Bezier Curve-Based Character Descriptor Considering Shape Information

Ferdous Ahmed Sohel, Gour C. Karmakar*,* and *Laurence S. Dooley*

*Gippsland School of Information Technology
Monash University, Churchill, Victoria – 3842, Australia.
E-mail: {Ferdous.Sohel, Gour.Karmakar}@infotech.monash.edu.au

Abstract

While a Bezier curve (BC)-based character descriptor calculates the control points (CPs), it does not consider the shape characteristics for a prescribed distortion. This paper presents a novel *Bezier Curve-based character descriptor considering shape information (BCDSI)* which automatically divides a shape into segments based on the curvature and cornerity properties. Each segment is then defined by either a BC or straight line, depending on the distortion criteria in terms of a set of CPs. Quantitative and qualitative performances of the algorithm have been rigorously analysed upon both English and Bangla alphabetic characters, with the results confirming that the new BCDSI algorithm exhibits superior performance over other currently available shape descriptor algorithms employing the BC.

Keywords: Bezier curves, shape descriptor, curvature, corner points, control points.

1. Introduction

Efficient encoding of the character outline is one of the prime challenges in mobile multimedia communications due to the inherent bandwidth limitations. Besides low bit-rate communications, efficient coding can benefit a raft of applications as diverse as image indexing and retrieval, character recognition, creation of fonts, and the conservation and protection of historic and antique calligraphic characters.

In [1] and [2] Bezier curves (BC) have been applied to describe Chinese calligraphic characters and the outline of Arabic characters respectively. In these cases, shapes are described by composite Bezier curves [3] so they are divided into multiple segments, with each segment then described by an individual cubic BC. A particular character is segmented depending on the curvature information of the shape. The start and end points of each segment are considered to be the first and last CPs of the corresponding curve and since the BC interpolates the terminal CPs [3], a series of BCs describes a shape of multiple segments. The other CPs, (the second and third), are generated using different methods [1]-[2]. In [1] the second and the third CPs are determined using computationally expensive trial and error methods, which like all such methods mean that convergence of the CP calculation process is highly dependent upon the initial positions of the CPs. Moreover, since it considers the second and third CPs to be on the tangents on the shape at the first and fourth CPs respectively, the quality of this descriptor is dependent on the shape direction at the segment terminals. Thus to ensure a maximum admissible distortion a large number of segments may be required. In [2], the second and third CPs for each segment are calculated by optimising the distortion produced by the descriptor. The main drawback of this technique is that it defines all segments by cubic BCs, thus requiring four CPs in order to describe each segment. However, characters of many alphabets, including Bangla and English, can be easily described by straight lines, which involve only two CPs. Therefore, depending on a segment's curvature property, either a straight line or BC can be employed to describe any segment thereby save on the number of CPs. This is ultimately reflected in a lower bit requirement, which is one of the prime motivations for any mobile multimedia communication application, where storage, transmission and processing cost is highly bit dependent. This paper addresses these issues and presents an efficient algorithm called *Bezier curve-based character descriptor considering shape information (BCDSI) w*hich automatically divides a shape into segments and depending on the curvature of each segment, deploys either a BC or straight line to reduce the number of CPs. The quantitative and qualitative performances of BCDSI have been rigorously tested upon both English and Bangla character sets. It is evident from the results analysis that the proposed algorithm exhibits better performance compared with other existing algorithms.

The remainder of the paper is organised as follows: Section 2 provides a short overview of classical BC theory with the rationale of choosing BC in this algorithm, while Section 3 presents the proposed BCDSI algorithm for describing characters using BC.

6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)
0-7695-2841-4/07 $25.00 © 2007 IEEE

Section 4 furnishes a comprehensive analysis of the results, with some conclusions drawn in Section 5.

2. Brief Overview of Bezier Curves

The functional Bernstein form of the Bezier curve is given by [3]:-

$$Q(t) = \sum_{k=0}^{N} p_k B_k^N(t); \quad 0 \le t \le 1 \qquad (1)$$

where $B_k^N(t) = \binom{N}{k}(1-t)^{N-k} t^k$ is the Bernstein function, t is the Bezier parameter, with the steps in t determining the number of points on the BC and $P = \{p_0, p_1, \cdots, p_N\}$ is an ordered set of control points of an N degree curve.

Being in a family of parametric curves, the beauty of the BC is that even with a few CPs at the encoder, at the decoding end, an arbitrary number of curve points can be generated depending on the steps in t, with the greater the number of CPs on a curve, the smoother the curve. This unique family of parametric curves includes both Hermite curves and splines, with BC chosen as a shape descriptor because it is computationally efficient and straightforward to compute. In this paper, Cubic BCs are used to define each shape segment, as a lower order curve such as a quadratic BC is less smooth. While higher degree curves are preferable for shape approximation, they also require a larger number of CPs to be computed and the generation is as a consequence computationally higher. Moreover, due to global effect of all the CPs, a higher degree BC may introduce unnecessary oscillations due to the orientation of the CPs. The cubic BC is defined as:-

$$Q(t) = (1-t)^3 p_0 + 3t(1-t)^2 p_1 + 3t^2(1-t)p_2 + t^3 p_3 \qquad (2)$$

3. The proposed BCDSI algorithm

The new BCDSI algorithm comprises three major components: segment extraction, control point calculation for each segment and CP encoding. Each of these is examined in the following three subsections.

3.1. Segment Extraction

It is vital to automatically find the starting point of a given shape from which segment extraction commences. As this point will clearly be the first point of the first segment, in this paper, the shape point with the highest curvature is chosen, since this is most likely to reduce the number of segments. After this process, the ranked set of vertices $V = \{v_0, v_1, \cdots, v_{M-1}\}$ represents the shape, where M is the number of points on the shape.

In segmenting the curves, the curvature and cornerity of the point on the shape are considered. For this purpose, Beus-Tiu corner detection method [4] is used, since as concluded in the discussions and analysis in [5], it is the nearest to that perceived by a human viewer. The Beus-Tiu method divides the shape such that the last shape point of the i^{th} segment is the first shape point of the $(i+1)^{th}$. However, in case of the last segment for a closed shape, the last point is the first point of the first segment. Moreover, one character may have multiple closed components. In this case, each component is regarded as a separate enclosed shape. Therefore, each component has a starting point and is encoded as if they were representing a separate shape. The example shown in Figure 1 has three closed components C_1, C_2 and C_3.

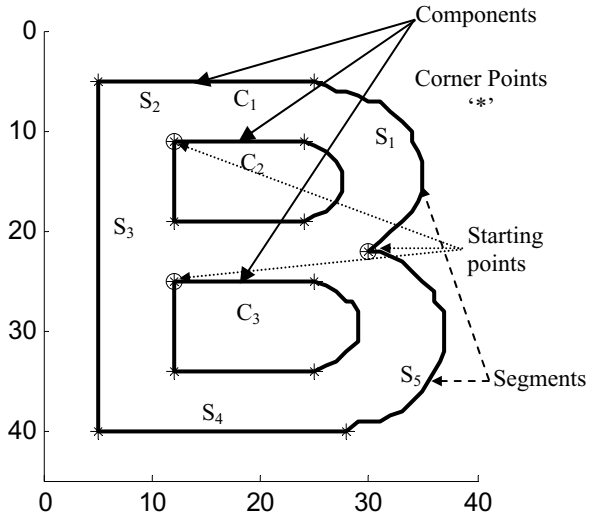


Figure 1: Example character shape divided into components and segments.

After detecting the corners, each component has a set number of segments, so in the Figure 1 example, component C_1 has five segments $S_1, S_2, \cdots, S_5$.

3.2. Control point calculation

Amongst the shape segments, some may be (almost) straight lines, while others will be curve-shaped. For example, in Figure 1, segments S_2, S_3 and S_4 are straight lines, while S_1 and S_5 are curve-shaped. For the curved shaped segments cubic BC can be efficiently used with four CPs, but for the segments which are quasi straight lines, BC become too inefficient since two end-points of the line are sufficient to represent it. The end-points of a line are also considered CPs hereafter. In this algorithm, a segment is firstly checked

as to whether it is in fact a straight line. If the distortion between the line connecting two segment end points and the segment is within a prescribed admissible distortion, the segment is defined by a straight line. Otherwise a BC is employed for that segment. If the distortion between the curve and the segment is higher than the permitted value, then that particular segment is further divided into two sub-segments at the maximum distortion point and each sub-segment is regarded as a separate segment thereon.

Calculation of CPs for a BC approximation for a particular segment:

Consider the i^{th} segment $S_i = \{s_{i,1}, s_{i,2}, \cdots, s_{i,|S_i|}\}$ where $|S_i|$ is the number of shape points in the segment S_i. Clearly all S_is form the shape V. The number of steps in the values of t in (2) indicates the number of points on the curve. To ensure the number of curve points is equal to the shape points in a segment the *chord-length parameterisation* (3) is used ascertain the values of t. As a result, each point on the shape has a corresponding point on the curve and will provide smoothness and fairness in the shape.

$$t_{i,k} = \begin{cases} 0 & if\ k = 1 \\ \dfrac{|s_{i,1}s_{i,2}| + |s_{i,2}s_{i,3}| + \cdots + |s_{i,k-1}s_{i,k}|}{|s_{i,1}s_{i,2}| + |s_{i,2}s_{i,3}| + \cdots + |s_{i,|S_i|-1}s_{i,|S_i|}|} & if\ k < |S_i| \\ 1 & if\ k = |S_i| \end{cases} \quad (3)$$

Trivially, the first and the last control points are respectively the first and the last point of a particular segment, i.e. $p_{i,0} = s_{i,1}$ and $p_{i,3} = s_{i,|S_i|}$. For the other two control points, the optimal method proposed in [6] has been used. The main philosophy behind this optimisation is to find the locations of the control points so that it minimises the square distortion between the shape segment and the generated curve. i.e.

$$E_i = \sum_{k=1}^{|S_i|} \left(Q_i(t_{i,k}) - s_{i,k}\right)^2 \quad (4)$$

To obtain optimal results, the partial derivatives E_i with respect to $p_{i,1}$ and $p_{i,2}$ are equated to zero and solved. The partial derivatives for $p_{i,1}$ is $\frac{\partial E_i}{\partial p_{i,1}} = 2\sum_{k=1}^{|S_i|} \left(Q_i(t_{i,k}) - s_{i,k}\right) \cdot \frac{\partial Q_i(t_{i,k})}{\partial p_{i,1}} = 0$ which implies that $\sum_{k=1}^{|S_i|} B_1^3(t_{i,k}) \cdot \left(Q_i(t_{i,k}) - s_{i,k}\right) = 0 \quad (5)$

Similarly, for $p_{i,2}$, $\sum_{k=1}^{|S_i|} B_2^3(t_{i,k}) \cdot \left(Q_i(t_{i,k}) - s_{i,k}\right) = 0 \quad (6)$

So solving (5) and (6) gives,

$$p_{i,1} = \frac{A_{i,2}F_{i,1} - A_{i,(1,2)}F_{i,2}}{A_{i,1}A_{i,2} - A_{i,(1,2)}^2} \quad \text{and}$$

$$p_{i,2} = \frac{A_{i,1}F_{i,2} - A_{i,(1,2)}F_{i,1}}{A_{i,1}A_{i,2} - A_{i,(1,2)}^2} \quad (7)$$

where, $A_{i,j} = \sum_{k=1}^{|S_i|} \left[B_j^3(t_{i,k})\right]^2$,

$A_{i,(1,2)} = \sum_{k=1}^{|S_i|} B_1^3(t_{i,k}) \cdot B_2^3(t_{i,k})$,

$F_{i,j} = \sum_{k=1}^{|S_i|} B_j^3(t_{i,k}) \left[s_{i,k} - B_0^3(t_{i,k}) p_{i,0} - B_3^3(t_{i,k}) p_{i,3}\right]$ and $j \in \{1,2\}$. Without loss of generality, the x and y-coordinate values of $p_{i,1}$ and $p_{i,2}$ respectively can be found accordingly using the corresponding coordinate values of $s_{i,k}$, $p_{i,0}$ and $p_{i,3}$.

3.3. Control points coding

The CPs are required to be encoded efficiently to ensure a precise descriptor. Since the shapes are considered closed, the final CP must not be duplicated while coding. Moreover, in this algorithm certain segments are described by BCs of four CPs while others are defined by straight lines comprising two CPs, so in order to indicate whether the next segment is a straight line or BC incurs a one bit overhead. These two issues are addressed in the encoder framework shown in

Figure 2, where for a straight line one CP is encoded for each segment and for a BC segment three CPs are encoded. However, for the last segment there is one fewer CP in both cases. Each rectangular box contains the necessary information to represent a particular CP depending on the encoding method deployed. Figure 3 shows the general framework for the decoder. The following paragraphs discuss a number of possible CP coding strategies.

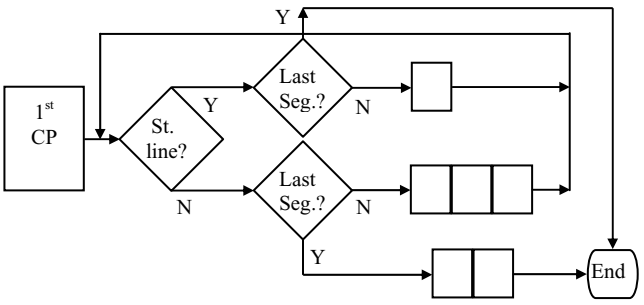


Figure 2: Control point encoding framework.

0-7695-2841-4/07 $25.00

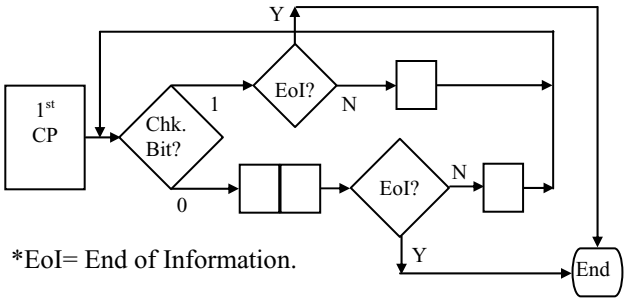


Figure 3: Control point decoding framework.

The simplest way of encoding the control points is to use the absolute coordinate values of successive CPs, however, in this case the bit requirement will be large. If the range of coordinate values is known, *fixed length coding* or *N-ring* generalised chain code [7] could be used to obtain coding efficiency. However, in many cases the CP can be located even outside of the image and for chain coding purposes, there may be additional quantisation errors. In addition, since these two techniques do not consider the correlation between successive points, they are inefficient. Coding efficiency can be improved if the correlation between successive CPs is exploited. Object adaptive vertex based shape coding method (OAVC) [8] explores the correlation between the vertices and eliminates redundancy to gain efficiency in polygonal approximation of a shape.

The straight-line/curve check bits can form either one separate bit-stream in parallel with the CP encoding bit-stream or merged into the control point encoding bit-stream with proper sequence and positions.

3.4. Shape information decoding

Depending on the encoding method used, the decoder knows the length and thus the delimiter of each parameter and so can correctly parse these parameters (see Figure 3) and thereby reconstruct the shape using the decoded information.

4. Experimental results and analysis

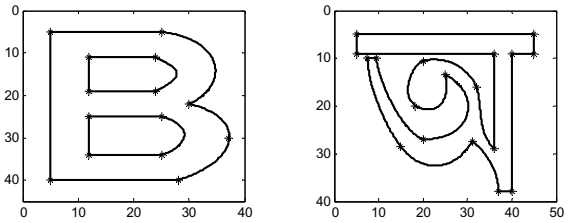
To quantitatively evaluate the performance of the new algorithm, the widely-used *shape distortion measurement* metrics [9] were employed. *Class one* distortion measures the maximum distortion D_{max} over the entire shape, while *Class two* distortion provides a measure of the mean-square (MS) distortion D_{ms} for a shape. The metrics are formally expressed as:-

$$D_{\max} = \max_{v_i \in V} d(l, v_i) \quad (8)$$

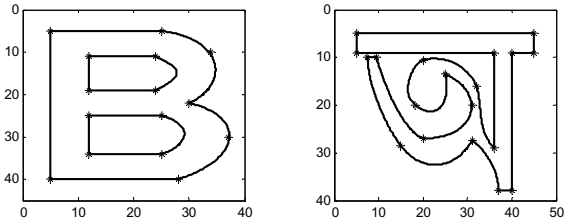
$$D_{MS} = \sum_{v_i \in V} d^2(l, v_i) \quad (9)$$

where $d(l, v_i)$ is the *actual shortest distance* [10] of the shape or polygon point v_i from the corresponding approximation l.

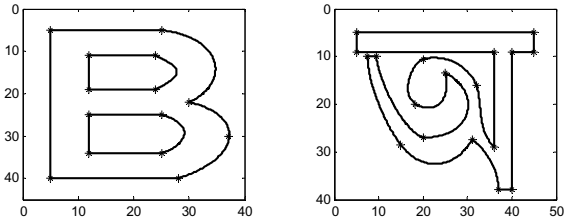
(a) Original character shape



(b) Decoded shape with BCDSI.



(d) Decoded shape with [1]



(d) Decoded shape with [2]

Figure 4: Comparison of the decoded shapes (*: segment ends) among the algorithms for $D_{\max} = 1$ pel.

Experiments were conducted upon the alphabetic characters from the English and Bangla language. One character from each language was analysed. From English script '**B**' and from Bangla '**Au**' are presented here. The main rational behind choosing them is that both characters have both straight lines and curve segments. Moreover, '**B**' has three separate

components. Figure 4 shows the comparative subjective results among the algorithms, while Table 1 provides the corresponding numerical results. Figure 4(a) shows the respective original character shapes used in the experiments. Figure 4(b), (c) and (d) respectively show the decoded shapes using the proposed BCDSI, [1] and [2] algorithms for $D_{\max} = 1$ pel. The corresponding quantitative results are summarised in Table 1.

For the '**B**' character BCDSI and [2] required 14 segments, while [1] required 15 segments. This is because [1] considers the control points on the tangents at the segment ends and thus produced higher distortion. For example, mean-squared distortion for BCDSI and [2] is 0.02 pel^2 while it is 0.15pel^2 in [1]. BCDSI outperformed the algorithms [1] and [2] when the comparisons are performed with respect to the bit requirement and the number of CP, which are the main performance metrics of a character descriptor. For this shape, the number of control point requirements for BCDSI, [1] and [2] are 21, 33 and 39 respectively which costs the respective bit requirements of 254, 385 and 450 bits using the object adaptive vertex-based shape coding method [8]. This is mainly because BCDSI considered that many segments could be encoded by a straight line rather than a BC. Similar observations were made for the '**Au**' character from Bangla script where the BCDSI required the lowest number of bits 345, to encode it, while for [1] and [2] these were 341 and 525 bits to maintain the class one distortion of $D_{\max} = 1$ pel.

Table 1: Numerical results for shape descriptors.

Character	Algorithm	MS distortion (pel^2)	Number of Segments	Number of Control Pts #CP	Bits
B	BCDSI	0.02	14	21	254
	[1]	0.15	15	33	385
	[2]	0.02	14	39	450
Au	BCDSI	0.03	18	36	345
	[1]	0.25	19	44	421
	[2]	0.03	18	54	525

5. Conclusions

This paper has presented a new *Bezier curve-based character descriptor considering shape information (BCDSI)* algorithm that automatically divides a shape into segments depending on the cornerity of the shape at the shape points. It utilises either a Bezier curve or straight line to approximate each segment based on a given admissible distortion and uses an efficient method to generate the Bezier control points to reduce distortion. The qualitative and quantitative performance analysis has proven the BCDSI algorithm is superior to existing Bezier curve-based character shape descriptors in terms of both the requisite bit-rate and number of control points.

References:

[1] H. -M. Yang, J. -J. Lu, and H. -J. Lee, "A Bezier curve-based approach to shape description for Chinese calligraphy characters," in Proc. of Sixth International Conference on Document Analysis and Recognition, pp.276-280, 2001.

[2] M. Sarfraz, and M. A. Khan, "Automatic outline capture of Arabic fonts," Information Sciences, Elsevier Science Inc., pp.269-281, 2002.

[3] F. S. Hill Jr., *Computer Graphics*, Prentice Hall, Englewood Cliffs, 1990.

[4] H. L. Beus, and S.S.H. Tiu, "An improved corner detection algorithm based on chain coded plane curves," Pattern Recognition, vol.20, no.3, pp.291-296, 1987.

[5] H.C. Liu, and M.D. Srinath, "Corner point detection from chain-code," Pattern Recognition, vol.3, no.12, pp.51-68, 1990.

[6] H.-H. Chang, and H. Yan, "Vectorization of hand-drawn image using piecewise cubic Bezier curves fittings," Pattern Recognition, vol.31, no.11, pp.1747-1755, 1998.

[7] H. Freeman, "Application of generalized chain coding scheme in MAP data processing," in IEEE Comp. Soc. Conf. Patt. Recog. Image Proc., pp.220-226, 1978.

[8] K. J. O'Connell, "Object-adaptive vertex-based shape coding method," IEEE Trans. Circuits Sys. Video Technol., vol.7, no.1, pp.251-255, 1997.

[9] G.M. Schuster, and A.K. Katsaggelos, *Rate-Distortion Based Video Compression-Optimal Video Frame Compression and Object Boundary Encoding*, Kluwer Academic Publishers, pp.222-223, 1997.

[10] F.A. Sohel, L.S. Dooley, and G.C. Karmakar, "Accurate distortion measurement for generic shape coding," Pattern Recognition Letters, vol.27, no.2, pp.133-142, 2006.

0-7695-2841-4/07 $25.00 © 2007 IEEE