Target Curvature Based Automatic Fairing of Planar B-Spline Curves

S. H. Xu, W. S. Li, G. Zhao
Software and Computing Program
Institute of High Performance Computing, Singapore 117528
{xush, liws, zhaog}@ihpc.a-star.edu.sg

Abstract

A new automatic local fairing technique for planar B-spline curves is presented in this paper. According to the design intent, a target curvature plot, which is provided by the designer, is used to identify 'bad' points and 'bad' curve segments. Then, the corresponding control points are modified in a process of local constrained optimization where the importance of fairness in the sense of energy minimizing versus coherence to the original design can be adjusted by the designer. Experimental results are given to show the validity of this method.

Keywords: *B-spline curves, Target curvature plots, Automatic local fairing*

1. INTRODUCTION

The notion of fairness of a curve is difficult to quantify precisely and tends to depend on the situation at hand and the subjective judgement of the individual viewing of the curve [4]. Nevertheless, according to the *principle of simplest shape* [1], beautiful objects are free of unessential features and simple in design. *A planar curve is fair if its curvature is continuous, has no redundant inflection points and flat points, and consists of only a few monotone, gradually changing pieces.* As B-spline curves are piecewise polynomial curves, the fairness has two meanings: (1) every curve segment should be fair enough; (2) the curvature plot should have as few as possible curvature extrema, inflection points and flat points.

A curve can be faired in interactive or automatic mode. Generally, automatic fairing methods are viewed as helping to obtain a fast approximation to an optimum. The 'optimally aesthetic' objects, if deemed necessary, are left to the designers, who have captured the 'design intent', to identify regions with curvature irregularities and adjust them in interactive mode [4]. This is often an experience-based, trial and error, and time-consuming process. Thus, improvement of the performance of automatic fairing will greatly decrease the design time and benefit product development.

The existing curve fairing algorithms can be classified into two categories, known as local fairing and global fairing, according to how the curve is modified to achieve fairness, i.e. locally or globally. The major disadvantages of the existing global fairing algorithms include: (1) they cannot produce fair curves consistently, (2) designers cannot involve in the fairing process, (3) they effect the whole curve while in practice designers are interested in modifying its local imperfections without altering the good parts in respect of fairness. In comparison with global fairing methods, the local fairing methods are free of the limitations (2) and (3) [7]. However, the effectiveness of existing local fairing methods is also uncertain. In fact, it is far from the truth to consider the curve fairing problem as solved.

To the best of our knowledge, the cardinal spline method proposed by Su et al. [13] may be the first attempt to fair curves locally. The curve is faired by minimizing the cumulated jump of their third derivatives at the knots where unwanted inflection points and/or wiggles occur. This algorithm can also be employed to fair B-spline curves by considering its control points as data points [13]. Later Kjellander [5] proposed a method of adjusting a 'bad' data point on a uniform parameterized curve to make the jump of the third derivative of the curve at the point to be equal to zero. Poliakoff et al. [8] introduced an extension of Kjellander's algorithm in order to apply the algorithm to non-uniformly parametrized curves, and automated this extended algorithm in their recent paper [9]. Zhang et al. [14] extended this algorithm to fairing two successive points of the curve.

The first B-spline fairing algorithm for cubic B-spline curves, *knot removal–reinsertion*, was proposed by Farin et al. [3]. When a curve is faired using this algorithm, the smoothness between the curve segments at the offending knots is increased from C^2 to C^3. Numerical examples demonstrate that generally the fairness of the curves can be improved. Sapidis and Farin [11] proposed the first automatic fairing algorithm for B-spline curves. The knot with the biggest jump of curvature variation is identified as the offending knot and the curve is faired using knot removal-reinsertion algorithm. However, Poliakoff et al. [9] reported that this algorithm frequently terminated prematurely and the global fairness indicator occasionally increased once before decreasing considerably. Another automatic algorithm, called local energy fairing, for B-spline curves of general order, was presented by Eck and Hadenfeld [2]. Their key idea is to minimize the integral of the squared l^{th} derivative of a given curve iteratively by changing only one control point where the largest improvement of the energy integral is to be expected in every step. However, minimizing the integral of the squared l^{th} derivative of a curve cannot guarantee to produce a fair curve consistently. Furthermore, the termination method of this algorithm is to restrict the number of iterations, which is very difficult to set in practice. What should be specially pointed out is that, nearly all curve fairing algorithms have been centred on examining the curve at the knots. Researches have seldom been carried out on how to evaluate and improve the internal fairness of B-spline curve segments, which is also important for the global fairness of a curve.

In this paper, we present a new algorithm for fairing planar cubic B-spline curves based on target curvature plots, which are provided by designers according to the design intent. Based on the target curvature plot, the curve's 'bad' points and 'bad' curve segments can be intuitively identified. Then, the corresponding control points are modified in a process of local constrained optimization where the importance of fairing in the sense of

energy minimizing versus coherence to the original design can be adjusted by the designer.

The rest of this paper is arranged as follows. Curve segment classification is introduced in Section 2. The target curvature based B-spline curve fairing method is described in Section 3, and an example is shown in Section 4. Section 5 draws conclusions.

2. CLASSIFICATION OF CUBIC B-SPLINE CURVE SEGMENTS

A *k^{th}-order B-spline curve* is defined by

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,k}(u)\mathbf{P}_i, \quad u \in [t_{k-1}, t_{n+1}] \tag{1}$$

where $\{\mathbf{P}_i\}$ are *control points*, $\{N_{i,k}(u)\}$ are the *k^{th}-order* B-spline basis functions defined on knot vector $T = \{t_0 = \ldots = t_{k-1}, t_k, \ldots, t_n, t_{n+1} = \ldots = t_{n+k}\}$. In this paper, attention is concentrated on planar cubic curves, i.e. $k = 4$.

For a planar curve ***C****(u)*=(*x(u), y(u)*), the curvature is defined by

$$k(u) = \frac{\dot{x}(u)\ddot{y}(u) - \dot{y}(u)\ddot{x}(u)}{(\dot{x}(u)^2 + \dot{y}(u)^2)^{3/2}} \tag{2}$$

For a B-spline curve, $k(t_i)$ is abbreviated to k_i, and the derivatives of the curvature at $\{t_j\}_{j=4}^{n-1}$ are defined by

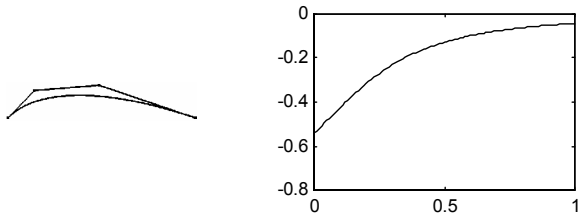
$$\dot{k}(t_j^+) = \frac{\dot{x}(t_j)\dddot{y}(t_j^+) - \dddot{x}(t_j^+)\dot{y}(t_j)}{(\dot{x}(t_j)^2 + \dot{y}(t_j)^2)^{3/2}} - 3k(t_j)\frac{\dot{x}(t_j)\ddot{x}(t_j) + \dot{y}(t_j)\ddot{y}(t_j)}{\dot{x}(t_j)^2 + \dot{y}(t_j)^2} \tag{3}$$

$$\dot{k}(t_j^-) = \frac{\dot{x}(t_j)\dddot{y}(t_j^-) - \dddot{x}(t_j^-)\dot{y}(t_j)}{(\dot{x}(t_j)^2 + \dot{y}(t_j)^2)^{3/2}} - 3k(t_j)\frac{\dot{x}(t_j)\ddot{x}(t_j) + \dot{y}(t_j)\ddot{y}(t_j)}{\dot{x}(t_j)^2 + \dot{y}(t_j)^2} \tag{4}$$

It is well known that a curve with a loop is self-intersecting, one with a cusp has a point where the unit tangent vector is discontinuous, and one with an inflection point has a point where the curvature vanishes. Loops, cusps and inflection points are called the characteristics of a curve, and these characteristics are mutually exclusive. According to its characteristics, a cubic B-spline curve segment can be classified into one of the five types known as *arch*, *one inflection point*, *cusp, two inflection points* and *loop* [12]. In engineering applications, especially in free-form shape design, curve segments that have cusp, two inflection points and loop are normally forbidden.

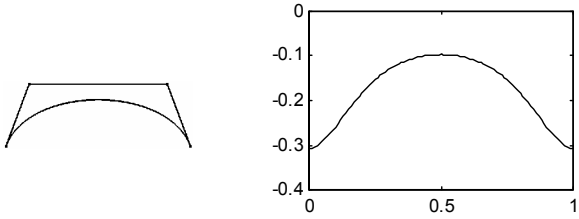
As a fair curve is free from undesirable wiggles, the curve segments should have as few as possible curvature extreme points, and a stricter version is monotone curvature [10]. To evaluate the fairness of curve segments, arches and segments with one inflection point are further classified according to the number curvature extrema. Based on the theories proven in [15], it is easy to deduce that, for an arch, the number of curvature extrema may vary from zero to three, as shown in Fig.1-4.

Apparently, an arch with no curvature extremum should be considered as a fair curve segment.



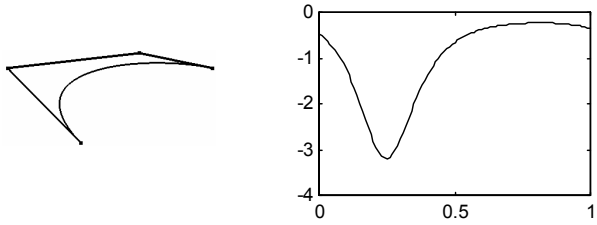
(a) (b)

Fig. 1: Arch with no extreme point: (a) the curve; (b) curvature plot of the curve



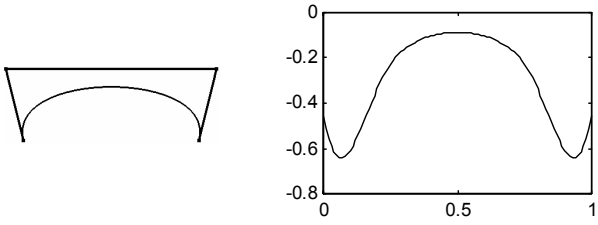
(a) (b)

Fig. 2: Arch with one extreme point: (a) the curve; (b) curvature plot of the curve



(a) (b)

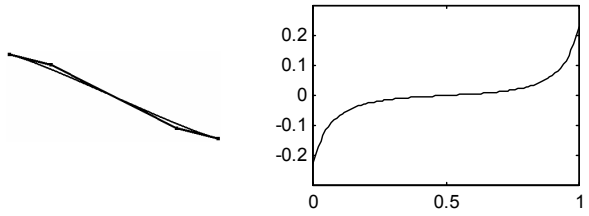
Fig. 3: Arch with two extreme points: (a) the curve; (b) curvature plot of the curve



(a) (b)

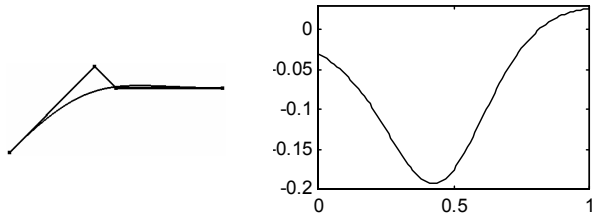
Fig. 4: Arch with three extreme points: (a) the curve; (b) curvature plot of the curve

For a curve segment with one inflection point, theoretically the number of curvature extrema might be up to four [15]. However, in practice, a curve segment with one inflection point and three or four curvature extrema rarely exists. So, we only consider the cases that it has up to two curvature extrema, as shown in Fig. 5-7. Similarly, the curve segments with no curvature extrema are fair and are preferred in applications.



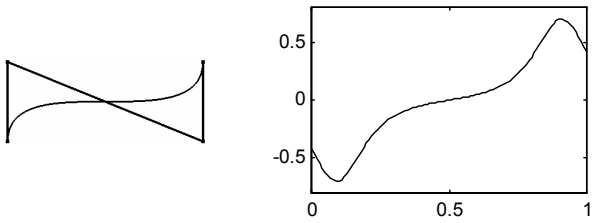
(a) (b)

Fig. 5: Curve segment with one inflection point and no extreme point (a) the curve; (b) curvature plot of the curve



(a) (b)

Fig. 6: curve segment with one inflection point and one extreme point (a) the curve; (b) curvature plot of the curve



(a) (b)

Fig. 7: curve segment with one inflection point and two extreme points (a) the curve; (b) curvature plot of the curve

To judge whether a curve segment (arch or segment with one inflection point) has curvature extrema, the derivatives of the curvatures at endpoints can be used. If $\dot{k}(t_i^+)\cdot\dot{k}(t_{i+1}^-)<0$, there must be at least one curvature extremum. Obviously, a curve segment with odd curvature extreme point(s), i.e. one or three, should meet this condition. To determine whether a curve segment has zero or two curvature extrema, i.e. $\dot{k}(t_i^+)\cdot\dot{k}(t_{i+1}^-)\geq 0$, we can use the dichotomy method to further subdivide the curve segment

3. B-SPLINE CURVE FAIRING

In this section, we describe how to identify 'bad' points and present a local constrained optimization method used to modify the control points corresponding to the bad points or bad curve segments.

3.1 Bad Point Identification

The target curvature plot can be used to identify a curve's undesirable features, i.e. redundant curvature extreme and inflection points. The design of a target curvature plot should satisfy the following criteria.

(1) The proposed method is intended for finish fairing. This means that the target curvature plot should be based on the real curve shape and its curvature distribution. Large scale and severe curvature changing is unexpected.

(2) A target curvature plot only intends to reflect the 'tendency' of the final curvature distribution. So, line segments will be used in the target curvature plot for efficiency.

(3) To describe the 'tendency' as accurate as possible, the target curvature plot should provide the following information: (i) curvature extreme positions, (ii) curvature segments in which inflection points lie, and (iii) tendency of the curvature distribution.

(4) A target curvature plot should consist of as few as possible monotone pieces.

Fig. 8 shows an example. In the picture, the target curvature is drawn as dashed line segments whereas the real curvature is drawn as solid curve segments. The break points of a target curvature plot can only be the knots. In other words, the target curvature between any two neighbouring knots is linear distributed.

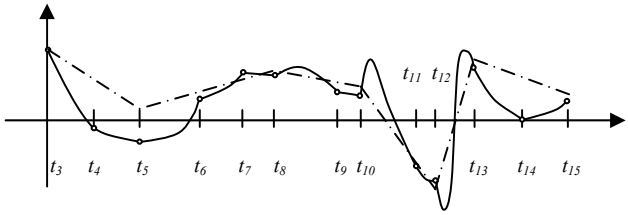


Fig. 8: Target curvature plot

Let (t_i,k_i^t) $(i=k-1,n+1)$ be the point coordinates sampled on the target curvature plot, and (t_i,k_i) be the point coordinates sampled on the real curvature plot. Bad points and bad curve segments can be identified using the following rules.

(1) If $(k_i-k_{i-1})(k_i^t-k_{i-1}^t)<0$ *or* $(k_{i+1}-k_i)(k_{i+1}^t-k_i^t)<0$, i.e. the tendencies of the target curvature segment and the real curvature segment between knots t_{i-1} and t_i *or* between knots t_i and t_{i+1} are different, this usually results in an unexpected curvature extremum at t_i and t_i is identified as a bad point, such as t_7, t_8 and t_{14} in Fig.8. Bad points of this type are called *type A* bad points. They are ranked according to $\left|k_i-\frac{(t_i-t_{i-1})\cdot k_{i+1}+(t_{i+1}-t_i)\cdot k_{i-1}}{t_{i+1}-t_{i-1}}\right|$.

(2) If $(k_i\cdot k_{i+1})(k_i^t\cdot k_{i+1}^t)\leq 0$, there is a redundant inflection point between t_i and t_{i+1}. If $k_i\cdot k_i^t>0$, t_{i+1} is a bad point, such as t_4 in Fig.8; else if $k_{i+1}\cdot k_{i+1}^t>0$, t_i is a bad point, such as t_5 in Fig.8. Bad points of this type are called

type B bad points. They are ranked according to $|k_i|$ (if t_i is the bad point) or $|k_{i+1}|$ (if t_{i+1} is the bad point). If the left part of the above formula is equal to zero, such as t_{14} in Fig.8, the bad point, for example t_i, can be ranked according to $|k_{i+1}|$. In the cases that a bad point has two ranked values, the greater one is chosen.

(3) If the curvature segment between knots t_i and t_{i+1} has extreme points as discussed in Section 2, t_i and t_{i+1} are identified as bad point pair, such as t_8 and t_9 in Fig.8. Bad point pairs of this type are called *type C* bad points. They are ranked according to the result of *Max*$\{\left|k'(t_i^+)-(k_{i+1}-k_i)/(t_{i+1}-t_i)\right|,\left|k'(t_{i+1}^-)-(k_{i+1}-k_i)/(t_{i+1}-t_i)\right|\}$

3.2 Local Constrained Optimization

Different from the local energy fairing algorithm [10], which fairs the curve by minimizing the energy of the curve with respect to the worst control point in one step, our goal function for curve fairing is

$$E=(1-\alpha)\cdot\int_{t_{k-1}}^{t_{n+1}}(\tilde{\mathbf{C}}''(u))^2\,du+\alpha\cdot\sum_{i=0}^{n}\beta_i\cdot(\tilde{\mathbf{P}}_i-\mathbf{P}_i)^2 \qquad (5)$$

where $\beta_i=\int_{t_{k-1}}^{t_{n+1}}(N_{i,k}''(u))^2\,du$. When $\{\mathbf{P}_i\}$ are replaced by $\{\tilde{\mathbf{P}}_i\}$, $\mathbf{C}(u)$ is replaced by $\tilde{\mathbf{C}}(u)$. α is at the user's disposal, expressing the relative importance assigned to fairing versus adherence to the original design, i.e. a smaller α represents greater emphasis on fairing. As we discussed in Section 1, the energy method, such as the first part of Eq.(5), is not consistent in producing fair curves. In comparison with the energy method, the linear combination shown in Eq. (5) gives designers more flexibility and makes it more feasible in producing fair curve than the existing algorithms. When more than one control points are modified, the influence of each control point could be self-adjusted.

Fairing more than two consecutive control points in one step may result in curve segments quite different from the original ones and violate the shape-preserving requirement of the fairing process. According to our experiments, normally up to two consecutive control points can be modified at the same time.

When t_i is identified as a bad point, three control points, $\{\mathbf{P}_j\}(j=i-3,i-2,i-1)$, are involved in fairing the curve at this point as a result of the local support property of B-spline curves [7]. According to the importance of the three control points to the bad point, we modify $\mathbf{P}_{i-2}$ when t_i is a bad point.

3.2.1 Modifying One Control Point

When control point $\mathbf{P}_m$ is modified, the shape of the curve segments that defined on $[a,b]$ will be changed. Here $a=\max\{t_m,t_{k-1}\}$, $b=\min\{t_{m+k},t_{n+1}\}$. $\tilde{\mathbf{P}}_m$ is determined by setting the derivative of *E* with respect to $\tilde{\mathbf{P}}_m$ to zero, i.e.

$$\partial E/\partial\tilde{\mathbf{P}}_m=0 \qquad (6)$$

where, $\tilde{\mathbf{P}}_i=\mathbf{P}_i(i=0,...m-1,m+1,...,n)$.

The solution is

$$\tilde{\mathbf{P}}_m=\alpha\cdot\mathbf{P}_m+(1-\alpha)\cdot\sum_{\substack{i=i0,\\ i\neq m}}^{i1}\delta_i\cdot\mathbf{P}_i \qquad (7)$$

where $\delta_i=-\int_a^b(N_{i,k}''(u)\cdot N_{m,k}''(u))du/\int_a^b(N_{m,k}''(u))^2\,du$, $i0=max\{0,\ m-k+1\}$, $i1=min\{\ m+k-1,\ n\}$.

3.2.2 Modifying Two Control Points

When control points $\mathbf{P}_m$ and $\mathbf{P}_{m+1}$ are modified, the shape of the curve segments that defined on $[a,b]$ will be changed. Here $a=\max\{t_m,t_{k-1}\}$, $b=\min\{t_{m+k+1},t_{n+1}\}$.

$\tilde{\mathbf{P}}_m$ and $\tilde{\mathbf{P}}_{m+1}$ are determined by setting the derivative of *E* with respect to $\tilde{\mathbf{P}}_m$ and $\tilde{\mathbf{P}}_{m+1}$ to zero, i.e.

$$\partial E/\partial\tilde{\mathbf{P}}_j=0 \qquad j=m,m+1. \qquad (8)$$

where $\tilde{\mathbf{P}}_i=\mathbf{P}_i(i=0,...m-1,m+2,...,n)$.

The solution is obtained by solving the equation system shown as follows.

$$\begin{bmatrix}1 & (1-\alpha)\cdot\lambda_1\\ (1-\alpha)\cdot\lambda_2 & 1\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{P}}_m\\ \tilde{\mathbf{P}}_{m+1}\end{bmatrix}=\begin{bmatrix}\alpha\cdot\mathbf{P}_m+(1-\alpha)\cdot\sum_{\substack{i=i0,\\ i\neq m,m+1}}^{i1}\delta_i\cdot\mathbf{P}_i\\ \alpha\cdot\mathbf{P}_{m+1}+(1-\alpha)\cdot\sum_{\substack{i=i0,\\ i\neq m,m+1}}^{i1}\delta_i\cdot\mathbf{P}_i\end{bmatrix} \qquad (9)$$

where

$$\lambda_1=\frac{\int_a^b N_{m,k}''(u)\cdot N_{m+1,k}''(u)du}{\int_a^b(N_{m,k}''(u))^2\,du},\quad \lambda_2=\frac{\int_a^b N_{m,k}''(u)\cdot N_{m+1,k}''(u)du}{\int_a^b(N_{m+1,k}''(u))^2\,du},$$

$$\delta_i=-\frac{\int_a^b(N_{i,k}''(u)\cdot N_{m,k}''(u))du}{\int_a^b(N_{m,k}''(u))^2\,du},\ i0=max\{0,\ m-k+1\},\ i1=min\{m+k-1,\ n\}.$$

According to our tests, when two successive control points are modified using Eq.(9), α should be set larger than the weight used in modifying one control point. Normally it is set to be >0.9.

3.3 The Fairing Procedure

When a curve is to be faired, typically the curve is also expected to satisfy certain constraints [8], such as interpolation to certain data points and/or derivative values, local convexity besides approximation to data with tolerance. The discussion of constraints is beyond the scope of this paper. Here, we call them constraints in general.

Before describing the fairing steps, we first introduce the concept of modifiability of a bad point. Besides the ranked value, each bad point is also given a value of modifiability, a Boolean value used to indicate whether this bad point is allowed to be modified in next iteration. The setting of a bad point's modifiability value is according to the following rules: (1) initially every bad point is modifiable (*TRUE*); (2) once a bad point is faired, failed or successfully, it cannot be faired in next iteration (*FALSE*). If successfully faired, the modifiability values of the relevant bad points are re-set to *TRUE*. For example, if t_i is a bad point, according to the introduction in subsection 3.2, control point $\mathbf{P}_{i-2}$

will be modified. The bad points located within the affected area of $\mathbf{P}_{i-2}$ (excluding t_i) are called the relevant bad points of t_i.

Fairing Steps

1) Identify and rank *Type A* bad points;
2) If there are modifiable bad points and t_i is the 'worst' one among them, compute $\tilde{\mathbf{P}}_{i-2}$ using Eq.(7) and set t_i's modifiability value to *FALSE*; otherwise go to step 4);
3) If new curve $\tilde{\mathbf{C}}(u)$ defined by $\{\mathbf{P}_0,...,\mathbf{P}_{i-3},\tilde{\mathbf{P}}_{i-2},\mathbf{P}_{i-1},...,\mathbf{P}_n\}$ fulfils the constraints, let $\mathbf{C}(u)=\tilde{\mathbf{C}}(u)$ and re-set the modifiability values of the relevant knots to *TRUE*, go to step 1). Otherwise, do not replace $\mathbf{P}_{i-2}$ and go to step 2);
4) Identify and rank *Type B* bad points;
5) If there are modifiable bad points and t_i is the 'worst' one among them, compute $\tilde{\mathbf{P}}_{i-2}$ using Eq.(7) and set t_i's modifiability value to *FALSE*; otherwise go to step 7);
6) If new curve $\tilde{\mathbf{C}}(u)$ defined by $\{\mathbf{P}_0,...,\mathbf{P}_{i-3},\tilde{\mathbf{P}}_{i-2},\mathbf{P}_{i-1},...,\mathbf{P}_n\}$ fulfils the constraints, let $\mathbf{C}(u)=\tilde{\mathbf{C}}(u)$ and re-set the modifiability values of the relevant knots to *TRUE*, go to step 4). Otherwise, do not replace $\mathbf{P}_{i-2}$ and go to step 5);
7) Identify and rank *Type C* bad points;
8) If there are modifiable bad points, and t_i and t_{i+1} are the 'worst' pair among them, compute $\tilde{\mathbf{P}}_{i-2}$ and $\tilde{\mathbf{P}}_{i-1}$ using Eq.(9) and set their modifiability values to *FALSE*; otherwise stop;
9) If $\tilde{\mathbf{C}}(u)$ defined by $\{\mathbf{P}_0,...,\mathbf{P}_{i-3},\tilde{\mathbf{P}}_{i-2},\tilde{\mathbf{P}}_{i-1},\mathbf{P}_i,...,\mathbf{P}_n\}$ fulfils the constraints, let $\mathbf{C}(u)=\tilde{\mathbf{C}}(u)$ and re-set the modifiability values of the relevant knots to *TRUE*, go to step 7). Otherwise, do not replace $\mathbf{P}_{i-2}$ and $\mathbf{P}_{i-1}$, and go to step 8).

In contrary to Sapidis' algorithm, where the 'worst point' is faired in each iteration, in cases that the worst point or the worst point pair could not be faired, the less worst points or point pairs will be used to fair the curve to the greatest potential.

4. EXAMPLE

The following example is from reverse engineering applications. Its data originate from a section of a car body (see Fig. 9(a)). The graphics output consists of two kinds of figures, namely curve figures and curvature plots. Solid lines in the curvature plot of the initial curve indicate the real curvature plot of the curve, whereas dashed lines indicate the target curvature plot. Sapidis' algorithm is constrained with tolerance to make it comparable with our method, and the tolerance constraints for Sapidis' algorithm in this example are the same as those used for our method.

The example shows how the presented algorithm treats a curve, consisted of 6 cubic B-spline curve segments (see Fig. 9(a)), of a partial section of a car body. The shape of this curve is very simple. The main problem of this curve is that it has three redundant inflection points. Setting tolerance constraint to be $0.5mm$, $\alpha=0.1$ for modifying one control point and $\alpha=0.95$ for modifying two control points, the faired curve with the presented algorithm is depicted in Fig.9(c). The curve faired with Sapidis' algorithm is shown in Fig. 9(e) for comparison.

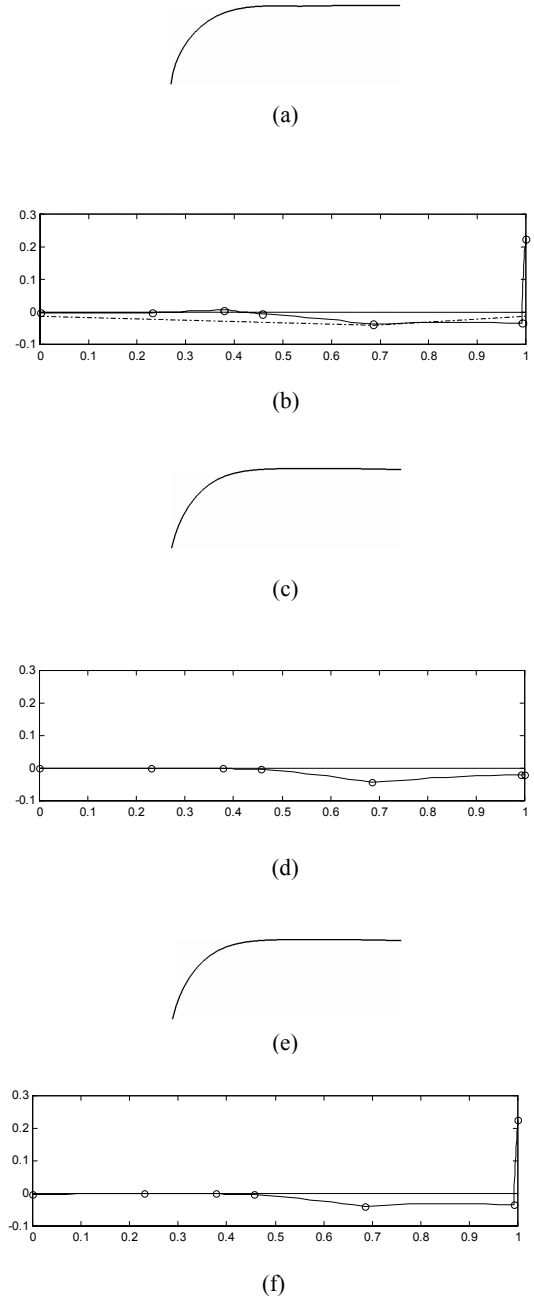


Fig. 9: The curve (a) and its curvature plot (b).The curve faired using our method (c) and its curvature plot (d).The curve faired using Sapidis' algorithm (e) and its curvature plot (f)

According to the above example, we can see that the fairing result using our method is better than that faired with Sapidis' algorithm in terms of curvature distribution.

Due to the need of integral computations, our approach is computational heavier. But it is fast enough for interactive design.

According to our test on a PC-PIII/450Mhz, when the curve is faired with the presented algorithm, the time consumed for this example is 0.04 CPU-seconds, without considering the time consumed for drawing target curvature plots. When the curve is faired with Sapidis' algorithm, the time consumed is 0.02 CPU-seconds.

5. CONCLUSIONS

A target curvature plot based algorithm for fairing planar cubic B-spline curves is presented. Different from the existing algorithms, the fairness of curve segments is considered in the fairing process. In order to identify bad points and bad point pairs, a target curvature plot, which is drawn by the designer according to the design intent, is employed. The bad points and point pairs are modified by local constrained optimization. The goal function is a weighted equation which can be used to adjust the importance of fairing and coherence to the initial design.

Currently this algorithm is restricted to cubic planar B-spline curves although the bad point modification scheme can be also used for fairing B-spline curves of general order.

6. REFERENCE

[1] Burchard HG, Ayers JA, Frey WH, Sapidis N. Approximation with aesthetic constraints. In: Sapidis N eds. Designing fair curves and surfaces: shape quality in geometric modeling and CAD, Philadelphia: SIAM, 1994. 3-28.

[2] Eck M, Hadenfeld J. Local energy fairing of B-spline curves. Computing/Supplement. 1995;10:129-147.

[3] Farin G, Rein G, Sapidis N, Worsey AJ. Fairing cubic B-spline curves. CAGD 1987;4: 91-103.

[4] Farin G, Sapidis N. Curvature and the fairness of curves and surfaces. IEEE Comput. Graph. Appl. 1989;9(2):52-57.

[5] Kjellander JA. Smoothing of cubic parametric splines. CAD 1983;15(3):175-179.

[6] Pigounakis KG, Kaklis PD. Convexity-preserving fairing. CAD 1996;28(12):981-994.

[7] Pigounakis KG, Sapidis NS, Kaklis PD. Fairing spatial B-spline curves. J. Ship Research 1996;40(4):351-367.

[8] Poliakoff JF. An improved algorithm for automatic fairing of non-uniform paramatric cubic splines. CAD 1996;28(1):59-66.

[9] Poliakoff JF, Wong YK, Thomas PD. An automatic curve fairing algorithm for cubic B-spline curves. J. Comput. Appl. Math. 1999;102:73-85.

[10] Roulier J, Rando T, Piper B. Fairness and monotone curvature. In: Chui CK eds. Approximation Theory and Functional Analysis, Academic Press, 1991. 177-198.

[11] Sapidis N, Farin G. Automatic fairing algorithm for B-spline curves. CAD 1990;22(2):121-129.

[12] Stone MC, DeRose TD. A geometric characterization of parametric cubic curves. ACM Transactions on Graphics 1989;8(3):147-163.

[13] Su BQ, Liu DY. Computational Geometry – Curve and Surface Modeling. Academic Press, 1989.

[14] Zhang CM, Zhang PF, Cheng FH. Fairing spline curves and surfaces by minimizing energy. CAD 2001;33(13):913-923

[15] Walton DJ and Meek DS. Curvature extrema of planar parametric polynomial cubic curves. J. of Computational & Applied Math. 134 (2001) :69-83

About the author

S. H. Xu is a senior research engineer at Institute of High Performance Computing, Singapore. His contact email is xush@ihpc.a-star.edu.sg.

W. S. Li is a post-doctoral research fellow at Institute of High Performance Computing, Singapore. His contact email is liws@ihpc.a-star.edu.sg.

G. Zhao a post-doctoral research fellow at Institute of High Performance Computing, Singapore. His contact email is zhaog@ihpc.a-star.edu.sg.