Approximation of digitized curves with cubic Bézier splines

Conference Paper *in* Proceedings / ICIP ... International Conference on Image Processing · September 2010

DOI: 10.1109/ICIP.2010.5651820 · Source: DBLP

APPROXIMATION OF DIGITIZED CURVES WITH CUBIC BÉZIER SPLINES

Alexander Kolesnikov

School of Computing, University of Eastern Finland, Joensuu, FINLAND
alexander.kolesnikov@uef.fi

ABSTRACT

In this paper we examine a problem of digitized curves approximation for raster graphics vectorization and develop an efficient implementation of a near-optimal Dynamic Programming algorithm for digitized curves approximation with cubic Bézier splines for a given distortion bound. For better fitting performance, we introduce the *inflection points* with relaxed constraint of tangent continuity. The proposed algorithm demonstrates superiority over the iterative breakpoint-insertion method in terms of segments number for a given distortion bound.

Index Terms— Spline functions; curve fitting; image shape analysis; graph theory.

1. INTRODUCTION

A contour of bitmap object in a raster image can be represented in a vector format. Approximation of digitized curves with spline functions is widely used for vectorization of the raster graphics, font generation, shape compression, and computer aided design [1−8]. Piecewise approximation includes a partition of input curve into segments with breakpoints and a subsequent approximation of the segments with spline functions under the proper condition for derivatives continuity at the breakpoints. In this paper, we focus on the problem of distortion-constraint piecewise approximation of digitized curves with cubic Bézier splines.

The following two issues related to the problem in question should be considered: 1) continuity of curve derivatives at the breakpoints; and 2) optimal number and location of the breakpoints.

For smooth curves, we preserve continuity of the first derivative of approximation spline functions at the breakpoints. However, at some points of a curve the first derivative is not necessary continuous. The known solution to the problem is to divide the curve into pieces at such *corner points* where the first derivative is discontinuous and to process each piece independently [2,3,5−8].

The problem of the optimal number and location of breakpoints can be solved with the Dynamic Programming (DP) algorithm with corresponding cost function [1,4]. In [1], the cost function was presented as a sum of approximation errors and a penalty for the number of breakpoints. In [4], the algorithm for contour compression with a minimal bitrate for a given threshold on distortion was introduced. The compression was obtained by the curve approximation with quadratic B-splines with the DP search for optimal location of the control points. The main drawback of the optimal algorithm is its high time complexity.

The most popular approach to the problem includes iterative insertion of one or two breakpoints into segments [2,3,5−8]. A new breakpoint can be inserted at the point where constraint on maximal distortion is violated, for example. The algorithm is fast and gives good results for curves with few breakpoints. However, this algorithm produces too much segments for large-size and noisy curves.

Our goal is to develop an efficient algorithm for approximation of digitized contours with cubic Bézier splines for a given threshold on maximum distortion. To overcome the problem with the first derivative discontinuity at corner points we offer to consider the *inflection points* of the input curve as candidates to breakpoints for piecewise approximation.

Then, we approach the problem of optimal approximation with the spline functions for a given distortion bound. We apply the DP algorithm to the problem at hand and discuss the ways to speed-up the DP search process.

In the Section 2 we introduce inflection point approach. In the Section 3 we formulate the problem as an optimization task, and in the Section 4 we suggest DP algorithm to the problem with full and reduced search in the feasibility graph.

Figure 1. Example of raster graphics vectorization: a) input raster image 72×42 pels; b) result of vectorization with cubic Bézier splines for δ_0=1 pixel (right).

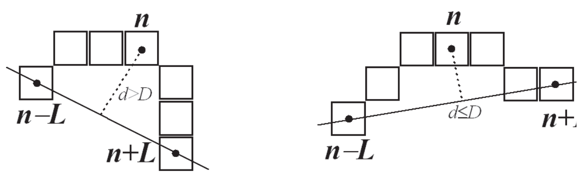

Figure 2. Scheme of inflection point detection: a) the vertex $\mathbf{p}(n)$ is an inflection point if $d>D$ (left); b) the vertex $\mathbf{p}(n)$ is a non-inflection point if $d \le D$ (right). Here L=3

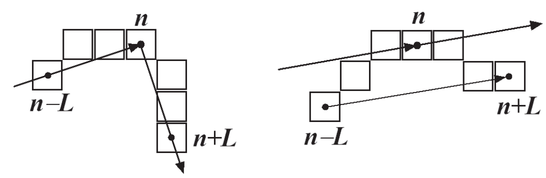

Figure 3. Scheme of tangent lines calculation at the point $\mathbf{p}(n)$: a) for the inflection point with non-continuous first derivative for the adjacent segments; b) for the non-inflection point $\mathbf{p}(n)$ with continuous first derivative for the adjacent segments. Here L=3.

2. INFLECTION POINTS

In fact, the contours of the objects can have the corners where the first derivative is not continuous. The known algorithms detect the corner points, divide the curve into pieces at these points [2,3,5–8], and process the pieces independently. The visual quality of such approximation depends on the proper selection of the corner points.

We approach to the problem in another way. First, we detect the points with high curvature (hereinafter referred to as the *inflection points*). Any efficient algorithm for corner points detection can be applied for this purpose, we use algorithm [7] (see Fig. 2). But instead of forcibly breaking the curve at these points we treat the latter as candidate breakpoints for the DP search where constraint on tangent continuity is released. This approach is less sensitive to efficiency of the corner detection algorithm. If more than necessary inflection points have been detected, only some of them will be actually used as the breakpoints. A point with high curvature is not necessary the breakpoint with non-continuous first derivative (see Fig. 4).

At the inflection point $\mathbf{p}(n)$, we evaluate the tangents for the adjacent segments independently, using the data only from the corresponding segments. For example, incoming and outcoming tangent lines at the point $\mathbf{p}(n)$ are defined by the line segments between points $\mathbf{p}(n-L)$ and $\mathbf{p}(n)$ and points $\mathbf{p}(n)$ and $\mathbf{p}(n+L)$, respectively (see Fig. 3). At the non-inflection breakpoint $\mathbf{p}(n)$ we evaluate the common tangent for two adjacent segments using the data from both segments around the point. Tangent lines of the curve at the point $\mathbf{p}(n)$ for adjacent segments are defined by the line segment between the points $\mathbf{p}(n-L)$ and $\mathbf{p}(n+L)$ (see Fig. 3). However, more advanced methods can be used for tangent estimation.

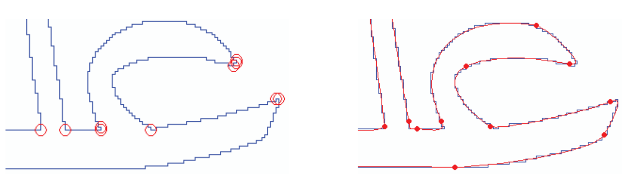

Figure 4. Test curve #2 (fragment): detected inflection points (red circles), D=1.5, L=3 (left), and approximating cubic Bézier splines (red line) with breakpoints (red dots) for δ_0=1 (right).

3. PROBLEM FORMULATION

An n-vertex planar curve $\mathbf{P}$ is defined as a sequence of points $\mathbf{P}=\{(\mathbf{p}(1)), \ldots, (\mathbf{p}(N)\}$, where $\mathbf{p}(n)=(x(n), y(n))$. The curve $\mathbf{P}$ is divided into M segments $S(i_m, i_{m+1})=[\mathbf{p}(i_m), \ldots, \mathbf{p}(i_{m+1})]$ by breakpoints $\mathbf{p}(i_m)$, where $i_1=1$, $i_{M+1}=N$, and $m=1, \ldots, M$. Each segment of the curve P is approximated by a cubic Bézier spline $\mathbf{F}(t)$.

The approximation error with L_∞-norm for the curve segment $S(i_m, i_{m+1})$ is defined as the maximal distance between the vertices of $\mathbf{P}$ and the approximating function $\mathbf{F}$:

$$e(i_m, i_{m+1}) = \max_{i_m \le n \le i_{m+1}} \left\{ \left\| \mathrm{p}(n) - \mathrm{F}(t_n) \right\|^2 \right\}. \tag{1}$$

The approximation error $E(\mathbf{P})$ for the curve $\mathbf{P}$ is the maximal distortion among all segments. Connectivity of the curve tangents at breakpoint is defined by type of the point (inflection/non-inflection).

The problem of error-bounded approximation for the curve $\mathbf{P}$ can be formulated in the following way: Given a curve $\mathbf{P}$, find piecewise approximation of the $\mathbf{P}$ with the spline functions, so that the approximation error $E(\mathbf{P})$ does not exceed the error threshold δ_0 and the total number of the segments M^* is minimal:

$$E(\mathrm{P}) = \max_{1 \le m \le M} \{ e(i_m, i_{m+1}) \} \le \delta_0, \tag{2a}$$

$$\text{subject to: } M^* = \min_{\{i_m\}} \{ M \}. \tag{2b}$$

To solve the problem, we have to find the minimal number of segments M and the curve partition $\{i_m\}$ taking into account continuity of the first derivative of the approximating spline at the breakpoints.

3. DP ALGORITHM FOR DISTORTION-CONSTRAINED APPROXIMATION

The approximation of the curve $\mathbf{P}$ with the minimal number of segments can be found with the DP algorithm as the shortest path in the directed acyclic feasibility graph G created on the vertices of the curve $\mathbf{P}$. Nodes of the graph G

correspond to points $\{\mathbf{p}(1), \ldots, \mathbf{p}(N)\}$ of $\boldsymbol{P}$. An edge $g(j,n)$ in graph G corresponds to the curve segment $S(j, n)$ approximated with the spline function $\mathbf{F}(t)$. The DP approach was used for optimal polygonal approximation [9−11] and digitized curves compression [4].

The edge $g(j, n)$ is called *feasible* if approximation of the segment $S(j, n)$ with the spline function $\mathbf{F}(t)$ satisfies the error bound δ_0: $e(j,n) \le \delta_0$. Weight $w(j, n)$ of the edge $g(j, n)$ is then defined by feasibility of the edge as follows:

$$w(j,n) = \begin{cases} 1; & e(j,n) \le \delta_0, \\ \infty; & \text{otherwise.} \end{cases} \quad (3)$$

Let us introduce a cost function $C(n)$ as the length (cardinality) of the shortest path in the graph $G(n)$ constructed for the first n-vertices of $\mathbf{P}$**,** where the initial condition $C(1)$ is 0. The length of the shortest path for P gives us the minimal number of approximating segments: $M^* = C(N)$. Consequently, the recursive equation for the function $C(n)$ amounts to:

$$C(n) = \min_{1 \le j < n} \{ C(j) + w(j,n) \}. \quad (4)$$

Then, we introduce an error function $D(n)$ for $G(n)$ as a L_2-norm approximation error of the corresponding sub-curve $\mathbf{P}(n) = \{(\mathbf{p}(1)), \ldots, (\mathbf{p}(n)\}$ with $C(n)$ segments:

$$D(n) = \sum_{m=1}^{C(n)} \sum_{i_m \le k \le i_{m+1}} \left\| \mathrm{p}(k) - \mathrm{F}(t_k) \right\|^2 . \quad (5)$$

We will use the error function $D(n)$ as an additional criterion to select candidate solution with smaller Integral-Squared Error among other candidates with the same cardinality $C(n)$. If we have two or more feasible candidate sub-solutions for the vertex n with same cardinality $C(n)$, we can select a candidate with the least approximation error $D(n)$:

$$D(n) = \min_{1 \le j < n} \{ D(j) + e(j,n) \mid C(n) = C(j) + 1 \}. \quad (6)$$

The number of the segments and the partition of $\mathbf{P}$ into segments are defined by backtracking of the DP solution.

The time complexity of the DP algorithm with full search is cubic. We can speed-up the process at cost of optimality. In [4], the depth of the search was constrained by a fixed size window. In [11], the stop rule depends on the approximation error for curve segment. Following this approach, we stop the DP search at a node j^* if distortion for the segment $S(j^*,n)$ becomes too large: $e(j^*,n) > \alpha\delta_0$, where $\alpha=2$ or 3. The *output-depend* time complexity of the algorithm with reduced search depth is defined as sub-cubic: $O(N^3/M^2)$.

We have considered approximation of curves under the assumption that the initial point is given. In the case of closed curves approximation, we have to find the optimal location of the start point. It can be done by the DP search in the extended graph constructed on vertices of the input curve **P** [12].

5. EXPERIMENTS AND DISCUSSIONS

The proposed algorithm was tested on the planar digitized curves obtained from bitmap objects and on other digitized curves. Some of the data are represented on Fig. 5, where N is the total number of vertices in the digitized contours. The digitized contours of bitmap objects were traced with crack-codes. The algorithms were tested on the 2.3 GHz Pentium 4. For calculation of control points for cubic Bézier spline the algorithm from [3] was applied.

The comparison of the inflection points detected before approximation with the breakpoints found with the DP algorithm shows that only some of the inflection points were used as the breakpoints. Some sharp edges labeled as the inflection points were approximated smoothly with splines, whereas only few inflection points were selected as the breakpoints (see example on Figs. 2 and 3). With the inflection points approach, we can achieve visual quality of approximation which is better than that with the fixed corner point method.

Then, we compared the number of segments obtained by the proposed optimal algorithm and by the spit-and-merge method. The original iterative breakpoint-insertion approach was modified for better performance by performing of *segments merging* and *breakpoints adjustment*. According to this approach, two adjacent segments can be merged into one segment if the approximation error for the resulting segment satisfies given error constraint. Location of the breakpoint between two adjacent segments is adjusted to

Figure 5. Test datasets: 1) 70×42 raster image of two Chinese hieroglyphs: 7 curves, N=677; 2) 292×222 raster image of Arabic character: one curve, N=1169; 3) 1001×699 raster map: 262 curves, N=33,073.

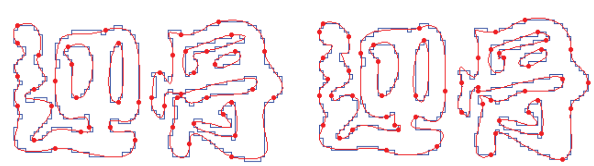

Figure 6. Example of approximation of test dataset #1 for δ_0=0.99 with the proposed algorithm, M=42 segments (left), and with the modified iterative breakpoint-insertion method, M=54 (right). The minimal number of segments M^*=42.

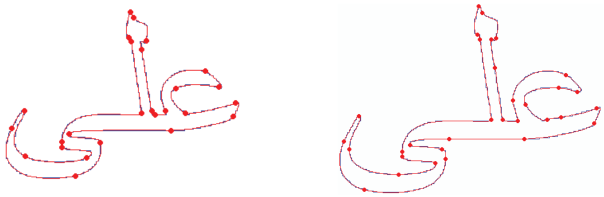

Figure 7. Example of the test data curve #2 approximation for δ_0=1 with the proposed algorithm, M=26 segments (left), and with the modified iterative breakpoint-insertion method, M=33 segments (right). The minimal number of segments is M^*=26.

decrease sum of the Integral Squared Errors for the segments. The post-processing with segments merging and vertex adjustment is repeated until no changes in the number of breakpoints. Time complexity of the heuristic algorithm is subquadratic.

Efficiency F of approximation is calculated as follows [13]: $F=(M^*/M)\cdot 100\%$, where M^* and M is the number of the segments for the optimal and evaluated solution, respectively. Optimal solution with 100% efficiency for the datasets #1, #2, and#3 can be obtained with the proposed full search DP algorithm in 1 min, 50 min and four hours, correspondingly.

The experiments have shown that efficiency of solutions obtained with the heuristic method for the test datasets is about 80% (see Figs. 6, 7 and 8). Processing time for the heuristic algorithm for the test datasets is 1.5s, 9s, and 223s, respectively.

The proposed DP algorithm with the reduced search provides good balance between efficiency and time performance. The balance we can control with the parameter α. Efficiency of the solutions for the test datasets is in the range of 98% and 100%. Processing time is 1.2s, 27s, and 670s, respectively. For large-size curves the processing time is still too big to be used in an interactive mode, but curves with few hundred vertices can be processed with high efficiency in a real time.

6. RESULTS AND DISCUSSIONS

We have developed a simple and efficient algorithm for piecewise approximation of digitized contours with cubic Bézier splines for a given threshold on maximal distortion. With the proposed DP algorithm, we can find a near-optimal solution to the problem, including the number and location of the breakpoints. Efficiency of the proposed algorithm is about 98% for the test datasets. To reduce the sensitivity of solution to the location and number of corner points in the input curve, we treat the candidate corner points as the inflection points with relaxed constraint on tangents continuity.

The algorithm has demonstrated its superiority over the heuristic algorithm with iterative breakpoint insertion and

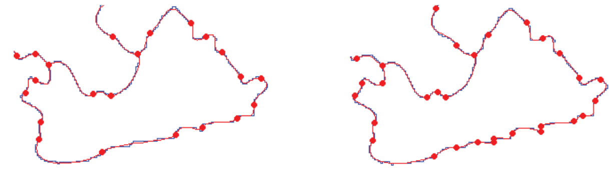

Figure 8. Example of the test dataset #3 approximation for δ_0=1.5 (fragment is shown) with the proposed algorithm, M=1064 segments (left); and with the modified iterative breakpoint-insertion method, M=1314 segments (right). The minimal number of segments is M^*=1044.

segments merging. The proposed algorithm can be used in practical applications for raster graphic vectorization, cp.

7. REFERENCES

[1] M. Plass, and M. Stone, "Curve Fitting with Piecewise Parametric Cubics, *Computer Graphics*, 17(3): 229–239, 1983.

[2] L. Shao, and H. Zhou, "Curve Fitting with Bezier Cubics, *Graphical Models and Image Processing*, 58: 223–232, 1996.

[3] P. Schneider, "An Algorithm for Automatic Fitting Digitized Curves, *Graphics Gem*, 612–626, 1990.

[4] F. W. Meier, G.M. Schuster, and A.K. Katsaggelos, "A Mathematical Model for Shape Coding with B-splines", *Signal Processing: Image Communications*, 15: 685–701, 2000.

[5] S. Pal, O. Ganguly, and P.K. Biswas, "Cubic Bézier Approximation of a Digitized Curve", *Pattern Recognition*, 40: 2730–2741, 2007.

[6] M. Safraz, and A. Masood, "Capturing Oulines of Planar Images Using Bézier Cubics", *Computers & Graphics*, 31: 719–729, 2007.

[7] A. Masood, and M. Safraz, "An Efficient Technique for Capturing 2D Objects", *Computers & Graphics*, 32: 93–104, 2008.

[8] A. Masood, and M. Sarfraz, "Capturing Outlines of 2D Objects with Bézier Cubic Approximation", *Image Vision Comput.* 27: 704–712, 2009.

[9] J. Dunham, "Optimum Uniform Piecewise Linear Approximation of Planar Curves", *IEEE Trans. Pattern. Analysis Machine Intell.*, 8: 67–73, 1986.

[10] W.S. Chan, and F. Chin, "On Approximation of Polygonal Curves with Minimum Number of Line Segments or Minimum Error", *Int. J. Comput. Geom. Appl.*, 6: 59–77, 1996.

[11] K. Schroeder, and P. Laurent, "Efficient Polygon Approximations for Shape Signatures", *Proc. Int. Conf. Image Processing-ICIP'99*, 2: 811–814, 1999.

[12] A. Kolesnikov, and P.Fränti, "Polygonal Approximation of Closed Digital Curves", *Pattern Recognition*, 40: 1282–1293, 2007.

[13] P.L. Rosin, "Techniques for Assessing Polygonal Approximation of Curves", *IEEE Trans. Pattern. Analysis Machine Intell.*, 14: 659–666, 1997.