. Thus, overall error between the two curves will reduce without increasing the number of segments.

- *Approximation error:* Determination of approximation error is very important to validate the quality of approximation. Calculation of error (e.g. Hausdorff distance) is time consuming process. Proposed algorithm gives an efficient method of error calculation (eq. 11 & 22) which can give exact value of error at any point along the curve.
- *Approximation of Corner Points:* Generally, curve approximation algorithms smoothes down the sharp turns and corners along given curve. The growing Bezier method can handle the corners as well. It detects the sudden change in error and marks this point as subdivision point. The methodology results in better approximation of curves having corners.
- *Curvature analysis:* Most curve approximation techniques are based on curvature analysis. Crampin's et al. [5] curve approximation technique was based on the notion that few points should be placed where radius of curvature is large and many where it is small. Some other good techniques were presented in [10,15,20] using similar ideas. The proposed algorithm does not specifically study the curvature but the detection of data points generally follows the same rule.
- *Approximation overhead:* Some curve approximation techniques use some additional parameters like tension [8,18,19,25], bias [1,13], local information [1,21] and some other control parameters [24] to increase flexibility. These parameters add complexity while calculating the approximation curve and needs for reconstruction of approximated curve. The proposed technique is based on standard Bezier curve with no additional overhead/parameters.
- *Quality of approximation:* In addition to above advantages, the proposed technique makes no compromise to the quality of approximation. The approximated curve is smooth, and accurate and the results are compatible to any other technique(s).

Conclusion

Curve approximation is a very useful technique in CAD, CAGD, painting and drawing packages, vector graphics, capturing 2D objects and animation. The proposed curve approximation technique uses Bezier curves which is simple and efficient and used in most graphic packages. In this technique, a quadratic or cubic Bezier curve starts growing along the given curve till it reaches to the end and determine the set of data points in its way. Most of the computation time of tradition curve approximation techniques was wasted in computing/analyzing curvature and approximation error. The proposed technique has no such limitations. In addition, incremental growth of approximating Bezier curve ensures high efficiency of

computation. Proposed technique can lead to various applications like representing shape outlines, font designing, handwriting recognition, signature recognition and OCR of various languages.

Acknowledgement

The author acknowledges the Higher Education Commission (HEC) of Pakistan for providing funds and University of Engineering and Technology (UET), Lahore for facilitating this research work.

REFERENCES

1. B. A. Barsky and J. C. Beatty, "Local Control of Bias and Tension in Beta-splines", *ACM Transactions on Graphics*, Vol. 2, No. 2, 1983, pp. 109-134.
2. R. H. Bartels, J. Beatty and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, San franscisco, CA: Morgan Kaufmann, 1987.
3. R. H. Bartels, J. C. Beatty and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. San Francisco, CA: Morgan Kaufmann, 1998.
4. P. E. Bézier, "*Emploi des Machines á Commande Numérique*', Paris: Mason et Cie, 1970. Tran: A. R. Forest, and A. F. Pankhurst, *P. Bézier, Numerical Control– Mathematics and Applications*, London: Wiley, 1972.
5. M. Crampin, R. G. Guifo and G.A. Read, "Linear approximation of curves with bounded curvature and a data reduction algorithm", *Computer Aided Design*, Vol. 17, No.6, 1985, pp. 257-261.
6. G. Farin, "*Curves and Surfaces for CAGD, A Practical Guide",* 5th ed. Academic Press, 2002.
7. F. Fritsch and R. Carlson, "Monotone piecewise cubic interpolation", *SIAM J. Numerical Analysis*, Vol. 17**,** No. 2, 1980, pp. 238–246.
8. J. A. Gregory, M. Sarfraz, "A rational cubic spline with tension", *Computer Aided Design*, Vol. 7, 1990, pp. 1-13.
9. Hearn and Baker, *Computer Graphics*, New Jersey: Prentice Hall, 1997.
10. G. E. Holzle. "Knot placement for piecewise polynomial approximation of curves", *Computer Aided Design*, Vol. 15, No. 5, 1983, pp. 295-296
11. W.C. Hu, H.T. Sheu, "Quadratic B-spline for Curve Fitting", *Proc. Natl. Sci. Counc, ROC(A),* Vol. 24, No. 5, 2000, pp. 373-381.
12. K. Itoh, Y. Ohno, "A curve fitting algorithm for character fonts", *Electronic Publishing,* Vol, 6, No. 3, 1993, pp. 195-198.
13. D. H. U. Kochanek, R. H. Bartels, "Interpolating splines with local tension, continuity, and bias control", *Proc. of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, 1984, pp. 33-41.
14. A. Masood, M. Sarfraz, "Capturing Outlines of Arabic Characters by Cubic Bezier Approximation", *Proc. of 1st International Conference on Geometric Modeling, Visualization & Graphics in conjunction with 8th Joint Conference on Information Sciences,* Salt Lake City, Utah, USA, 2005.
15. A. Masood, M. Sarfraz, S. A. Haq, "Curve Approximation with Quadratic B-splines", *Proc. of 9th IEEE International Conference on Information Visualisation,* London, UK, IEEE Computer Society Press, USA, 2005.
16. A. Masood and S. A. Haq, "Object Coding for Real Time Image Processing Applications", *Lecture notes in Computer Science, Vol. 3687: Pattern Recognition and Image Analysis,* Editors: Sameer Singh, Maneesha Singh, Chid Apte and Petra Perner: Springer-Verlag, 2005, pp. 550 – 559.
17. M. Plass, M. Stone, "Curve-Fitting with Piecewise Parametric Cubics", *Computer Graphics*, Vol. 17, No. 3, 1983, pp. 229-239.
18. S. Pruess, "Alternatives to the exponential spline in tension", *Math. Comp.*, Vol. 33**,** 1979, pp. 1273–1281.
19. G. M. Nielson, "Some piecewise polynomial alternatives to splines under tension", eds., *Computer Aided Geometric Design*, New York: Academic Press, 1974, pp. 209–235.
20. A. Razdan, "Knot placement for B-spline curve approximation", *Technical Report, Arizona State University*,1999,http://prism.asu.edu/publications.html .
21. F. A. Sohel, G. C. Karmakar, L. S. Dooley, J. Arkinstall, "Enhanced Bezier curve models incorporating local information", *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing,* 2005.
22. M. Sarfraz, M. A. Khan, "Automatic Outline Capture of Arabic Fonts," *Information Sciences*, Elsevier Science Inc., 2002, pp. 269-281.
23. M. Sarfraz, M. R. Asim and A. Masood, "Capturing Outlines Using Cubic Bezier Curves", *Proc. of 1st IEEE International Conference on Information and Communication Technologies: from Theory to Application*, 2004, pp. 539-540.
24. M. Sarfraz, M. F.A.Razzak, "An algorithm for automatic capturing of the font outlines", *Technical section of Computers & Graphics,* Vol. 26, 2002, pp. 795–804.
25. D. G. Schweikert, "An interpolation curve using a spline in tension", *J. Math. Phys.* 45, (1996), pp. 312-317.
26. F. A. Sohel, G. C. Karmakar, L. S. Dooley, "A Generic Shape Descriptor Using Bezier Curves", *Proc. of the International Conference on Information Technology: Coding and Computing,* Vol. 2*,* 2005, pp. 95-100.
27. M. Sarfraz, A. Masood, "Capturing outlines of planar images using Bezier cubics", *Computers and Graphics*, In Press, 2007.
28. A. Masood, M. Sarfraz, S. A. Haq, "Curve Approximation with Quadratic B-splines", *Proc. of 9th IEEE International Conference on Information Visualisation,* London, UK, 2005, pp. 419-424.
29. M. Plass, M. Stone, "Curve-Fitting with Piecewise Parametric Cubics", *Computers and Graphics*, Vol. 17, No. 3, 1983, pp. 229-239.