Capturing Outlines of Arabic Characters by Cubic Bezier Approximation

M. Sarfraz[1] A. Masood[2]

[1]Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, KFUPM # 1510, Dhahran 31261, Saudi Arabia. (sarfraz@kfupm.edu.sa).

[2]Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan. (asif@uet.edu.pk).

Abstract

An efficient outline approximation technique is presented using cubic Bezier curves which is ultimately used for capturing Arabic characters. Approximation technique is based on computation of data points which is a mixture of interpolating and approximating data points. Approximating data points can represent blobby and circular shapes very efficiently and interpolating points are useful to preserve the original shape of outlines at sharp corners. A normal piecewise cubic Bezier spline through these data points is an approximating curve which is CG^1 continuous. Data point detection process is based on efficient control point search algorithm. Recursive segment subdivision is employed to keep the approximation error with in specified threshold limits. Demonstrated results show that only few data points can accurately represent the captured outlines. Any transformation operation on these data points results in overall transformation of captured outlines.

Keywords: Cubic Bezier curves; Corner points; Control point search; Subdivision points.

1. Introduction

Visual information of various objects and shapes may be stored in computer memory as image files. Complete image is normally taken as a whole for any further processing, which is an expensive operation and may result in deterioration of quality as well. Capturing techniques may be used to give some mathematical/computer representations to these shapes. This representation has many advantages, like data reduction, subsequent computational efficiency, easy to add transformation, shading, coloring and other effects.

Lot of work has been presented in the field of capturing techniques [5, 10-13]. In most of the previously presented techniques, data points were the interpolating points of given curve. The presented technique is based on combination of interpolating and approximating data points which results in high reduction of data points with increased quality of curve approximation. The determined data points are actually the control points of CG^1 continuous piecewise cubic Bezier curve. Various other techniques like control point search, corner detection, and recursive subdivision are used in this capturing system.

The paper is organized as follow. Curve approximation technique including control point search algorithm and recursive subdivision is presented in section 2. Curve approximation technique is generalized for capturing object outlines in section 3. Few capturing results are presented in section 4 and section 5 concludes this presentation.

2. Cubic Bezier Approximation

This section deals with approximation of given cubic Bezier curves, which is generalized for any given curve in later part of this section. Cubic Bezier curves are simple, efficient and easy to implement and provide enough flexibility. The cubic Bezier curves are generated with four control points $P_i = (x_i, y_i)$, with i varying from 0 to 3. These control points can be blended to produce the below position vector $p(u)$, which describes the path of an approximating Bezier polynomial function between P_0 and P_3 [4].

$$p(u) = (1-u)^3 P_0 + 3(1-u)^2 P_1 + 3u^2(1-u)P_2 + u^3 P_3 \qquad 0 \le u \le 1 \qquad (1)$$

Cubic Bezier curve with four control points is shown in figure 1. Control points P_0 and P_3 are the endpoints of cubic Bezier curves and P_1 and P_2 are the two approximating points. The tangents T_1 & T_2 at the beginning and end of cubic Bezier curve is along the line P_0P_1 and P_2P_3 respectively.

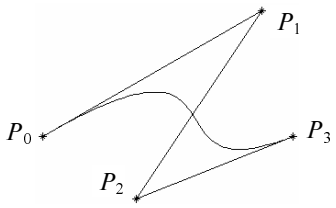

Fig. 1: Cubic Bezier curve.

2.1. Control Point Search

For any cubic Bezier curve, we need to find its four control points. Two control points (P_0 & P_3) lie at the curve endpoints and other two (P_1 & P_2) are to be searched, which lies along the tangents T_1 & T_2. To optimize the search algorithm, it is implemented in two phases. Phase 1 is very efficient but determines an approximate location of control points. Positions of these control points are refined in phase 2 which is relatively slow but accurate.

1. $P_1 = P_0$
2. $P_2 = P_3$
3. Calculate M_1 and M_2
4. Calculate AE
5. Do $P_1 = P_1 \pm M_1$ While(AE reduces)
6. Do $P_2 = P_2 \pm M_2$ While(AE reduces)
7. Repeat step 5 and 6 till P_1 or P_2 does not change

Fig. 2: Control point search algorithm.

Search for the control points P_1 & P_2 starts from the control points P_0 & P_3 respectively. Control points P_1 & P_2 are moved along their tangents T_1 & T_2 till approximation error (AE) is minimum. The algorithm is given in figure 2. The algorithm for phase 2 is also same except the value of AE, M_1 and M_2 are different. In phase 1, AE is the accumulated distance between two curves at five equally separated points along the curve. In phase 2, AE is the total area between two curves. M_1 & M_2 is one step movement for control points P_1 & P_2 in the direction of tangents T_1 & T_2. Value of M_1 & M_2 is 10 pixels for phase 1 and 1 pixel for phase 2.

The algorithm is separated in two phases to optimize the control point search efficiency. Phase 1 is computationally very efficient but looks for an approximate location of control points P_1 & P_2. Figure 3a shows approximating curve after phase 1. Phase 2 is an expensive operation but require slight adjust of control points. Figure 3b shows approximating curve after phase 2. It can be observed that control points get close to their destination in phase 1 and hit the target location very accurately after phase 2.

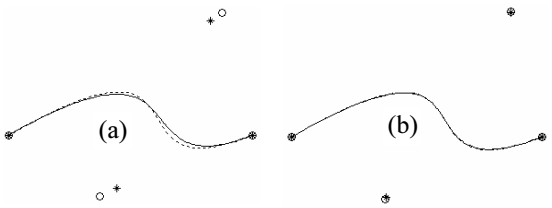

Fig. 3: Control point search. Circles and asterisks show the actual and current location of control points and dashed line shows the original curve (a) After phase1 (b) After phase2.

2.2. Recursive Subdivision

In recursive subdivision, maximum distance between two curves is evaluated. The curve is broken into two from that maximum error point if it is beyond specified error limits. Acceptable error limit depends upon the size/resolution of given curve. We take three pixels distance between two curves as default error limit. The point from where curve is broken into two is known as subdivision point. Segments are recursively subdivided into two till approximation error in each segment is below threshold error limit.

Control point search method (discussed above) is used to find the cubic Bezier control points for each sub-segment. In order to maintain CG^1 continuity between two neighboring segments, control point P_2 & P_3 of previous sub-segment and P_0 & P_1 of next segment must lie on same tangent line (shown with two sided arrow in figure 4). Results of segment subdivision are shown in figure 4a and 4b.

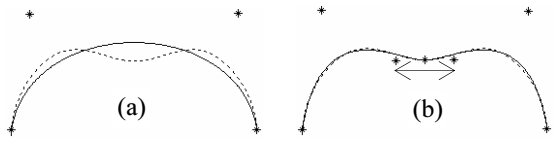

Fig. 4: Segment subdivision. (a) Before subdivision. (b) After subdivision.

3. Outline Capturing Process

Cubic Bezier approximation method was discussed in section 2. The same technique is generalized for capturing object outlines in this section. The capturing process can be mainly divided in three main phases. First, the outlines are extracted from given shapes. Second, corner points are detected and outline is divided into curve segments from these corner points. Third, each curve segment is processed for curve approximation. This phase is mainly focused here.

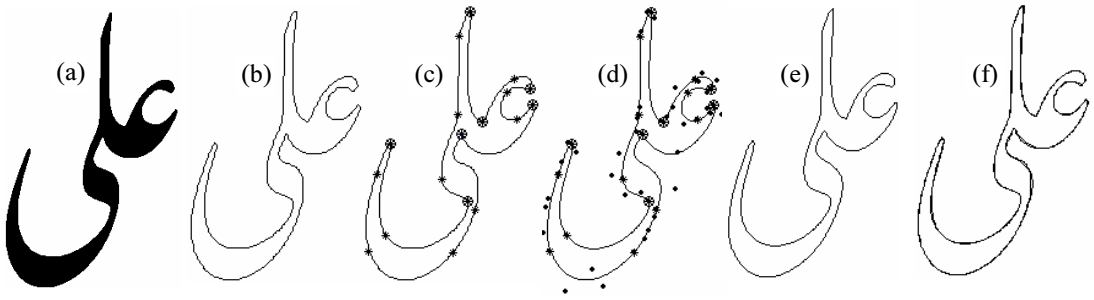

Fig. 5: Outline capturing for the shape of Arabic name 'Ali'. (a) Original bitmap image. (b) Extracted outline. (c) Extracted outline marked with segment endpoints (asterisks). Corners are marked with circles. (d) Computed outline marked with detected cubic Bezier control points. Intermediate control points P_1 & P_2 are marked with dots. (e) Computed outline. (f) Computed outline drawn over the original outline.

3.1. Outline Extraction

Object outlines can be extracted directly from the gray-scale images [6]. More reliable approach is to convert gray-scale image to binary and extract boundaries from that binary image. We convert gray-scale images to binary using [8]. Numbers of algorithms are available for boundary extraction from binary images [7,14]. We use a simple procedure to extract boundaries, which is any pixel with a pixel value 0 (black) is a boundary point if any of its four connected neighbors (upper, lower, left and right) has a pixel value 1 (white). Boundary points are then arranged in sequence by tracing each boundary loop. Boundary tracing algorithm can be found in [15] but this does not handle multiple and illegal loops. We use its modified version, given in [13]. Figure 5b show outline extraction result.

3.2. Corner Detection

Extracted outline is decomposed into pieces (curve segments) at the detected corners. Decomposition reduces boundary's complexity and thus simplifies the curve approximation process. Corners are the high curvature points and appear to be the natural break points. Precise detection of corners in an outline is very important to preserve the overall shape of object outlines. Readers are referred to some commonly used algorithms for detection of corner points [1,2,3,9]. In this paper, [2] is used for detection of corners. Figure 5c show corner detection result.

3.3. Outline Approximation

The extracted outline is decomposed into curve segments at the detected corner points. Curve approximation technique, discussed in section 2, is used for approximation of each segment. Recursive segment subdivision will ensure the overall quality of approximating curve. Set of control points/data points for each segment is the end product. Data points are detected such that the normal piecewise cubic Bezier curves over them is a captured outline. The approximation points (control points P_1 & P_2) are very useful to represent blobby and circular shapes and interpolation points (P_0 & P_3) preserve the shape of outline at sharp corners. Approximation error parameter can be used to control the quality of captured outlines and the number of data points. Outline capturing results are discussed next.

4. Results Demonstration

Quality of outline capturing algorithm can be measured by its computational efficiency, shape preservation, approximation error and data reduction. An efficient method is presented for searching cubic Bezier control points. A simple cubic Bezier curve over these control points is a computed outline. No extra control parameters are used to enhance flexibility of cubic curves. Thus, this arrangement makes this algorithm computationally very efficient. Combination of data interpolation and data approximation results in high reduction of data and also improves the over all shape. Computed outline is compared with original in demonstrated results Approximation error is the area between two curves. It can be controlled by a given threshold limit. Set of data points represents the complete shape of given object. Any transformation effect (like translation, scaling, rotation and shearing etc) on the data points will transform the captured outlines.

The proposed algorithm was tested on various 2D shapes and it produced quite elegant results. Results of only one shape are demonstrated (figure 5)

due to lack of space. Test results are demonstrated at default threshold value of 3. Figure 5 shows the results of Arabic name "Ali". The results are distributed in six sub-figures and their brief description is as follow. Figure 5a shows the original bitmap image taken for testing this algorithm. Figure 5b shows the extracted outline. Outline extraction method was given in section 2. Figure 5c is the segment endpoints (including corners and subdivision points) marked over the original outline. Points in circle are the detected corner points and points marked with asterisks are the subdivision points. Figure 5d shows all cubic Bezier control points marked over computed outlines. In addition to the endpoints, points represented with dots are the detected intermediate control points (P_1 & P_2). Figure 5e shows computed outlines. Comparison of computed outline with extracted object outline is made in figure 5f. Two outlines are drawn over each other to demonstrate its accuracy. One can observe that the algorithm performs equally well both for straight lines and circular arcs.

5. Conclusion

An algorithm for capturing outlines of 2D shapes has been presented in this paper. Combination of data interpolation and data approximation is used which not only results in high reduction of data points but also improves the quality of captured outlines. The proposed technique is suitable for approximation of any curve(s), shapes and object outlines. The captured outlines are CG^1 continuous and preserve the shape of outline at sharp corners as well. The presented method of control point search is computationally very efficient, simple and easy to implement and produces very elegant results. Recursive segment subdivision is used to keep the approximation error within specified error limits. Demonstrated results show that the computed outlines are very close to the original outlines of given 2D shapes.

6. Acknowledgments

The first author acknowledges the support of King Fahd University of Petroleum and Minerals for funding this work under the Project No. EE/AUTOTEXT/232. The second author acknowledges the Higher Education Commission of Pakistan and The University of Engineering and Technology for supporting this research work.

7. References

[1] H.L. Beus, S.S.H Tiu, "An Improved Corner Detection Algorithm based on Chain Coded Plane Curves", *Pattern Recognition*, vol. 20, pp. 291-296, 1987.

[2] D. Chetverikov, Z. Szabo, "A Simple and Efficient Algorithm for Detection of High Curvature Points in Planner Curves", *Proc. 23rd workshop of Australian Pattern Recognition Group, Steyr*, pp. 175-184, 1999.

[3] H. Freeman, L.S. Davis, "A Corner Finding Algorithm for Chain-Coded Curves", *IEEE Trans. Computers*, vol. 26, pp. 297-303, 1977.

[4] D. Hearn, M.P. Baker, *Computer Graphics*, Prentice Hall publication, 1997.

[5] F. Hussain, B. Zalik, S. Kolmanic, "Intelligent Digitization of Arabic Characters", *Proc. Inter. Conf. on Info. Visualization*, pp. 337-342, 2000.

[6] R. Katz, C. Delrieux, "Boundary Extraction through Gradient-Based Evolutionary Algorithm", *JCS&T*, vol. 3, no. 1, 2003.

[7] K.M. Lam, H. Yan, "Locating head boundary by snakes", *Proc. Int. Sym. on Speech, Image Processing and Neural Networks*, pp 17-20, 1994.

[8] N.A. Otsu, "Threshold Selection Method from Gray-Level Histograms", *IEEE Trans. on Sys., Man, and Cyber.*, vol. 9, no. 1, pp. 62-66, 1979.

[9] A. Rosenfeld, J.S. Weszka, "An Improved Method of Angle Detection on Digital Curves", *IEEE Trans. Comp.*, vol. 24, pp. 940-941, 1975.

[10] M. Sarfraz, M.F.A. Razzak, "An Algorithm for Automatic Capturing of Font Outlines", *Journal of Computers & Graphics, Elsevier Science*, vol. 26(5), pp. 795-804, 2002.

[11] M. Sarfraz, "Outline Representation of Fonts using Genetic Approach", *Advances in Soft Computing: Engineering Design and Manufacturing, Springer*, pp. 109-118, 2003.

[12] M. Sarfraz, "Some Algorithms for Curve Design and Automatic Outline Capturing of Images", *IJIG*, vol 4(2), pp. 301-324, 2004.

[13] M. Sarfraz, M.R. Asim, A. Masood, "Capturing Outlines using Cubic Bezier Curves", *Proc. of 1st Inter. Conf. on Info.and Comm. Tech.: from Theory to Application*, 2004.

[14] J. Serra, *Image Analysis and Mathematical Morphology*, Acad. Press, 1982.

[15] M. Sonka, V. Hlavac, R. Boyle, *Image Processing, Analysis, and Machine Vision*, Brooks/Cole publication, pp 142-143, 2001.