Computer Aided Drafting, Design and Manufacturing
Volume 26, *Number* 1, *March* 2016, *Page* 26

CADDM

An explicit fairing indicator for 2D curves

Han Xiaoguang[1], *Zhang Qing* [2], *Dong Guangchang*[3], *Liu Ligang*[4]

1.Department of Computer Science, The University of Hong Kong, Hong Kong 999077, China

2. Shenzhen Institute of Information Technology, Shenzhen 518172, China;

3. School of Mathematical Sciences, Zhejiang University, Hangzhou 310027, China;

4. School of Mathematical Sciences, University of Science and Technology of China, Hefei 230026,China.

Abstract: Resorting to cubic spline function instead of parametric spline representation, an explicit fairness indicator and an efficient fairing algorithm for 2D curves are presented. The input point sequence is firstly partitioned into several overlapped function segments. For each segment, a cubic spline function is used as the representation tool which entails a polyline approximation of curvature plot. Based on the extrinsic relationship between the polyline and the positions of data points, a coarse-to-fine faring method is proposed which efficiently identifies and eliminates the unnecessary inflection points. Our algorithm generates the best results to date, which is validated by numerous practical examples.

Key words: extrinsic fairness indicator; polyline curvature plot; curve fairing

1 Introduction

Input a sequence of points, the first step of curve design is to interpolate them using mathematical tools (for examples, the parametric spline). Curve smoothing, as another important procedure, is aiming to identify "bad" points and modify their positions to ensure visual pleasing of the generated curve.

To quantify "bad" or "good" of the data points, curvature plot is used by all previous works as the fairness indicator [1-6]. The "bad" points are usually located as the jump parts of curvature plot in these papers, and all the algorithms are designed to eliminate these jumps by modifying the positions of input points. However, this is very challenging due to the complicated and implicit relationship between displacements of data points and the curvature represented using parametric spline.

There also exists another category of methods which formulate curve smoothing as an energy minimization problem. They use data term and smoothness term to form the cost function, where the smoothness term is usually represented by the integral of the square of l_{th} derivative [7-8] and the data term defines the closeness to input data using displacements of data points directly. A weight is also necessary to balance the smoothness and the closeness. Because of the existence of local minima, a curve can still own bumps after smoothing. This can be observed by one of our experiments. Another drawback of these methods is that the weight is usually input-dependent and is very difficult to determine.

To address the above issues, our paper presents a novel curve smoothing algorithm. The key idea based on an observation that the curvature plot of cubic spline function can be well approximated by its second derivative, which is a polyline. Furthermore, there exists an extrinsic relationship between this polyline and displacements of the input data points. Our approach is therefore starting from partitioning data points into several overlapped function segments and each of them is represented by a cubic spline function. Then, our fairing procedure is performed segment by segment iteratively. For each segment, we use the polyline as fairness indicator and a coarse-to-fine smoothing algorithm is also developed to identify and eliminate unnecessary inflection points and vibrations of the curvature directly. Compared with

Project item: Supported by the National Natural Science Foundation of China (61222206, 11526212).

Corresponding author: Liu Ligang, Male, Ph.D., Professor, E-mail: lgliu@ustc.edu.cn.

previous methods, our algorithm produces the best results to date.

In summary, the contributions of this work are twofold.

An explicit fairness indicator is presented based on cubic spline function, and the relationship between displacements of data points and this indicator is derived.

A coarse-to-fine smoothing method for a function segment and a novel curve fairing pipeline for arbitrary curves are developed which give rise to the best results to date.

2 Related work

We give a detailed literature review of all existing curve fairing methods in this section. Comparisons between different fairness criteria and fairing methods can also be found in reference [9]. In general, all previous algorithms can be classified into the following two categories.

2.1 Global fairing methods

The global fairing methods are based on minimizing certain energy over curvature of the curves. The method in reference [7] used the square of the second derivative norm as the fairness criterion and considered fairing Bezier curves as an approximation problem with constraints. The algorithm proposed in reference [10] applied nonlinear optimization techniques to minimize a fairness functional based on variation of curvature while satisfying geometric interpolarity constraints on positions, normals, and curvatures. The variational problem formulation leads to a nonlinear system of equations in reference [11], which is solved numerically. Its fairness criterion is defined as the linear combination of the square of the second and the third derivative norm. The integral of the square of the l_{th} derivative of a given curve is minimized in reference [8] by modifying the control points which is expected to reduce the energy integral most significantly at each iteration. An automatic shape control method is described in reference [12] as a nonlinearly constrained optimization problem whether monotone curvature variation (MCV) constraints are taken into account to satisfy the monotone curvature pieces requirement from the designer. Recently, the reference [13] introduced a new automatic fairing algorithm which is based on the modification of the control points set using a method based on a suitable integral spline operator. Due to well studied global smoothing methods, some mathematical packages such as Matlab Spline Toolbox also provided curve smoothing algorithm, which is based on minimizing weighted combination of displacements of data points (data term) and smoothness energy (smoothness term) directly. Amati G. [14] proposed another kind of global algorithm based on a multi-level representation of cubic B-spline curves, which decomposed a curve into low resolution and details function parts, and it used the detail function to identify and remove bad control points.

2.2 Local fairing methods

Local fairing methods consider modifying either the control points or the knot vector to fair curves and have advantages over global fairing methods in terms of automaticity and local affection. A method for fairing a cubic parametric spline curve is proposed in reference [1] by modifying the bad data points such that the jumps in the third derivatives of the curve at those points are zeros. Reference [3] and [4] then extended this work to non-uniform parametric curves, where they faired a single data point in each modification step, and it was also extended by reference [6] to fair two successive points on the curve simultaneously. A cubic B-spline curve fairing algorithm was developed by reference [2] based on a knot removal-reinsertion process on removing the identified bad data points. This process can be iteratively repeated until a satisfactory curve is obtained. Sapidis and Farin[15] proposed an automatic fairing algorithm for B-spline curves. The algorithm firstly identified the offending knot with the biggest jump in curvature variation, and then the curve is faired by changing the position of that point. Pigounakis et al.[5] developed a two-stage automatic algorithm for fairing cubic parametric B-splines. The first stage is a global procedure with constraints of local-convexity, local-tolerance and end constraints. The second stage is a local fine fairing procedure which employed an iterative knot-removal and knot-insertion technique to reduce curvature-slope discontinuity.

The proposed algorithm in this paper belongs to local fairing methods, which is also aiming to modify the positions of points to eliminate the irregularities of the curve. Different from all the above local fairing works, this paper firstly involves cubic spline function for curve smoothing which entails a polyline approximation of curvature plot and an extrinsic relationship between it and the displacements of data points. This is also the basic reason why our algorithm outperforms all existing methods.

3 Extrinsic fairness indicator

A function segment is defined as a sequence of points which can be represented by a function. Our fairing algorithm for arbitrary curvess described in

next section is based on partitioning data points into several overlapped function segments and faring them one by one iteratively. In this section, we give our novel fairness indicator and the relationship between it and displacements of input points for a function segment. Denote $\{P_i=(x_i,y_i)\}(i=1,2,\cdots,n)$ are the sequential data points of one function segment. We have $x_i<x_{i+1}$ $(i=1,2,\cdots,n-1)$.

3.1 Cubic spline function

$\{P_i\}$ can be represented by a C_2 piecewise cubic polynomial function $y(x)$, which is called *cubic spline function*.

We represent $y(x)$ in each interval as

$$y(x)=A_i(x-x_i)^3+B_i(x-x_i)^2+C_i(x-x_i)+D_i,\quad x\in[x_i,x_{i+1}] \tag{1}$$

for i=1, 2⋯, n–1.

Denote $y''(x_i)=y_i''$ and $y(x_i)=y_i(i=1,2,\cdots,n)$, we can obtain

$$y(x)=y_i+m_i(x-x_i)-\frac{(x-x_i)(x_{i+1}-x)}{(x_{i+1}-x_i)}[c_i(2x_{i+1}-x-x_i)+c_{i+1}(x_{i+1}+x-2x_i)] \tag{2}$$

for $x\in[x_i,x_{i+1}], i=1,2,\cdots,n-1$, where $m_i=(y_{i+1}-y_i)/(x_{i+1}-x_i)(i=1,2,\cdots,n-1)$, $c_i=y_i''/6(i=1,2,\cdots,n)$.

Its derivatives are

$$\begin{aligned}y'(x_i+0)&=m_i-(x_{i+1}-x_i)(2c_i+c_{i+1})\\ y'(x_{i+1}-0)&=m_i+(x_{i+1}-x_i)(c_i+2c_{i+1})\end{aligned} \tag{3}$$

Based on the continuity of $y'(x)$, specifically $y'(x_i+1)=y'(x_i-0)$, we can obtain the well-known Three-Moment Equations:

$$(x_i-x_{i-1})c_{i-1}+2(x_{i+1}-x_{i-1})c_i+(x_{i+1}-x_i)c_i+1=m_i-m_{i-1} \tag{4}$$

for i=2,3,⋯, n–1.

This is a tri-diagonal linear system.

However, this system has n–2 equations and n variables $c_i(i=1,2,\cdots,n)$. In order to generate a unique cubic spline, two other conditions must be imposed upon the system. There are three kinds of endpoint conditions in practical lofting: ① y_1' and y_n' are given; ②$c_1=c_n=0$; ③$c_1=c_2$ and $c_{n-1}=c_n$. We use the third endpoint constraint which produces best results and requires no user interaction. We will derive the spline form from it in the following.

Starting from $c_1=c_2$, we convert each equation of Eq.(4) into its equivalent form $c_i+a_ic_{i+1}=b_i(i=1,2,\cdots,n-1)$ by doing a forward substitution. Hence $a_1=-1$ and $b_1=0$. From here, we substitute this value in Eq.(4) and get

$$a_i=\frac{x_{i+1}-x_i}{2(x_{i+1}-x_{i-1})-a_{i-1}(x_i-x_{i-1})} \tag{5}$$

$$b_i=\frac{m_i-m_{i-1}-(x_i-x_{i-1})b_{i-1}}{2(x_{i+1}-x_{i-1})-a_{i-1}(x_i-x_{i-1})} \tag{6}$$

for i=2, 3⋯, n–1.

Similarly, starting from $c_{n-1}=c_n$, we convert each equation of Eq.(4) into the form $c_i+a_i^*c_{i-1}=b_i^*(i=n,n-2,\cdots,2)$ by doing a backward substitution. Therefore, from $a_n^*=-1$ and $b_n^*=0$ we have

$$a_i^*=\frac{x_i-x_{i-1}}{2(x_{i+1}-x_{i-1})-a_{i+1}^*(x_{i+1}-x_i)} \tag{7}$$

$$b_i^*=\frac{m_i-m_{i-1}-(x_{i+1}-x_i)b_{i+1}^*}{2(x_{i+1}-x_{i-1})-a_{i+1}^*(x_{i+1}-x_i)} \tag{8}$$

for i= n–1, n–2,⋯, 2.

For a fixed i=j, from Eq.(4) we have

$$c_j=\frac{m_j-m_{j-1}-(x_j-x_{j-1})b_{j-1}-(x_{j+1}+x_j)b_{j+1}^*}{2(x_{j+1}-x_{j-1})-a_{j-1}(x_j-x_{j-1})-a_{j+1}^*(x_{j+1}-x_j)} \tag{9}$$

and

$$\begin{aligned}c_i&=b_i^*-a_i^*c_{i+1},i>j\\ c_i&=b_i^*-a_i^*c_{i-1},i>j\end{aligned} \tag{10}$$

Thus we obtain all formulas of $c_i(i=1,2,\cdots,n)$.

3.2 Approximate curvature plot and fairness indicator

We firstly review some fairness criteria proposed previously. Most of them are closely related to the distribution of curvature of a curve. Among which, the widely well-known ones are listed as:

A curve is characterized as fair, if:

(1) It is C^2 continuous and has uniform curvature variation without any unnecessary inflection points[16];

(2) Its curvature plot is continuous, has the appropriate sign (if the convexity of the curve is prescribed), and it is as close as possible to a piecewise monotone function with as few monotone pieces as possible[15];

(3) The curvature plot should be free of any unnecessary variation, i.e., the distribution of curvature

on a fair curve must be as uniform as possible[3];

(4) Its curvature plot is continuous and consists of relatively few monotone pieces[17].

Based on these criteria, our fairing objective is to eliminate unnecessary zero-value points and unnecessary vibrations of the curvature plot (zero-value points of curvature plot means inflections of the curve).

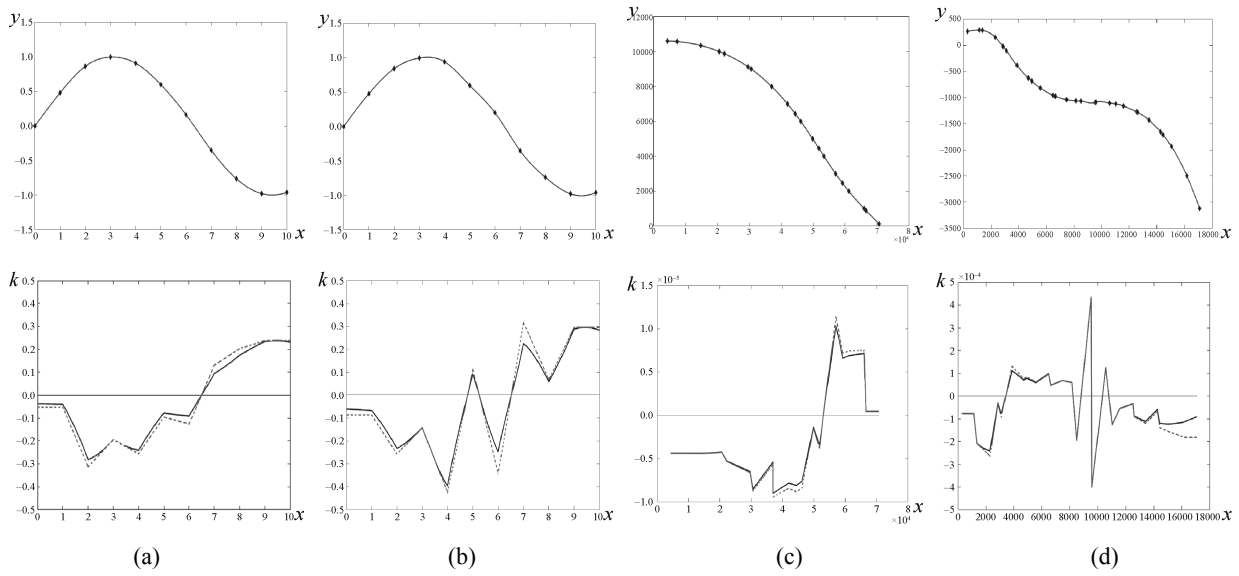
We give a polyline approximation of the curvature plot in the following. Based on it, zero-value points and vibrations can be identified directly. We will also derive the relationship between this polyline and displacements of $\{y_i\}$ in the next subsection.

The curvature formula of cubic spline function is $\kappa = \dfrac{y''}{(1+y'^2)^{3/2}}$. In our paper, we found that it can be approximated by y'', which is a polyline. The curvature $\kappa(x)$ shares similarities with the second derivative $y''(x)$ in three aspects. Firstly, it is obvious that they have same signs. Secondly, there exist two positive constants C_1 and C_2 which satisfy

$$\kappa'(x_i+0)-\kappa'(x_i-0)=C_1\frac{y'''(x_i+0)-y'''(x_i-0)}{[1+y'(x_i)^2]^{\frac{3}{2}}}=C_2\left[\frac{y''_{i+1}-y''_i}{x_{i+1}-x_i}-\frac{y''_i-y''_{i-1}}{x_i-x_{i-1}}\right] \quad (11)$$

for i=1, 2,⋯, n.

This shows that $\kappa(x)$ and $y''(x)$ have the same turning direction and amplitude (with a positive ratio) at x_i(i=1,2,⋯,n). In order to well-represent the structure of input curve, the sampled data points are usually distributed in varying density. Generally speaking, more data points are needed in the part the curve bends too much. Therefore, in each interval (x_i, x_{i+1}), the curve is nearly flat everywhere which means y' is nearly a constant. This is to say $y''(x)$ is directly proportional to $\kappa(x)$ inside every interval. This is the third aspect. Fig. 1 shows the effectiveness of the approximation by two synthetic examples and two industrial examples. From the above observations, the polyline $y''(x)$ can reveal the vibrations and zero-value points of the curvature $\kappa(x)$ very well. We use this polyline to express the fairness instead of curvature plot.



(a) (b) (c) (d)

Fig. 1 Examples of cubic spline curves with their curvatures and second derivatives. Upper row: plots of $y(x)$ (the data points are shown in diamond). Lower row: plots of $k(x)$ (in solid line) and $y''(x)$ (in dashed line). (a)-(b) The same segments of $y=\sin\dfrac{x}{2}$ with different random Gaussian noises. (c) A Waterline segment of a ship hull. (d) A Buttock Line segment of a ship hull.

For convenience, we take $y''/6$ as our fairness indicator and its turning points are located at (x_i, c_i) (i=1,2⋯,n). It is also called x–c plot. Based on this polyline, an inflection point exists when $c_ic_{i+1}<0$ happens, and a vibration of curvature is identified at the position $e_ie_{i+1}<0$ happens $\left(e_i=\dfrac{c_{i+1}-c_i}{x_{i+1}-x_i}-\dfrac{c_i-c_{i-1}}{x_i-x_{i-1}}\right)$.

3.3 Displacing data points

In this section, we derive the relationship between the polyline and movements of the data points, i.e. how c_i(i=1, 2,⋯, n) and e_i(i=2,⋯, n–1) change according to displacements of y_i.

We firstly introduce the unit spline functions which help the analysis. Denote $b_i(x)$ (i=1,2,⋯,n) as the

cubic spline functions defined on the input points $\{(x_i, \delta_i^j), i = 1,2,\cdots,n\}$, respectively, where δ_i^j is the Kronecker delta function $\delta_i^j = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$. We call $\{b_j(x), j = 1,2,\cdots,n\}$ the unit spline functions.

Without loss of generality, we only displace the interior points $(x_i, y_i)\ (i = 2,3,\cdots,n-1)$, suppose they are perturbed by some minor offsets $\gamma_i (i = 2,3,\cdots,n-1)$ individually. Thus the new spline function $\tilde{y}(x)$ defined by the points $\{(x_1, y_1), (x_2, y_2 + \gamma_2),\cdots,(x_{n-1}, y_{n-1} + \gamma_{n-1}), (x_n, y_n)\}$ can be represented as follows:

$$\tilde{y}(x) = y(x) + \sum_{j=2}^{n-1} \gamma_j b_j(x) \tag{12}$$

Computing the second derivative of the above equation we have

$$\tilde{c}_i = c_i + \sum_{j=2}^{n-1} \gamma_j \tilde{c}_i^j \tag{13}$$

for *i*=1, 2,⋯, *n*. where the coefficients $\tilde{c}_i^j$ can be computed as solving a Three-Moment Equations system as follows

$$\tilde{c}_j^j = \frac{-\left(\dfrac{1 + a_{j-1}}{x_j - x_{j-1}} + \dfrac{1 + a_{j+1}^*}{x_{j+1} - x_j}\right)}{2\left(x_{j+1} - x_{j-1}\right) - a_{j-1}\left(x_j - x_{j-1}\right) - a_{j+1}^*\left(x_{j+1} - x_j\right)} \tag{14}$$

for *j*=2, 3,⋯, *n*–1.

$$\begin{aligned} \tilde{c}_{j-1}^j &= a_{j-1}\left[\frac{1}{(x_j - x_{j-1})^2} - \tilde{c}_j^j\right] \\ \tilde{c}_{j+1}^j &= a_{j+1}^*\left[\frac{1}{(x_{j+1} - x_j)^2} - \tilde{c}_j^j\right] \end{aligned} \tag{15}$$

$$\tilde{c}_i^j = -a_i \tilde{c}_{i+1}^j,\ \ i = j-2,\cdots,1 \tag{16}$$

$$\tilde{c}_i^j = -a_i^* \tilde{c}_{i-1}^j,\ \ i = j+2,\cdots,n \tag{17}$$

Eq. (13) shows how the curvatures at the data points change according to their minor displacements. For any specific *j*, from $a_1 = a_n^* = -1$, Eq.(5) and (7), we know $0 < a_i < \frac{1}{2}, i = 2,3,\cdots,n,$ and $0 < a_i^* < \frac{1}{2}$, $i = n-1, n-2,\cdots,1$.

It can be seen that the coefficients $\tilde{c}_i^j$ have the following properties:

(1) Signs: for any specific *j*, $\tilde{c}_j^j < 0$ and $\tilde{c}_{j\pm1}^j > 0$, $\tilde{c}_{j\pm2}^j < 0$, $\tilde{c}_{j\pm3}^j < 0$, ... That is, the coefficients $\tilde{c}_2^j$, $\tilde{c}_3^j,\cdots,\tilde{c}_{n-1}^j$ are positive and negative alternatively.

(2) Values: $\tilde{c}_{j-1}^j, \tilde{c}_j^j, \tilde{c}_{j+1}^j$ have the largest absolute values while the others $\tilde{c}_i^j (i \neq j-1, j, j+1)$ decrease according to the exponential scale $2^{-|i-j|}$. In addition to, $\tilde{c}_1^j = \tilde{c}_2^j$, $\tilde{c}_{n-1}^j = \tilde{c}_n^j$.

From the above properties and Eq. (13) we can obtain the following observations.

Observation 1. Changes of c_i according to displacements of data points

Assuming that we move one point (x_i, y_i) by a small offset to $(x_i, y_i + \gamma_i)$ while keeping the other points fixed. If $\gamma_i > 0$, then $c_i \downarrow, c_{i\pm1} \uparrow, c_{i\pm2} \downarrow, \cdots$ ($\downarrow$ means decrease while $\uparrow$ means increase), and c_n and c_1 have the same increase or decrease changes with c_{n-1} and c_2 respectively. Except the three values c_{i-1}, c_i, c_{i+1}, the others $c_j (j \neq i-1, i, i+1)$ change slightly.

Similarly, we study the change of e_i according to the points' displacements as follows

$$\tilde{e}_i = e_i + \sum_{j=2}^{n-1} \gamma_j \tilde{e}_i^j,\ \ i = 1,2,\cdots,n-1 \tag{18}$$

where $\tilde{e}_i = \dfrac{\tilde{c}_{i+1} - \tilde{c}_i}{x_{i+1} - x_i} - \dfrac{\tilde{c}_i - \tilde{c}_{i-1}}{x_i - x_{i-1}}$.

From $e_i = \dfrac{c_{i+1} - c_i}{x_{i+1} - x_i} - \dfrac{c_i - c_{i-1}}{x_i - x_{i-1}}$, we can obtain the much similar properties of e_i as c_i as follows.

Observation 2. Changes of e_i according to displacements of data points.

Assuming that we move one point (x_i, y_i) by a small offset to $(x_i, y_i + \gamma_i)$ while keeping the other points fixed. If $\gamma_i > 0$, then $e_i \uparrow, e_{i\pm1} \downarrow, e_{i\pm2} \uparrow, \cdots$ ($\downarrow$ means decrease while $\uparrow$ means increase). Except the three values e_{i-1}, e_i, e_{i+1}, the others e_j ($j \neq i-1, i, i+1$) change slightly.

These two observations tell us that displacements of data points affect the curvature locally. And, c_i can be changed by changing y_i in the opposite direction and e_i can be changed by changing y_i in the same direction.

4 Our fairing algorithm

In this section, we propose our smoothing algorithm based on the polyline fairness indicator.

At the beginning, we give a two-phase method to fair a function segment in subsection 4.1 and the algorithm to smooth arbitrary curves is presented in subsection 4.2.

4.1 Fairing a function curve

Similar to previous local curve fairing method, our fairing approach is also trying to identify the "bad" points (unnecessary zero-value points and vibrations of the curvature) and eliminate them. For a function segment, it is performed in the following two phases.

Phase 1. Coarse fairing - eliminating unnecessary curvature vibrations.

This phase is going to reduce the redundant vibrations of curvature. Vibrations are considered as unnecessary if they appear frequently, i.e. two successive vibrations are too close. We consider the vibrations in three cases:

Case 1. (Small curvature vibration). When $e_ie_{i+1}<0$ happens, a small curvature vibration is located (see the left figure of Fig 2(a)). This vibration is eliminated via decreasing $|e_ie_{i+1}|$ which can be implemented by changing c_i and c_{i+1} along opposite directions (e.g., decreasing c_i and increasing c_{i+1} as shown in the middle of Fig 2(a)). It can be performed by offsetting y_i and y_{i+1} along opposite directions. The method in this case goes as follows:

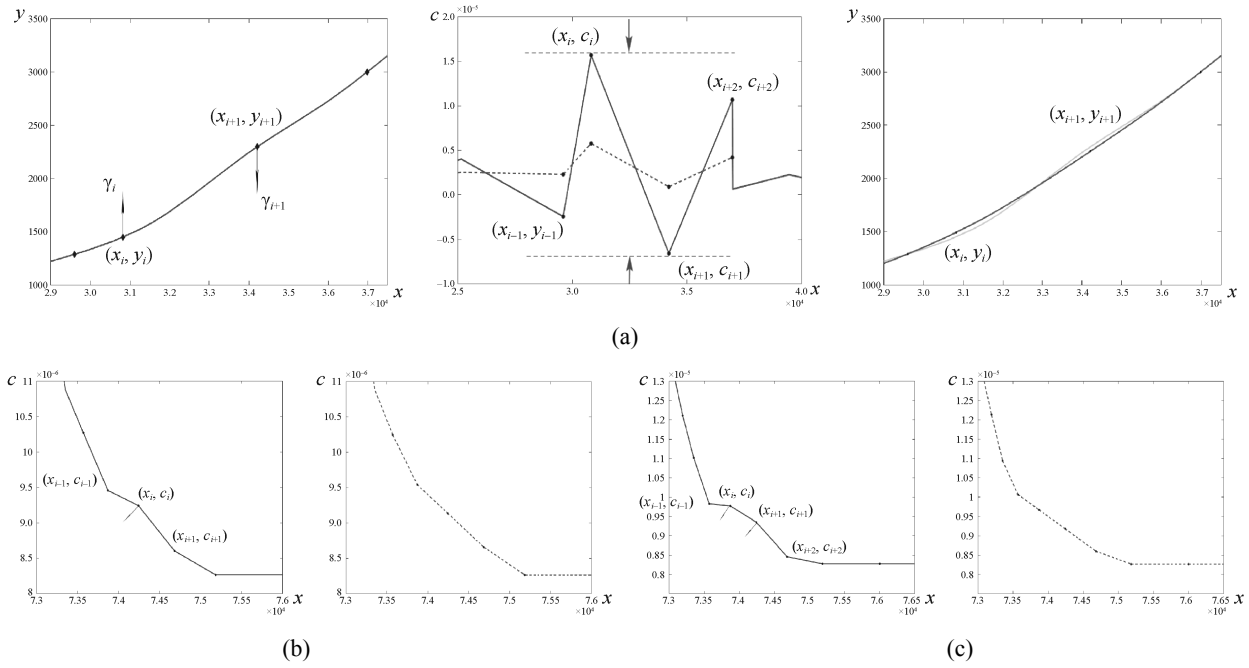


Fig. 2 The illustration of eliminating small curvature vibration (a), medium curvature vibration (b) and large curvature vibration (c) during coarse fairing phase. The *x*–*c* plot of the original curve and faired curve are shown in solid line and dashed line respectively (see the middle one of (a), (b) and (c)). The left figure of (a) shows the original curve and the right one shows the modified curve (in solid line) compared with the original curve (in dashed line).

Step 1. Loop i from 2 to n–1, when $e_ie_{i+1}<0$, we assume $\gamma_i=\sigma$ and $\gamma_{i+1}=-\sigma$ and solve two σ from equation $\tilde{e}_i\tilde{e}_{i+1}=0$. Then let σ be the one with minimum absolute value.

Step 2. Displace y_i and y_{i+1} by offsets $\gamma_i=\lambda\sigma$ and $\gamma_{i+1}=-\lambda\sigma$ respectively. We set $\lambda=0.8$ in practical.

Case 2. (Medium curvature vibration). If e_i has different sign with its neighbors, that is, $e_ie_{i-1}<0$ and $e_ie_{i+1}<0$ (Fig. 2(b)), we call it a medium curvature vibration. In this case, we aim to make e_i be a zero value. The steps are as follows.

Step 1. Loop i from 3 to n–2, when $e_ie_{i-1}<0$ and $e_ie_{i+1}<0$ go to the next step.

Step 2. Assume $\gamma_i=\sigma$. The parameter σ is determined by solving $\tilde{e}_i=0$.

Step 3. Displace y_i by offset γ_i.

We say the end point i=2 has a medium curvature vibration if $e_2e_3<0$ and $e_2e_4<0$. Then we make e_2 to be zero. The small curvature vibration at the other end $i=n-1$ is similarly handled.

Case 3. (Large curvature vibration) If e_i and e_{i+1} own the same sign, and have different sign with their neighbors, that is, $e_ie_{i-1}<0$, $e_ie_{i+2}<0$, $e_ie_{i-2}<0$ and $e_ie_{i+3}<0$ (Fig. 2(c)), we call it a large curvature vibration. The steps for this case are shown as:

Step 1. Loop i from 3 to $n-3$, when $e_i e_{i+1} > 0$, $e_i e_{i-1} < 0$, $e_i e_{i+2} < 0$, $e_i e_{i-2} < 0$ and $e_i e_{i+3} < 0$, go to next step.

Step 2. Assume $\gamma_i = \sigma$ and $\gamma_{i+1} = \tau$. The parameter σ and τ are determined by solving $\tilde{e}_i = \tilde{e}_{i+1} = 0$ (a linear system with two variables).

Step 3. Displace y_i and y_{i+1} by offsets γ_i and γ_{i+1} respectively.

We have to deal with the large curvature vibrations at the end points as well. Here we only consider the case of i=2 (i = n–1 is similar), the steps are:

Step 1. If e_2 has same sign with e_3 and different sign with e_4 and e_5, i.e., $e_2e_3>0$, $e_2e_4>0$ and $e_2e_5>0$, goto the next step. Otherwise, exit.

Step 2. Assume $\gamma_2 = \sigma$ and $\gamma_3 = \tau$. We obtain σ and τ by solving $\tilde{e}_2 = \tilde{e}_3 = 0$ (a linear system with two variables).

Step 3. Displace y_2 and y_3 by offsets γ_2 and γ_3 respectively.

Phase 2. Fine fairing – reducing unnecessary inflections.

The fine fairing phase aims to eliminate redundant inflection points. Similar to cur-vature vibrations, we said inflection points are redundant if they appear in a high frequency (i.e. two successive inflection points close to each other). It is treated in the following two cases.

Case 1. (Small curve vibration). If c_i has different sign with its neighbors, that is, $c_ic_{i-1}<0$ and $c_ic_{i+1}<0$ (Fig. 3 (a)), we call it a *small curve vibration*. In this case, we aim to adjust c_i to make it has the same sign with its neighbors. The details are shown in the following:

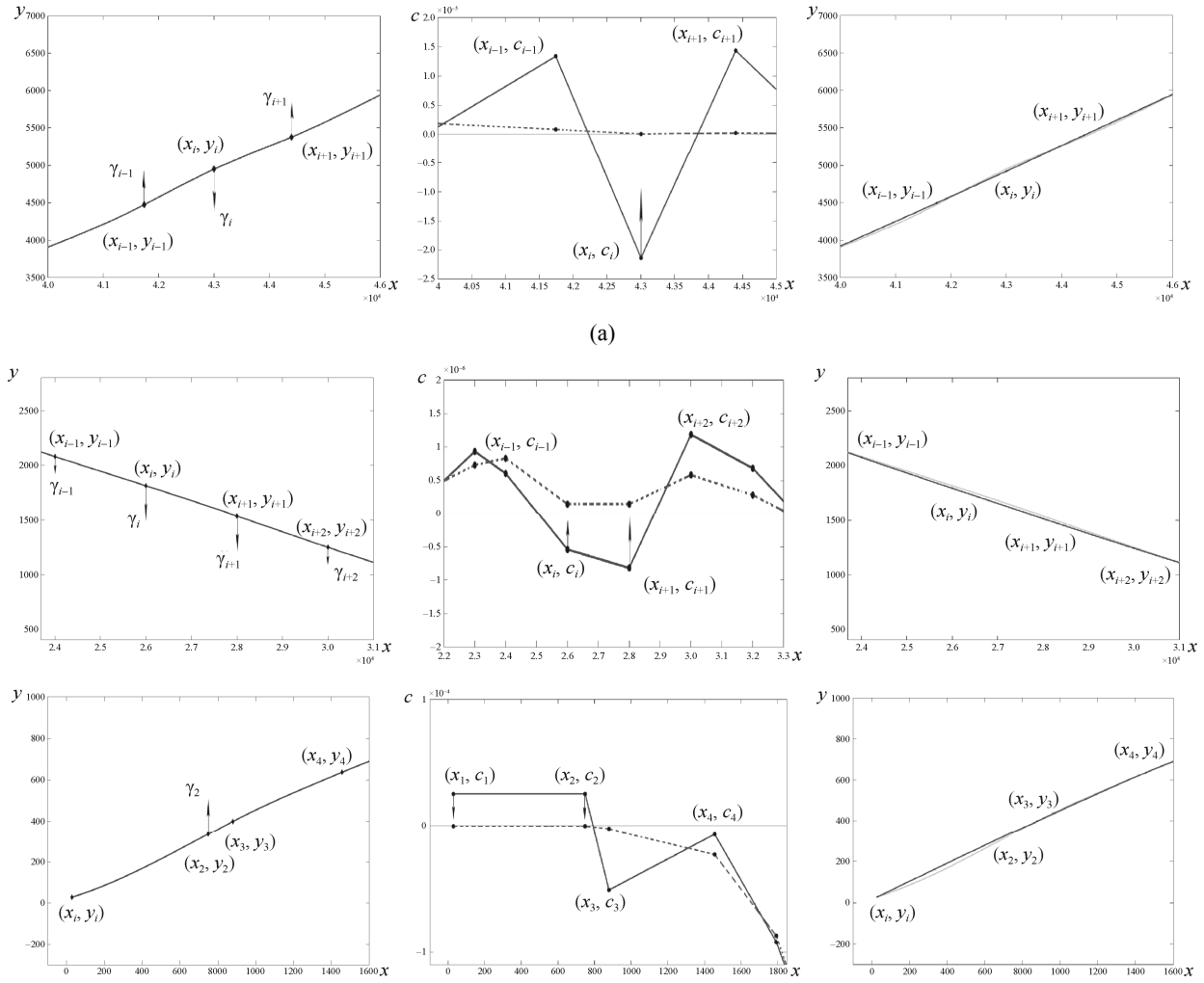


Fig. 3 Fine fairing phase. (a) Small curve vibration. (b) Medium curve vibration. Upper: interior cases. Lower: boundary case i=1. Left: original curve. Middle: original x–c plots (in solid line) and revised ones (in dashed line). Right: faired curves (solid line) and original curves (dashed line).

Step 1. Loop i from 2 to n–1, when $c_ic_{i-1}<0$ and $c_ic_{i+1}<0$, we assume $\gamma_i=\sigma$ and $\gamma_{i-1}=\gamma_{i+1}=-\lambda\sigma$. We obtain σ by solving $\tilde{c}_i=\alpha$. Here we set $\alpha=0$ firstly.

Step 2. Displace y_{i-1},y_i,y_{i+1} by offsets $\gamma_{i-1},\gamma_i,\gamma_{i+1}$ respectively. We set $\lambda=0.75$ in practical.

Step 3. If both $\tilde{c}_{i-1}$ and $\tilde{c}_{i+1}$ do not change their signs after Step 2, we set $\alpha=\frac{1}{5}\min(\tilde{c}_{i-1},\tilde{c}_{i+1})$ and do Step 1 and Step 2 again.

Case 2. (Large curve vibration). If there are two successive points c_i and c_{i+1} whose signs are same but different from their neighbors', that is, $c_ic_{i+1}>0$, $c_ic_{i-1}<0$, $c_ic_{i+2}<0$ and $c_ic_{i+3}<0$ (Fig 3 (b)), we call it a *medium curve vibration*. To eliminate this vibration, we adjust c_i and c_{i+1} to make them have same signs with their neighbors. The fairing steps are as follows:

Step 1. Loop i from 2 to n–3, when $c_ic_{i+1}>0$, $c_ic_{i-1}<0$, $c_ic_{i+2}<0$ and $c_ic_{i+3}<0$, go to the next step.

Step 2. Assume $\gamma_i=\gamma_{i+1}=\sigma$ and $\gamma_{i-1}=\gamma_{i+2}=\tau$. The parameter σ and τ are deter-mined by solving $\tilde{c}_i=\tilde{c}_{i+1}=\alpha$ (a linear system with two variables). Here we set α=0 firstly.

Step 3. Displace $y_{i-1},y_i,y_{i+1},y_{i+2}$ by offsets γ_{i-1}, $\gamma_i,\gamma_{i+1},\gamma_{i+2}$ respectively.

Step 4. If both $\tilde{c}_{i-1}$ and $\tilde{c}_{i+2}$ do not change their signs after Step 2, we set $\alpha=\frac{1}{5}\min(\tilde{c}_{i-1},\tilde{c}_{i+2})$ and do Step 2 and Step 3 again.

We have to deal with the large curve vibration at the end points as well (see the bottom three figures in Fig. 3 (b)). We show the steps when i=1 as (the case $i=n$ is similar):

Step 1. If c_1=c_2 have different signs with c_3 and c_4, i.e., c_1c_3<0 and c_1c_4<0, go to next step. Otherwise, exit.

Step 2. Assume $\gamma_2=\sigma$ which can be obtained by solving the equation $\tilde{c}_2=0$.

Step 3. Displace y_2 by offset γ_2.

In our approach, the two primary phases are performed in an alternative way iteratively. Notice that, there might exist other redundant curvature vibrations or redundant inflections other than the cases listed as the above. For example, if k+1 successive points c_i to c_{i+k} own same signs but different signs from their neighbors, a redundant inflection might exist when k is a small number. Our method does not deal with these cases especially since we find that they can be avoided very well by our alternative two-phases fairing. Fig. 4 shows the performance of our two-phases fairing approach, which generate a much fairer curve and visual pleasing x–c plot.

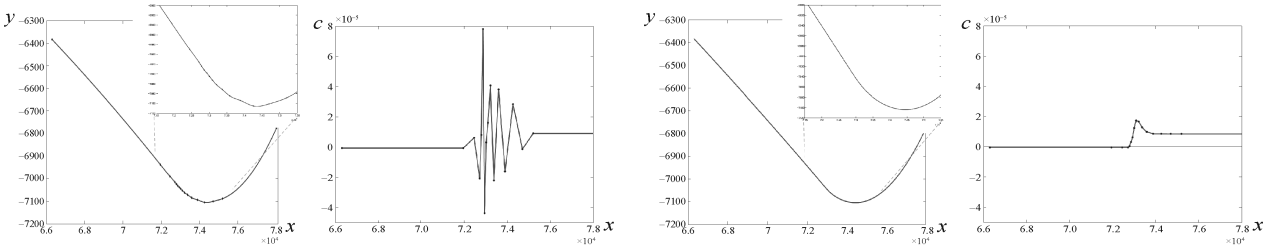


Fig. 4 An example of our coarse-to-fine fairing algorithm. (a) Original curve with its x–c plot. (b) Final faired result with its x–c plot.

4.2 Fairing arbitrary curves

We show our fairing pipeline for arbitrary curves in this subsection. Input a sequence of data points $\{P_i\}(i=1,2,\cdots,n)$. Our algorithm has two steps. Firstly, the points are partitioned into several overlapped function segments, and then all segments are faire-d one by one iteratively.

4.2.1 Data points partitioning

Denote α_i as the oriented angle from P_1P_2 to P_iP_{i+1}(i=1,2,⋯,n–1) $\left(-\frac{\pi}{2}<\alpha_i<\frac{\pi}{2}\right)$. The data points $\{P_i\}$ can not be represented by a function (with a coordinate frame transformation) when the biggest turning angle is bigger than π, that is $\max\{\alpha_i\}$-$\min\{\alpha_i\}$>π. Therefore, our partitioning is firstly aiming to divide the input points se-quence into multiple segments whose biggest turning angles are smaller than π. Then, each segment is transformed into a new coordinate system which makes function representation successfully.

4.2.2 Partitioning

We check the biggest turning angle $\max_{1\leqslant i\leqslant s}\{\alpha_i\}-\min_{1\leqslant i\leqslant s}\{\alpha_i\}$ for each s from 1 to n to find the first s

which makes it be bigger than π. And the points sequence from P_1 to P_{s-1} is considered as the first segment. The remaining data points are then partitioned in a similar way. To avoid discontinuity between two successive segments, we allow each adjacent segments to own k points overlapped (5 is preferred for k in our experiments).

4.2.3 Coordinate transforming

In order to represent each segment by a function, we also need to do a coordinate system transformation. Take the first segment P_1 to P_f as an example, we denote u=argmax$_{1\leqslant i\leqslant f}\{\alpha_i\}$, l=argmax$_{1\leqslant i\leqslant f}\{\alpha_i\}$. Therefore, the angular bisector of the angle from vector P_uP_{u+1} to P_lP_{l+1} can be viewed as the x-axis of the new coordinate system and all of points can be transformed directly.

4.2.4 Iteratively fairing

After partitioning and coordinate transforming are done, two-phases fairing method is then applied on all of the segments one by one. And, the overlapped part is generat-ed by blending its two adjacent faired segments together. This pipeline is also carried out several times iteratively.

Fig. 5 uses *strophoid* curve to show our whole fairing procedure. It has the formula as $\left\{x=\frac{10(t^3-t)}{t^2+1}, y=\frac{10(t^2-1)}{t^2+1}; -2\leqslant t\leqslant 2\right\}$. We sample 65 equally spaced points on the curve and perturb 40 internal nodes with gaussian noise The transformed curves and x-c plots before and after two-phases fairing of the first two successive segments are shown in Fig 5 (d) and (e). Furthermore, Fig 5 (b) and (c) show the faired results with and without overlapping, from which we can see the necessity of overlapping and blending.

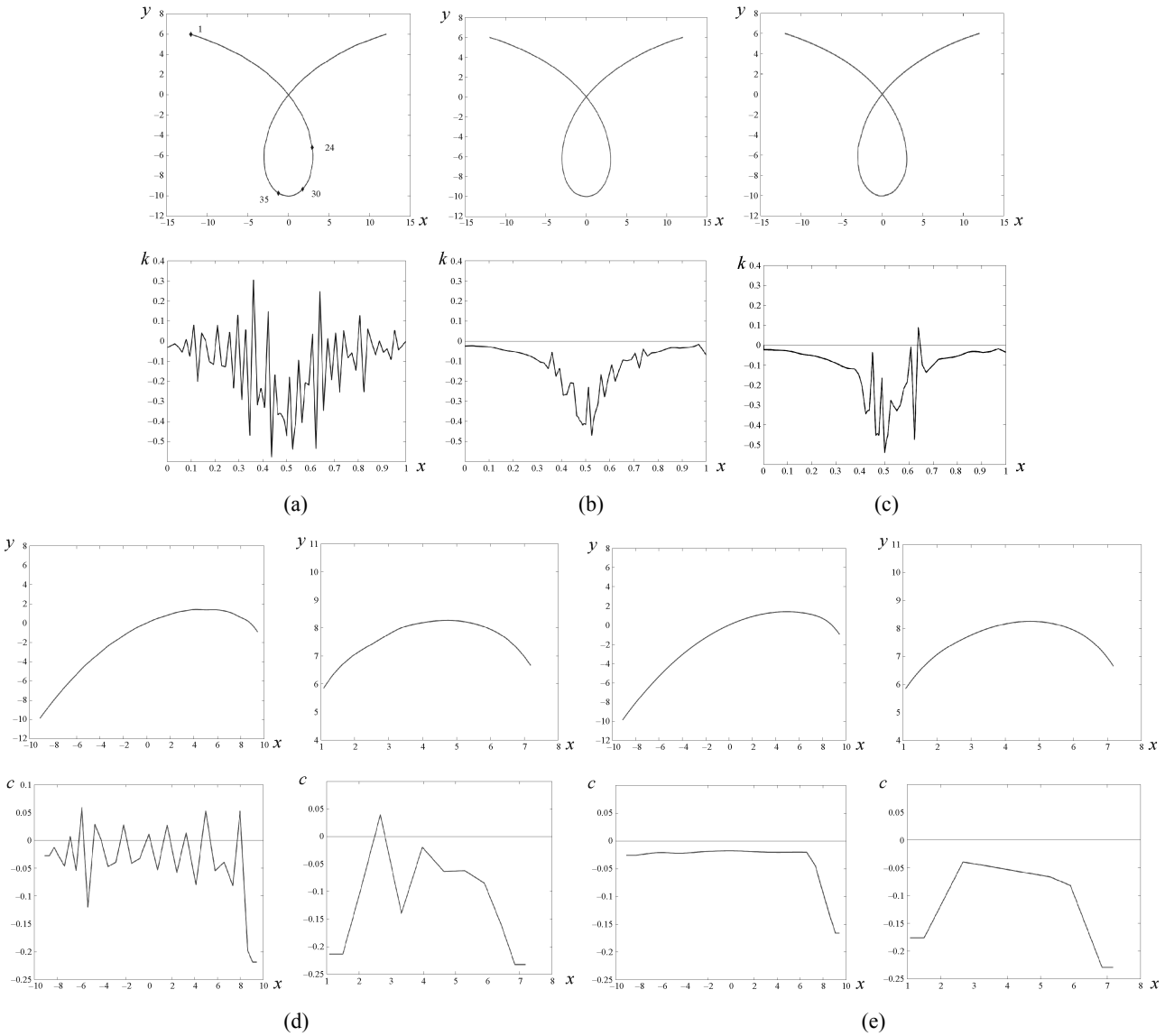


Fig. 5 An example of curve fairing. Upper row: *x*–*y* plots. Lower row: *x*–*k* plots or *x*–*c* plots. (a) Original curve. (b) We faired several segments separately without blending. (c) Our fairing results with the blending operation. (d) The first two successive segments and their *x*–*c* plots. (e) the faired results and *x*–*c* plots of (d).

5 Experimental results

In this section, we compare our method with previous local and global algorithms by several practical curves and synthetic curves. Sapidis-Farin (SF) method [15] is selected as the local one on the ground of its typicalness. For the global method, we use the smoothing function in Matlab Spline Toolbox which we called MST method. It smooths the curve by minimizing the energy $E = \lambda E_d + (1-\lambda)E_s$, where $E_d = \sum D^2(i)$ is the data term which reflects closeness of the result curve to original dada (D is the displacement vector of points) and E_s expresses smoothness of the faired curve. Here, λ is a weight to balance the data term and the smoothness term. Note that, for each example in our experiments, three different values of λ are careful cho-se which give rise to similar results as our method (those values are very large since E_d and E_s have very different magnitude).

In our comparisons, the original curves and their corresponding faired results accompanying with the curvature plots using SF method, MST method and our method are shown side by side. For some curves who own only one function segment, we show their *x–c* plot instead of the curvature plot (example 1 and example 2). And, we also show the number of their inflection points $Infl_{num}$ and the number of curvature vibrations Vib_{num} to validate the efficiency of our algorithm. They can be defined by the x-c plot as $Infl_{num} = num\{i \mid c_i c_{i+1} < 0\}$ and $Vib_{num} = num\{i \mid e_i e_{i+1} < 0\}$. To quantify the displacements of data points, we use two norms of the displacement vector $D: S_d = \| D \|_1 = sum(D)$ and $M_d = \| D \|_\infty = \max(D)$. And we also show numerical results of S_d/M_d and the smoothness energy E_s. For the global fairing method MST, we use three different smoothness weight λ where the middle one has similar displacements with our method.

Example 1. Station line of a ship hull.

The first example is a station line segment of a ship hull (SL curve). It is not needed to be partitioned. Seen from 5, the original curve has several bumps which makes it look unfair. Especially, it has 4 inflections and 19 vibrations on its curvature. All of the three methods eliminate redundant inflections and curvature vibrations, and reduce the smoothness energy a lot (Fig. 5 and Table 1). From the result curve and numerical results in Table 1, our algorithm eliminates the most unnecessary inflections and unnecessary curvature vibrations although its E_s is not lower than MST method.

Table 1 Numerical comparisons for fairing curve SL in Fig. 6.

Method	$Infl_{num}$	Vib_{num}	S_d/M_d	$E_s(e+9)$
Original	4	19		1.50
Our	2	5	141.9/12.6	1.25
MST (λ = 0.99995)	2	9	210.8/25.5	1.19
MST (λ= 0.99997)	2	7	149.5/18.7	1.22
MST (λ= 0.99999)	2	9	69.8/10.0	1.27
SF	2	13	355.1/108.0	1.32

Table 2 Numerical comparisons for fairing the plane section curve in Fig. 7.

	$Infl_{num}$	Vib_{num}	S_d/M_d	E_s
Original	18	19		9759.20
Our	5	7	0.25/0.026	1992.30
MST (λ= 0.99997)	6	9	0.44/0.053	922.19
MST (λ= 0.99999)	8	9	0.27/0.034	1386.60
MST (λ= 0.999993)	8	14	0.22/0.029	1552.10
SF	10	13	0.33/0.05	3812.30

Example 2. Plane body section.

The input of this example is a series of data points sampled from a plane section (Fig. 7). It is also not needed to be partitioned. Although its curvature vibrates too much, our method still performs efficiently. Compared with other algorithms, it also outputs the smallest $Infl_{num}$ and Vib_{num} (Table 2) and the best x-c plot (fig. 7).

Example 3. Turbine blade.

The data points of this example are originated from the turbine blade. It needs to be partitioned using our method (Fig. 8). We show the curvature plot x-k directly. Table 3 shows S_d/M_d and E_s of the three algorithms we used. From fig 8 (e) and (f), we can find an obvious bump at the lower part of the curve near the end point though it has lower energy. Although Fig 8 (d) generates the best curvature plot, the displacements of data points are too large and the faired curve is too far from the original input data.

More examples

More results are shown in Fig. 9. The left column shows an arc with random noise, the middle one shows a Buttock line of the ship hull and the right column shows an example sampled from a shoe profile. To visualize the fairness of the curve, we

use curvature bar instead of curvature plot, whose direction is perpendicular to the curve and its length reflects the value of curvature at that point. And, we only show one *MST* result here which has the similar displacements of input data points with our method.

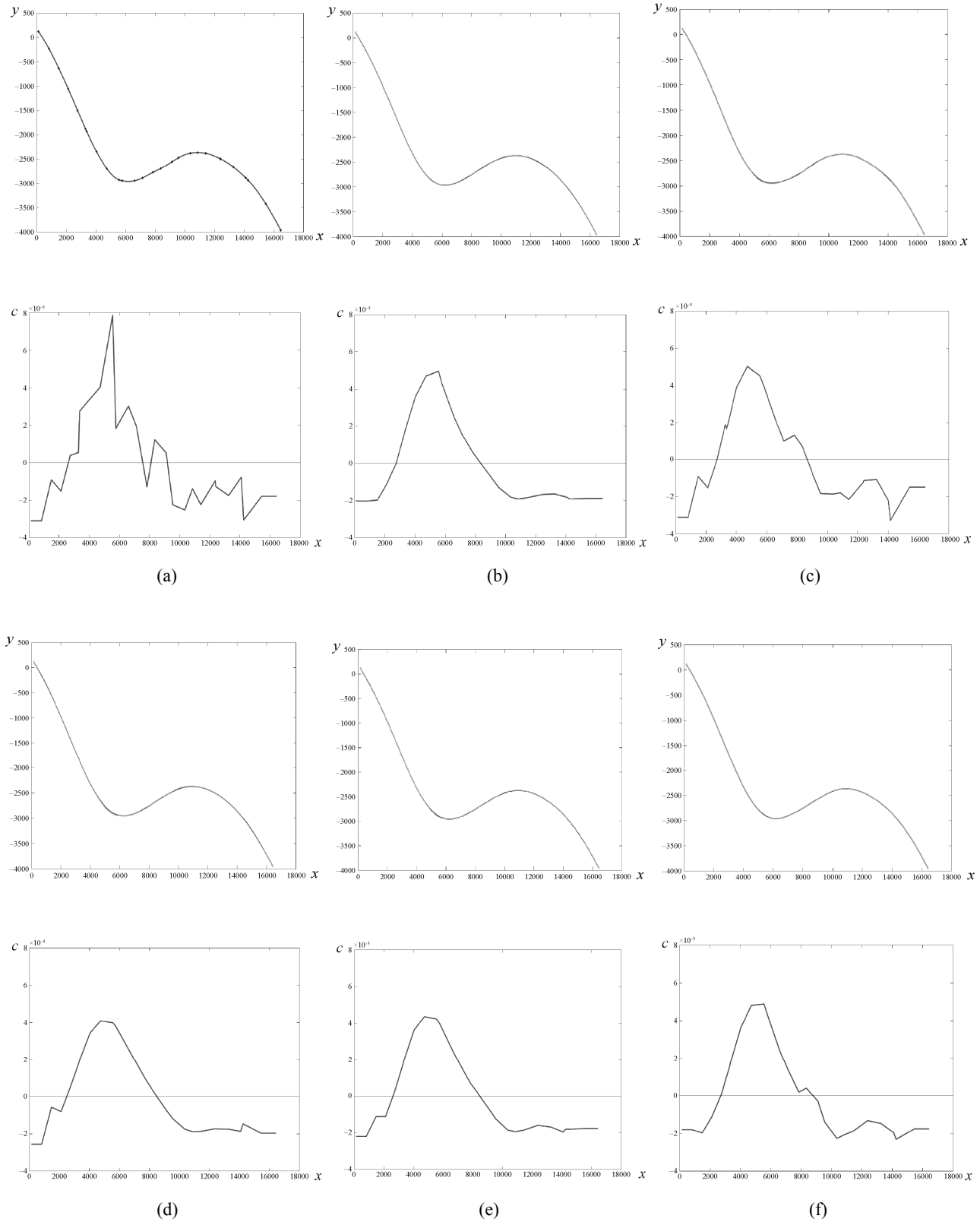


Fig. 6 Comparisons with SF method and MST method on a Station Line segment from a ship hull. (a) Original curve and x–c plot, red is the data points. (b) Faired result of our method. (c) The result of SF method. (d)-(f) The results of MST method with 3 different λ(0.99995,0.99997,0.99999).

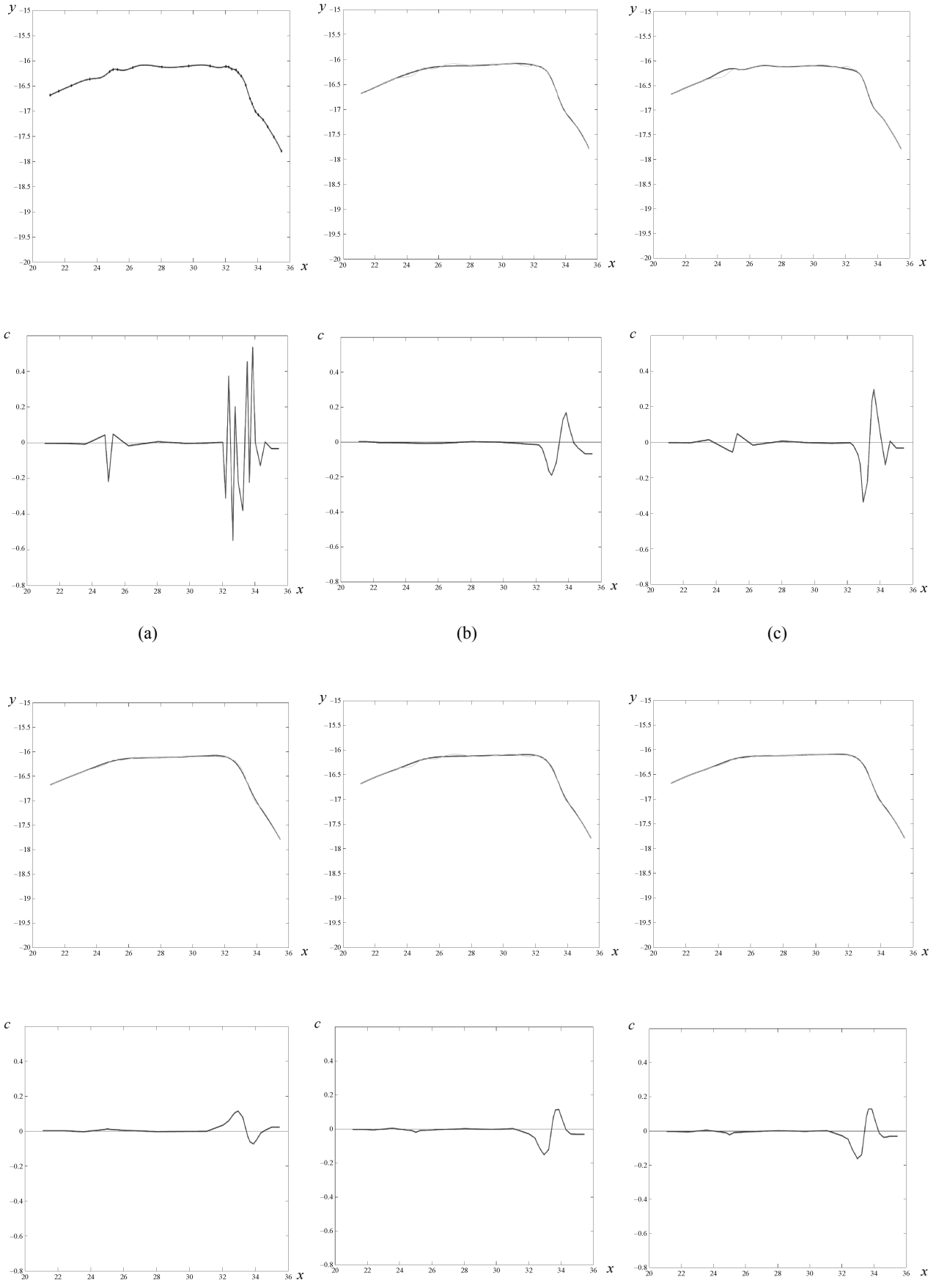


Fig. 7 Comparisons with SF method and MST method on a plane section curve. (a) Original. (b) Ours. (c) SF. (d) MST (λ = 0.99997). (e) MST (λ = 0.99999). (f) MST (λ = 0.999993).

Table 3 Numerical comparisons for fairing Turbine blade in Fig. 8.

	Original	Our	MST (λ = 0.99999)	MST (λ = 0.999999)	MST (λ = 0.9999999)	SF
S_d/M_d		0.94/0.0988	2.43/0.1099	0.81/0.039	0.28/0.014	0.12/0.026
E_s	4.0e+5	1.7e+5	1.3e+5	1.46e+5	1.68e+5	2.37e+5

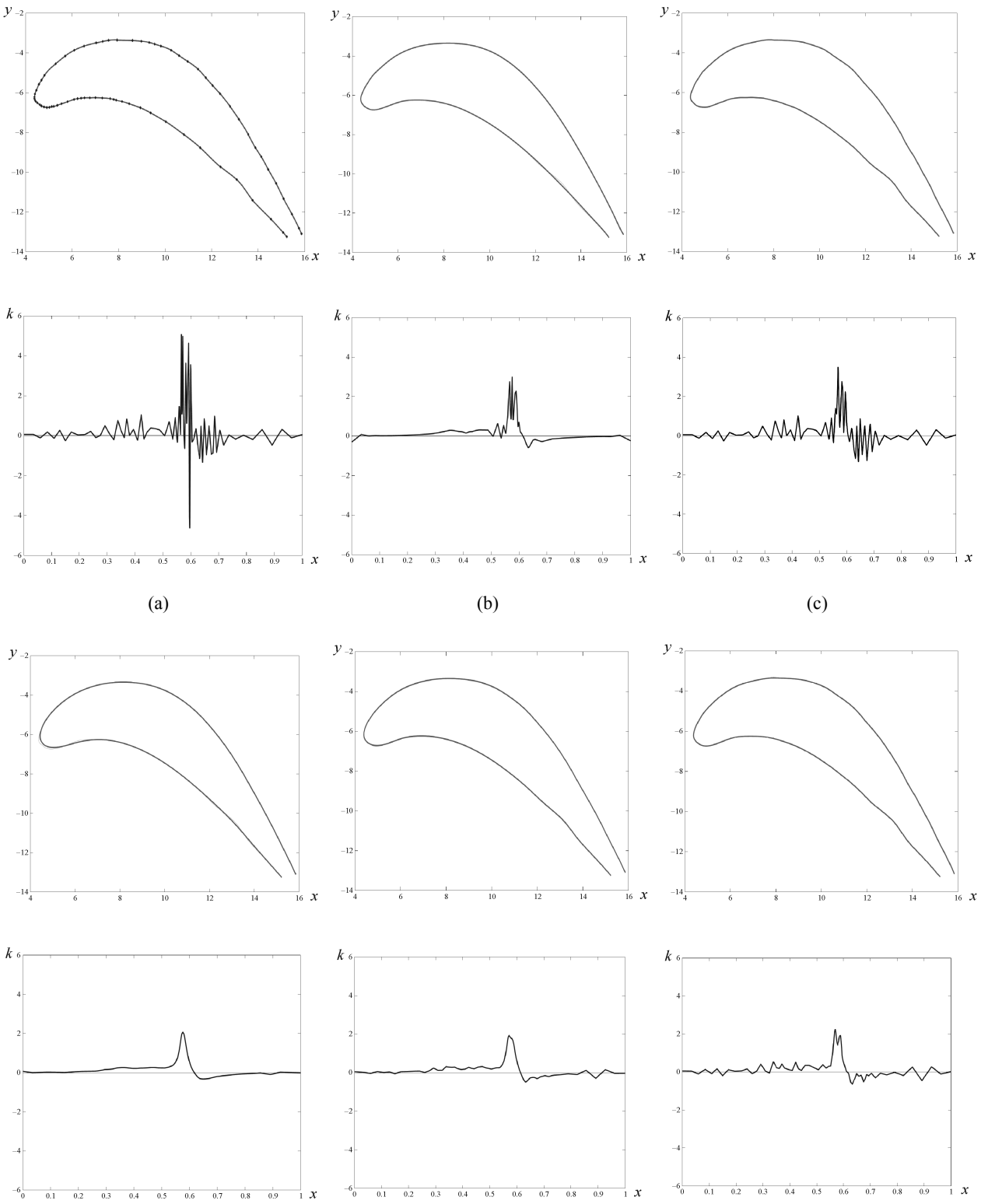


Fig. 8 Comparisons with SF method and MST method on a Turbine blade. (a) Original. (b) Ours. (c) SF. (d) MST (λ = 0.99999). (e) MST (λ = 0.999999). (f) MST (λ = 0.9999999).

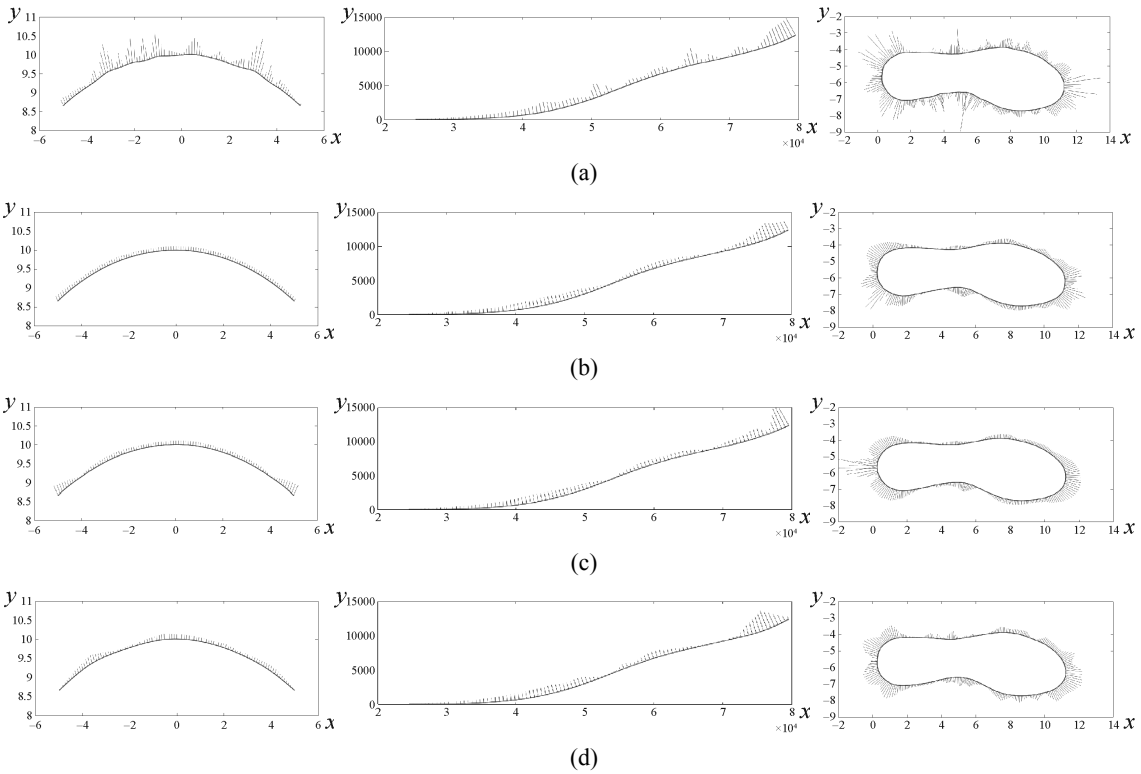


Fig. 9 More results shown by curvature bar. Left column: arc with random noise. Middle column: buttock line of ship hull. Right column: shoe sole. (a) Original. (b) Ours. (c) MST. (d) SF.

6 Conclusion

We have presented a novel fairing method for planar curves in this paper, which is the first attempt to involve cubic spline function for curve fairing. We split the input data points into several parts, and use cubic spline function to represent each one. For each segment, the curvature is approximated by a polyline and an extrinsic relationship between this polyline and displacements of data points is also derived. Take this polyline as the fairness indicator, a two-phases fairing approach has been developed to identify and eliminate unnecessary inflection points and curvature vibrations efficiently. The proposed method in this paper generates the best results which is validated by numerous practical and synthetic examples.

References

[1] Kjellander J. Smoothing of cubic parametric splines [J]. Computer-Aided Design, 1983, 15(3): 175-179.

[2] Farin G, Rein G, Sapidis N, et al. Faring cubic b-spline curves [J]. Computer Aided Geometric Design, 1987, 4(1-2): 91-103.

[3] Poliakoff J. An improved algorithm for automatic fairing of non-uniform parametric cubic splines [J]. Computer-Aided Design, 1996, 28(1): 59-66.

[4] Poliakoff J, Wong Y, Thomas P. An automated curve fairing algorithm for cubic b-spline curves [J]. Journal of Computational and Applied Mathematics, 1999, 102: 73-85.

[5] Pigounakis K, Sapidis N, Kaklis P. Fairing spatial b-spline curves [J]. Journal of Ship Research, 1996, 40(4): 351-367.

[6] Zhang C M, Zhang P F, Cheng F H. Fairing splines and surfaces by minimizing energy [J]. Computer-Aided Design, 2001, 33(13): 913-923.

[7] Nowacki H, Liu D, Lv X. Fairing bezier curves with constraints [J]. Computer Aided Geometric Design, 1990, 7(1-4): 43-55.

[8] Eck M, Hadenfeld J. Local energy fairing of b-spline curves [J]. Computing (Supplement), 1995,10(2): 129-147.

[9] Applegarth I, Kaklis P, Wahl S. Benchmark tests on the generation of fair shapes subject to constraints: based on the benchmarking experiment organized by the HCM-Network FAIRSHAPE ERB-CHRXCT-940522, Spring-summer 1997 [M]. Teubner, 2000: 23-32.

[10] Moreton H P, Skquin C H. Functional optimization for fair surface design [J]. Computer Graphics, 1992, 26(2): 167-176.

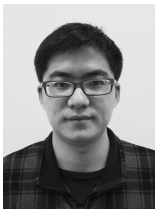
[11] Nowacki H, Lv X. Fairing composite polynomial curves with constraints [J]. Computer Aided Geometric Design, 1994, 11(1): 1-15.

[12] Wang Y, Zhao B, Zhang L, et al. Designing fair curves using monotone curvature pieces [J]. Computer Aided Geometric Design, 2004, 21(5): 515-527.

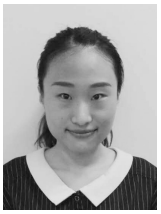
[13] Calio F, Miglio E, Rasella E. Curve fairing using integral spline operators [J]. International Journal for Numerical Methods in Biomedical Engineering, 2010, 26(26): 1674-1686.

[14] Amati G. A multi-level filtering approach for fairing planar cubic bspline curves [J]. Computer Aided Geometric Design, 2009, 24(1): 53-66.

[15] Sapidis N, Farin G. Automatic fairing algorithm for b-spline curves [J]. Compuer-Aided Design, 1990, 22(2): 121-129.
[16] Su B Q, Liu D. Computational geometry: curve and surface modeling [M]. San Diego: Academic Press Professional, Inc., 1989: 103-109.
[17] Farin G. Curves and surfaces for cagd: a practical guide [M]. 5th ed. San Francisco: Morgan Kaufmann Publishers Inc., 2002: 191-203.



Han Xiaoguang is currently a Ph.D. student with the Department of Computer Science at the University of Hong Kong since 2013. He received his M.Sc. in Applied Mathematics (2011) from Zhejiang University, and his B.S. in Information and Computer Science (2009) from Nanjing University of Aeronautics and Astronautics, China. He was also a Research Associate of School of Creative Media at City University of Hong Kong during 2011 to 2013. His research interests include computer graphics and computer vision.



Zhang Qing is currently a full-time Teaching Assistant at the School of Digital Media, Shenzhen Institute of Information Technology. She received her M.FA. (2012) from City University of Hong Kong, and her B.A. (2010) from Communication University of China-Nanguang College. Her research interests include video post-editing and computer graphics.

Dong Guangchang was born in 1928 and was retired as a distinguished professor at the Department of Mathematics, Zhejiang University. He has published over 50 papers in the area of spline theory, PDE, and number theory. He invented the elastic local linear interpolation for spline fairing and had successfully applied it in ship lofting. He has published 4 books and his book 'Nonlinear second order partial differential equations' was published by American Mathematical Society in 1991.

Liu Ligang is a professor at the School of Mathematical Sciences, University of Science and Technology of China. He received his B.Sc. (1996) and his Ph.D. (2001) from Zhejiang University, China. Between 2001 and 2004, he worked at Microsoft Research Asia. Then he worked at Zhejiang University during 2004 and 2012. He paid an academic visit to Harvard University during 2009 and 2011. His research interests include digital geometric processing, computer graphics, and image processing. He serves as the associated editors for journals of Computer Aided Geometric Design and IEEE Computer Graphics and Applications. His research works could be found at his research website: http://staff.ustc.edu.cn/~lgliu.