Computers & Graphics 35 (2011) 967–975

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

An extended iterative format for the progressive-iteration approximation[☆]

Hongwei Lin[*], Zhiyu Zhang

State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310058, China

ARTICLE INFO

Article history:
Received 27 December 2010
Received in revised form
16 July 2011
Accepted 18 July 2011
Available online 28 July 2011

Keywords:
Progressive-iteration approximation
Data fitting
Progressive interpolation
Geometric design
Parallel computing

ABSTRACT

Progressive-iteration approximation (PIA) is a new data fitting technique developed recently for blending curves and surfaces. Taking the given data points as the initial control points, PIA constructs a series of fitting curves (surfaces) by adjusting the control points iteratively, while the limit curve (surface) interpolates the data points. More importantly, progressive-iteration approximation has the local property, that is, the limit curve (surface) can interpolate a subset of data points by just adjusting a part of corresponding control points, and remaining others unchanged. However, the current PIA format requires that the number of the control points equals that of the data points, thus making the PIA technique inappropriate to fitting large scale data points. To overcome this drawback, in this paper, we develop an *extended PIA* (EPIA) format, which allows that the number of the control points is less than that of the given data points. Moreover, since the main computations of EPIA are independent, they can be performed in parallel efficiently, with storage requirement $O(n)$, where n is the number of the control points. Therefore, due to its local property and parallel computing capability, the EPIA technique has great potential in large scale data fitting. Specifically, by the EPIA format, we develop an incremental data fitting algorithm in this paper. In addition, some examples are demonstrated in this paper, all implemented by the *parallel computing toolbox* of Matlab, and run on a PC with a four-core CPU.

1. Introduction

Data fitting is a fundamental tool in solving scientific and engineering problems found in the real world, which constructs a curve or patch, or a mathematical function, that has the best fit to a series of data points, possibly subject to constraints. A desirable data fitting method should be able to control the fitting precision for each data point individually. Thus, the data points can be fitted adaptively, that is, only the data points with dissatisfied precision need to be dealt with. In this manner, the computation resource can be saved greatly, especially in fitting large scale data points. However, traditional methods for data fitting usually solve a linear system, hence it is impossible to control the fitting precision for each data point individually.

Moreover, in fitting large scale data points, the incremental manner is often employed, which starts from an initial fitting curve. If the precision of the current fitting curve does not meet the requirement, the number of the control points of the fitting curve is increased, and the data points are fitted again, improving the fitting precision. The desirable manner for re-fitting the data points is able to take advantage of the last fitting result, thus saving the computational resource. However, when we re-fit the data points using the conventional method, such as the least square fitting technique, the last fitting result is totally discarded.

Recently, a new data fitting technique, *progressive-iteration approximation* (PIA) is presented. The PIA method is an iterative procedure, starting by an initial blending curve with control points and blending basis (it also works for tensor product surfaces, we omit the surface case for brevity here), and the limit curve interpolates the given data points. In each iteration, the main computation is to evaluate the foot points corresponding to the given data points. The point evaluation in each iteration is independent completely, so it is suitable to be calculated in parallel in its nature. Moreover, it has been shown that the PIA method has the local property, that is, it can control the fitting precision for each data point separately. In addition, while it requires $O(n^2)$ memory to solve the linear system for data fitting, where n is the number of unknowns, the PIA method decreases the storage requirement to $O(n)$.

However, in the classical PIA format, the number of the control points is equal to that of the data points. It is not feasible when the number of data points is very large. In this paper, we present an *extended PIA* (EPIA) format, where the number of the control points is less than that of the data points. Together with the local property and parallel computing capability of PIA, the extended

[☆] This article recommended for publication by Jun-Hai Yong.
[*] Corresponding author.
E-mail address: hwlin@cad.zju.edu.cn (H. Lin).

0097-8493/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2011.07.003

PIA is a desirable tool for fitting large scale data. By the EPIA format, we develop an incremental data fitting algorithm in this paper, which has the following advantages:

- it can control the fitting precision to each data point individually;
- in fitting data points incrementally, the new fitting procedure starts from the last fitting result.

This paper is organized as follows. In Section 1.1, we review the related work briefly. Section 2 overviews the classical PIA method presented previously. In Section 3, we present the extended PIA method, and show its convergence. Moreover, in Section 4, we develop the incremental data fitting algorithm by the EPIA format. Section 5 demonstrates some examples, all implemented by the *parallel computing toolbox* of Matlab. Finally, Section 6 concludes the paper.

1.1. Related work

The *progressive-iteration approximation* (PIA) is a new technique to seek the curve or patch fitting the data points. The PIA property of the uniform cubic B-spline curve, first discovered by Qi et. al. [1] and de Boor [2], respectively, generates a sequence of curves by adjusting the control points iteratively, and the limit curve interpolates the given data points. In Ref. [3], Lin et al. show that the nonuniform cubic B-spline curve and surface also hold the property. Furthermore, the PIA method is extended to the blending curve and surface with normalized totally positive basis [4]. In Ref. [5], the convergence rates of different bases are compared, and the basis with the fastest convergence rate is found. Moreover, it is proved that the rational B-spline curve and surface (NURBS) have the property, too [6]. Recently, Martin et al. [7] devise an iterative format for fitting, which is actually the progressive-iteration approximation (PIA) format for the uniform periodic cubic B-spline. Very recently, Lu [8] devises a weighted PIA format to speed up the convergence of the PIA method. More importantly, Lin [9] discovers the local property of the PIA, by which PIA can control the fitting precision of each data point individually.

While the PIA format depends on the parametric distance between the data points and the corresponding foot points on the curve with the same parameters, Maekawa et al. invent an iterative fitting format, called *interpolation by geometric algorithm* [10,11], which is similar to PIA format, but relies on geometric distance between the data points and their closest points on the curve. Lin [12] shows the convergence of the interpolation by geometric algorithm. Moreover, the geometric interpolation algorithm [10] is extended to approximate the vertices of a triangular mesh using Loop subdivision surface [13]. On the other hand, the squared distance minimization method [14,15] is also an iterative data fitting algorithm.

Furthermore, the PIA format has been extended to subdivision surface fitting, named *progressive interpolation* (PI). Cheng et al. design the PI format of subdivision fitting for Loop subdivision surface [16,17], and prove its convergence. Fan et al. develop the PI format of Doo–Sabin subdivision surface fitting [18]. The PI format for Catmull–Clark subdivision surface fitting is proposed in Ref. [19].

2. Overview of the classical progressive-iteration approximation

The classical PIA method begins with an initial blending curve or patch with normalized totally positive basis, and generates a sequence of curves or patches by adjusting the control points iteratively.

Specifically, given an ordered data point set

$\{\boldsymbol{P}_0,\boldsymbol{P}_1,\ldots,\boldsymbol{P}_n\}$,

each point is assigned a parameter $t_i, i=0,1,\ldots,n$, where $t_0<t_1<\cdots<t_n$. Taking these data points as the initial control points, that is, $\boldsymbol{P}_i^0=\boldsymbol{P}_i, i=0,1,\ldots,n$, the initial blending curve $\boldsymbol{P}^0(t)$ can be constructed as

$$\boldsymbol{P}^0(t)=\sum_{i=0}^{n}\boldsymbol{P}_i^0B_i(t), \tag{1}$$

where $\{B_i(t);\ i=0,1,\ldots,n\}$ is the normalized totally positive blending basis.

Suppose the kth curve $\boldsymbol{P}^k(t)$ has been generated. By computing the foot points $\boldsymbol{P}^k(t_i)$ on the kth curve $\boldsymbol{P}^k(t)$, constructing the *difference vectors* $\boldsymbol{\Delta}_i^k=\boldsymbol{P}_i-\boldsymbol{P}^k(t_i)$, and moving the control points along the corresponding difference vectors, that is, $\boldsymbol{P}_i^{k+1}=\boldsymbol{P}_i^k+\boldsymbol{\Delta}_i^k, i=0,1,\ldots,n$, we get the next curve $\boldsymbol{P}^{k+1}(t)$,

$$\boldsymbol{P}^{k+1}(t)=\sum_{i=0}^{n}\boldsymbol{P}_i^{k+1}B_i(t). \tag{2}$$

In this way, a sequence of curves, $\{\boldsymbol{P}^k(t);\ k=0,1,\ldots\}$, are generated. It has been proved in Refs. [3,4] that the limit curve of the sequence interpolates the data points, i.e., $\lim_{k\to\infty}\boldsymbol{P}^k(t_i)=\boldsymbol{P}_i, i=0,1,\ldots,n$, if the blending basis $\{B_i(t);\ i=0,1,\ldots,n\}$ is normalized totally positive, and its collocation matrix on $t_0,t_1,\ldots,t_n$ is nonsingular.

Similarly, PIA can also be employed to fit a patch to a data array. Readers can refer to Refs. [3,4] for more details.

More importantly, the PIA technique has the local property. That is, if only a subset of the control points are adjusted iteratively, and the others remain unchanged, the limit curve will interpolate the subset of data points, corresponding to the adjusted control points [9]. The local property of the PIA implies that, the fitting precision of each data point can be controlled separately. As a result, the data points can be fitted adaptively [9].

It can be seen from the generation of the curve sequence $\{\boldsymbol{P}^k(t), k=0,1,\ldots\}$ that, in each iteration, the computation of the foot points, the construction of the difference vectors, and the movement of the control points are fully independent, so it is well suited to be implemented by parallel computing.

3. The extended PIA format and its convergence

As stated above, in the classical PIA, the number of the control points of the blending curve or patch is equal to that of the given data points. It is not feasible in fitting a large number of data points. Therefore, in this paper, we develop an extended PIA (EPIA) format, and show its convergence, where the number of the control points is less than that of the data points.

3.1. The extended PIA for curve fitting

Given a data point sequence,

$$\{\boldsymbol{P}_j; j=0,1,\ldots,n\}, \tag{3}$$

after parametrization, each data point $\boldsymbol{P}_j$ is assigned a parameter $t_j, j=0,1,\ldots,n$ with $t_j<t_{j+1}$.

Suppose the initial curve is

$$\boldsymbol{C}^0(t)=\sum_{i=0}^{l}\boldsymbol{C}_i^0B_i(t), \tag{4}$$

where $\boldsymbol{C}_i^0,\ i=0,1,\ldots,l$, are the initial control points selected from the data points, and $B_i(t), i=0,1,\ldots,l$, are the normalized totally positive blending basis [20,21]. The selection of the initial control points $\boldsymbol{C}_i^0, i=0,1,\ldots,l$, and the knot construction of the initial curve $\boldsymbol{C}^0(t)$ will be explained in Section 4.

Furthermore, the data points (3) are classified into $l+1$ groups, $G_i, i=0,1,\ldots,l$, each of which corresponds to a control point of the initial curve (4), that is,

$$G_i = \{\boldsymbol{P}_j, j \in I_i\}, \quad i=0,1,\ldots,l, \tag{5}$$

where I_i is the index set of the data points in G_i. Note that, the parameters of the data points with indices in I_i should be less than that of the data points with indices in $I_{i+1}, i=0,1,\ldots,l-1$, i.e.,

$$t_{i_0} < t_{i_1} < \cdots < t_{i_l} \quad \text{where } i_0 \in I_0, \ldots, i_l \in I_l. \tag{6}$$

Details on the classification of the data points (3) will be elucidated in Section 4.

Next, supposing the kth curve,

$$\boldsymbol{C}^k(t) = \sum_{i=0}^{l} \boldsymbol{C}_i^k B_i(t), \tag{7}$$

has been generated, to produce the $(k+1)$th curve, we need to calculate the *difference vector for each control point* $\boldsymbol{\Delta}_i^k$. To this end, we first compute the *difference vector for each data point* $\boldsymbol{\delta}_j^k$, which is from the point $\boldsymbol{C}^k(t_j)$ to the corresponding data point $\boldsymbol{P}_j$, namely,

$$\boldsymbol{\delta}_j^k = \boldsymbol{P}_j - \boldsymbol{C}^k(t_j), \quad j=0,1,\ldots,n, \tag{8}$$

and organize them into $l+1$ groups according to the index sets $I_i, i=0,1,\ldots,l$. In this way, we construct the *difference vector for each control point*, i.e.,

$$\boldsymbol{\Delta}_i^k = \frac{\sum_{j \in I_i} \boldsymbol{\delta}_j^k}{|I_i|}, \quad i=0,1,\ldots,l, \tag{9}$$

where $|I_i|$ is the cardinality of the index set I_i.

Moreover, by moving the control point $\boldsymbol{C}_i^k$ along the difference vector $\boldsymbol{\Delta}_i^k$, we get the $(k+1)$th curve, that is,

$$\boldsymbol{C}^{k+1}(t) = \sum_{i=0}^{l} \boldsymbol{C}_i^{k+1} B_i(t), \tag{10}$$

where $\boldsymbol{C}_i^{k+1} = \boldsymbol{C}_i^k + \boldsymbol{\Delta}_i^k, i=0,1,\ldots,l$.

Convergence: As the result of the above iterative procedure, a curve sequence $\{\boldsymbol{C}^k(t), k=0,1,\ldots\}$ is generated. To show its convergence, the difference vectors for the control points are arranged as

$$\Delta^k = \{\boldsymbol{\Delta}_0^k, \boldsymbol{\Delta}_1^k, \ldots, \boldsymbol{\Delta}_l^k\}^{\mathrm{T}}. \tag{11}$$

Thus, the iterative format for the difference vectors (11) is

$$\Delta^{k+1} = D\Delta^k, \tag{12}$$

where $D = I - C$, I is the identity matrix, and,

$$C = \begin{bmatrix} \frac{1}{k_0} \sum_{i_0 \in I_0} B_0(t_{i_0}) & \frac{1}{k_0} \sum_{i_0 \in I_0} B_1(t_{i_0}) & \cdots & \frac{1}{k_0} \sum_{i_0 \in I_0} B_l(t_{i_0}) \\ \frac{1}{k_1} \sum_{i_1 \in I_1} B_0(t_{i_1}) & \frac{1}{k_1} \sum_{i_1 \in I_1} B_1(t_{i_1}) & \cdots & \frac{1}{k_1} \sum_{i_1 \in I_1} B_l(t_{i_1}) \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{k_l} \sum_{i_l \in I_l} B_0(t_{i_l}) & \frac{1}{k_l} \sum_{i_l \in I_l} B_1(t_{i_l}) & \cdots & \frac{1}{k_l} \sum_{i_l \in I_l} B_l(t_{i_l}) \end{bmatrix}. \tag{13}$$

Here, $k_i = |I_i|$ is the cardinality of the index set $I_i, i=0,1,\ldots,l$. Evidentally, the ∞-norm of the matrix C equals 1, i.e., $\|C\|_\infty = 1$.

On the other hand, each square minor of matrix C (13), i.e.,

$$M = \det(C([p_1, p_2, \ldots, p_m; q_1, q_2, \ldots, q_m])), \tag{14}$$

where $p_1, p_2, \ldots, p_m$ are row indices, and $q_1, q_2, \ldots, q_m$ are column indices, can be represented as the sum of the corresponding minors of the following matrices:

$$C_d = \begin{bmatrix} \frac{1}{k_0} B_0(t_{i_0}) & \frac{1}{k_0} B_1(t_{i_0}) & \cdots & \frac{1}{k_0} B_l(t_{i_0}) \\ \frac{1}{k_1} B_0(t_{i_1}) & \frac{1}{k_1} B_1(t_{i_1}) & \cdots & \frac{1}{k_1} B_l(t_{i_1}) \\ \vdots & \vdots & \cdots & \vdots \\ \frac{1}{k_l} B_0(t_{i_l}) & \frac{1}{k_l} B_1(t_{i_l}) & \cdots & \frac{1}{k_l} B_l(t_{i_l}) \end{bmatrix}, \quad i_0 \in I_0, i_1 \in I_1, \ldots, i_l \in I_l. \tag{15}$$

with the same indices as M, i.e.,

$$\det(C_d([p_1, p_2, \ldots, p_m; q_1, q_2, \ldots, q_m])). \tag{16}$$

Since the basis $\{B_i(t), i=0,1,\ldots,n\}$ are normalized totally positive, and $t_{i_0} < t_{i_1} < \cdots < t_{i_l}$ (Eq. (6)), each matrix C_d (15) is totally nonnegative. So, all of its minors are nonnegative. As the sum of the minors (16) of the matrices C_d (15), any square minor M (14) of the matrix C (13) is nonnegative, too. Therefore, the matrix C (13) is a totally nonnegative matrix.

Moreover, if the matrix C (13) is also nonsingular, all of its eigenvalues $\lambda_i(C), i=0,1,\ldots,l$ are positive real numbers [20,21]. Coupled with $\|C\|_\infty = 1$, we have $0 < \lambda_i(C) \le 1$. As a result, the eigenvalues of the iterative matrix D (12) satisfy $0 \le \lambda_i(D) = 1 - \lambda_i(C) < 1$, which means that the iterative format (12) is convergent, that is,

$$\lim_{k \to \infty} \boldsymbol{\Delta}_i^k = 0, \quad i=0,1,\ldots,l. \tag{17}$$

Therefore, the curve sequence $\{\boldsymbol{C}^k(t), k=0,1,\ldots\}$ is also convergent, i.e.,

$$\lim_{k \to \infty} \boldsymbol{C}^k(t) = \boldsymbol{C}^\infty(t).$$

And then, the difference vector $\boldsymbol{\delta}_j^k$ for each data point converges to

$$\lim_{k \to \infty} \boldsymbol{\delta}_j^k = \boldsymbol{P}_j - \boldsymbol{C}^\infty(t_j), \quad j=0,1,\ldots,n.$$

3.2. The extended PIA for patch fitting

Given a data point array

$$\{\boldsymbol{P}_{ij}; i=0,1,\ldots,m, j=0,1,\ldots,n\}, \tag{18}$$

each of them is assigned a parameter (u_i, v_j). Suppose the initial patch is chosen as

$$\boldsymbol{S}^0(u,v) = \sum_{g=0}^{l_u} \sum_{h=0}^{l_v} \boldsymbol{S}_{gh}^0 B_g(u) B_h(v), \tag{19}$$

where $\boldsymbol{S}_{gh}^0$ are the initial control points, $B_g(u)$ and $B_h(v)$ are normalized totally positive basis. The selection of the initial control points $\boldsymbol{S}_{gh}^0$, and the knot construction of the initial patch $\boldsymbol{S}^0(u,v)$ will be explained in Section 4.

Assume again that the data points are classified into groups

$$G_{gh} = \{\boldsymbol{P}_{ij}; (i,j) \in I_{gh}\}, \quad g=0,1,\ldots,l_u, h=0,1,\ldots,l_v,$$

where I_{gh} is the index set of the data points in the group G_{gh}. Each of the groups corresponds to an initial control point. The classification of the data points is based on the parameters of the data points. If

$$\boldsymbol{P}_{i_1,j_1} \in G_{g,h_1}, \quad \boldsymbol{P}_{i_2,j_2} \in G_{g+1,h_2},$$

their parameters (u_{i_1},v_{j_1}), (u_{i_2},v_{j_2}) should satisfy $u_{i_1} < u_{i_2}$; on the other hand, if

$$\boldsymbol{P}_{i_3,j_3} \in G_{g_1,h}, \quad \boldsymbol{P}_{i_4,j_4} \in G_{g_2,h+1},$$

their parameters (u_{i_3},v_{j_3}), (u_{i_4},v_{j_4}) should satisfy $v_{j_3} < v_{j_4}$. The data point classification will be interpreted in Section 4 in detail.

Suppose the kth patch,

$$\boldsymbol{S}^k(u,v) = \sum_{g=0}^{l_u}\sum_{h=0}^{l_v} \boldsymbol{S}^k_{gh}B_g(u)B_h(v), \tag{20}$$

has been generated. Similar to the EPIA for curve fitting (Section 3.1), we first calculate the difference vector $\boldsymbol{\delta}^k_{ij}$ for each data point, i.e.,

$$\boldsymbol{\delta}^k_{ij} = \boldsymbol{P}_{ij} - \boldsymbol{S}^k(u_i,v_j), \quad i=0,1,\ldots,m, \ j=0,1,\ldots,n, \tag{21}$$

and organize them into groups according to the index sets $I_{gh}, g=0,1,\ldots,l_u, h=0,1,\ldots,l_v$. Then, the difference vector $\boldsymbol{\Delta}^k_{gh}$ for each control point is constructed as

$$\boldsymbol{\Delta}^k_{gh} = \frac{\sum_{(i,j)\in I_{gh}} \boldsymbol{\delta}^k_{ij}}{|I_{gh}|}, \quad g=0,1,\ldots,l_u, \ h=0,1,\ldots,l_v, \tag{22}$$

where $|I_{gh}|$ is the cardinality of the index set I_{gh}. By moving the control point $\boldsymbol{S}^k_{gh}$ along the difference vector $\boldsymbol{\Delta}^k_{gh}$, we get the $(k+1)$th patch, that is,

$$\boldsymbol{S}^{k+1}(u,v) = \sum_{g=0}^{l_u}\sum_{h=0}^{l_v} \boldsymbol{S}^{k+1}_{gh}B_g(u)B_h(v), \tag{23}$$

where $\boldsymbol{S}^{k+1}_{gh} = \boldsymbol{S}^k_{gh} + \boldsymbol{\Delta}^k_{gh}, g=0,1,\ldots,l_u, h=0,1,\ldots,l_v$.

In this way, a sequence of patches $\{\boldsymbol{S}^k(u,v); k=0,1,\ldots,\}$ is generated. To show its convergence, we arrange the difference vectors for the control points in a one-dimensional array,

$$\Delta^k = \{\boldsymbol{\Delta}^k_{00}, \boldsymbol{\Delta}^k_{01}, \ldots, \boldsymbol{\Delta}^k_{0,l_v}, \boldsymbol{\Delta}^k_{10}, \ldots, \boldsymbol{\Delta}^k_{1,l_v}, \ldots, \boldsymbol{\Delta}^k_{l_u,0}, \ldots, \boldsymbol{\Delta}^k_{l_u,l_v}\}^{\mathrm{T}}. \tag{24}$$

The iterative format for $\{\Delta^k; k=0,1,\ldots\}$ is

$$\Delta^{k+1} = (I - C_s)\Delta^k, \quad k=0,1,\ldots,$$

where I is the identity matrix, and the matrix C_s is the Kronecker product of two totally nonnegative matrices with ∞-norm equal to 1, one for the parameter u, the other one for the parameter v. Both are similar to Eq. (13). Since the eigenvalues of the Kronecker product of two matrices are the products of the eigenvalues of the two matrices [22], the eigenvalues of the matrix C_s satisfy $0 < \lambda(C_s) \le 1$, if it is nonsingular. Therefore, $0 \le \lambda(I - C_s) < 1$, and the EPIA iterative format (24) for the patch fitting is convergent.

4. The incremental data fitting method by the EPIA format

In this section, an incremental data fitting method is developed using the EPIA format. For clarity, we first list the main steps of the incremental data fitting algorithm in Algorithm 1, where there are two fitting errors, e_d for the data points, defined as

$$e_d = \max_j\{\|\boldsymbol{\delta}^k_j\|\} \quad \text{(refer Eq. (8)) for curve}$$

or

$$e_d = \max_{ij}\{\|\boldsymbol{\delta}^k_{ij}\|\} \quad \text{(Eq. (21)) for patch,} \tag{25}$$

and e_c for the control points, defined as

$$e_c = \max_i\{\|\boldsymbol{\Delta}^k_i\|\} \quad \text{(Eq. (9)) for curve}$$

or

$$e_c = \max_{gh}\{\|\boldsymbol{\Delta}^k_{gh}\|\} \quad \text{(Eq. (22)) for patch.} \tag{26}$$

Algorithm 1. Incremental data fitting algorithm by the EPIA.

```
1  Parametrize the given data points;
2  Construct the initial control points and knots;
3  Classify the data points into groups, each of which
   corresponds to a control point;
4  Check the validation of the initial control points;
5  $k=0$;
6  while The fitting error $e_d$ is not reached do
7      Calculate the foot points and difference vectors on the
       $k$th curve (patch);
8      Generate the new control points for the $(k+1)$th curve
       (patch);
9      if $e_c$ is less than a pre-defined threshold $t_c$ or the current $e_d$
       is larger than the last $e_d$ then
10         Insert a new knot, and divide the corresponding
           groups;
11     end
12     $k=k+1$;
13 end
```

The details on the incremental data fitting algorithm by the EPIA format are elucidated as follows.

Data point parametrization: On one hand, the data point sequence (3) are parametrized by the normalized accumulated chord length method, assigning a parameter $t_j, j=0,1,\ldots,n$ to each data point.

On the other hand, to parametrize the data array (18), we first employ the normalized accumulated chord length method to parametrize each column,

$$\{\boldsymbol{P}_{0j}, \boldsymbol{P}_{1j}, \ldots, \boldsymbol{P}_{mj}\}, \quad j=0,1,\ldots,n,$$

generating the parameter sequence (see Fig. 1),

$$\{u^j_0, u^j_1, \ldots, u^j_m\}, \quad j=0,1,\ldots,n.$$

Then, the parameters in each row are averaged, i.e., $u_i = (1/(n+1))\sum_{j=0}^n u^j_i$, producing the *row parameter sequence*,

$$\{u_0, u_1, \ldots, u_m\}. \tag{27}$$

Similarly, we can calculate the *column parameter sequence*,

$$\{v_0, v_1, \ldots, v_n\}. \tag{28}$$

Thus, the parameter of the data points $\boldsymbol{P}_{ij}$ is $(u_i,v_j), i=0,1,\ldots,m, j=0,1,\ldots,n$.

Initial control point selection: The data point sequence (3) is called *digital curve* in computer vision community, where there are lots of methods presented for calculating the discrete curvature at each data point, and detecting the dominant points of the point sequence. In this paper, we employ the method developed in Ref. [23] to calculate the discrete curvatures at the points, and detect the dominant points of the point sequence (3). As proposed in Ref. [23], we first smooth the point sequence by the Gauss filter

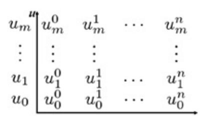
u
u_m u_m^0 u_m^1 $\cdots$ u_m^n
$\vdots$ $\vdots$ $\vdots$ $\vdots$
u_1 u_1^0 u_1^1 $\cdots$ u_1^n
u_0 u_0^0 u_0^1 $\cdots$ u_0^n

Fig. 1. Calculation of the u parameters for the data array.

with adaptive window size, and then calculate the discrete curvature using the following formula, i.e.,

$$k_j = \frac{\|\Delta \boldsymbol{P}_j \times \Delta^2 \boldsymbol{P}_j\|}{\|\Delta \boldsymbol{P}_j\|^3}, \quad j = 0,1,\ldots,n, \tag{29}$$

where $\Delta \boldsymbol{P}_j$ and $\Delta^2 \boldsymbol{P}_j$ are the first and second order difference of the point sequence (3) at the point $\boldsymbol{P}_j$. They constitute the curvature sequence

$$K = \{k_0, k_1, \ldots, k_n\}. \tag{30}$$

And then, the data points with local maximum curvatures are selected as the dominant points of the data point sequence (3) [23].

Thus, we take the selected dominant points as the initial control points, denoted as

$$\{\boldsymbol{C}_i^0 = \boldsymbol{P}_{j_i}, i = 0,1,\ldots,l\}, \tag{31}$$

where $k_{j_i-1} < k_{j_i} < k_{j_i+1}, i = 1,2,\ldots,l-1$. In addition, we also take the head point and tail point as the initial control points, i.e., $\boldsymbol{P}_{j_0} = \boldsymbol{P}_0, \boldsymbol{P}_{j_l} = \boldsymbol{P}_n$.

On the other hand, to choose the initial control points from the data array (18) for patch fitting, we first calculate the curvature for each data point $\boldsymbol{P}_{ij}$ along each row using Eq. (29) by the method proposed in Ref. [23]. Then, each data point gets a curvature k_{ij}^r. Next, we compute the curvature for each data point along each column, and each point gets another curvature k_{ij}^c. Finally, the *total curvature* for each data point $\boldsymbol{P}_{ij}$ is defined as (Fig. 1)

$$k_{ij} = (k_{ij}^r)^2 + (k_{ij}^c)^2, \quad i = 0,1,\ldots,m, \ j = 0,1,\ldots,n.$$

Similar to the generation of the row parameter sequence (27) and column parameter sequence (28), the total curvatures in each row are averaged, generating the *row curvature sequence*,

$$K_{row} = \{k_0^r, k_1^r, \ldots, k_m^r\} \quad \text{where } k_i^r = \frac{\sum_{j=0}^n k_{ij}}{n+1}; \tag{32}$$

the total curvatures in each column are averaged, producing the *column curvature sequence*,

$$K_{col} = \{k_0^c, k_1^c, \ldots, k_n^c\} \quad \text{where } k_j^c = \frac{\sum_{i=0}^m k_{ij}}{m+1}. \tag{33}$$

Similar to the curve case, the indices of the local maximum curvatures in K_{row}, together with the head and tail indices, constitute the *row indices* of the initial control points, i.e.,

$$\{i_0, i_1, \ldots, i_{l_u}\}, \tag{34}$$

where $i_0 = 0, i_{l_u} = m$; the indices of the local maximum curvatures in k_{col}, together with the head and tail indices, constitute the *column indices* of the initial control points, i.e.,

$$\{j_0, j_1, \ldots j_{l_v}\}, \tag{35}$$

where $j_0 = 0, j_{l_v} = n$. Thus, the initial control points for the initial patch are

$$\{\boldsymbol{S}_{gh}^0 = \boldsymbol{P}_{i_g, j_h}, g = 0,1,\ldots,l_u, h = 0,1,\ldots,l_v\}. \tag{36}$$

Knot construction: In our implementation, we employ the *averaging* technique presented in Ref. [24] to construct the knot sequence. As stated above, the initial control points (31) of the initial B-spline curve have the parameter sequence,

$$t_{j_0} = t_0, t_{j_1}, t_{j_2}, \ldots, t_{j_{l-1}}, t_{j_l} = t_n. \tag{37}$$

Then, the knot sequence $\{\bar{t}_j\}$ for the initial B-spline curve with degree p (order $p+1$) are taken as

$$\bar{t}_0 = \bar{t}_1 = \cdots = \bar{t}_p = t_0, \quad \bar{t}_{l+1} = \bar{t}_{l+2} = \cdots = \bar{t}_{l+p} = t_n,$$

$$\bar{t}_{k+p} = \frac{1}{p} \sum_{i=k}^{k+p-1} t_{j_i}, \quad k = 1,2,\ldots,l-p. \tag{38}$$

In the patch fitting case, corresponding to the row indices (34) and the column indices (35), the initial control points of the initial B-spline patch has the row parameter sequence,

$$u_{i_0} = u_0, u_{i_1}, u_{i_2}, \ldots, u_{i_{l_u-1}}, u_{i_{l_u}} = u_m \tag{39}$$

and the column parameter sequence,

$$v_{j_0} = v_0, v_{j_1}, v_{j_2}, \ldots, v_{j_{l_v-1}}, v_{j_{l_v}} = v_n. \tag{40}$$

So, the knots for the initial B-spline patch with degree (p, q) (order $(p+1, q+1)$) are taken as

$$\bar{u}_0 = \bar{u}_1 = \cdots = \bar{u}_p = u_0,$$

$$\bar{u}_{l_u+1} = \bar{u}_{l_u+2} = \cdots = \bar{u}_{l_u+p} = u_m,$$

$$\bar{u}_{k_u+p} = \frac{1}{p} \sum_{k=k_u}^{k_u+p-1} u_{i_k}, \ k_u = 1,2,\ldots,l_u-p; \tag{41}$$

$$\bar{v}_0 = \bar{v}_1 = \cdots = \bar{v}_q = v_0,$$

$$\bar{v}_{l_v+1} = \bar{v}_{l_v+2} = \cdots = \bar{v}_{l_v+q} = v_n,$$

$$\bar{v}_{k_v+q} = \frac{1}{q} \sum_{k=k_v}^{k_v+q-1} v_{j_k}, \ k_v = 1,2,\ldots,l_v-q. \tag{42}$$

As mentioned in Ref. [24], such knot selection manner is efficient for the B-spline fitting.

It should be pointed out that, in B-spline fitting, the knot construction method based on the dominant points in the point sequence is first developed in Ref. [25], where the inflection points are taken as the dominant points. Moreover, Park et al. suggest that the data points with local maximum curvatures can be taken as the dominant points, and present the corresponding knot placement methods for B-spline curve fitting [26], and B-spline patch fitting [27], respectively. The knot construction method in this paper is similar to the method in Refs. [26,27].

Data point classification: The data points should be classified into groups, each of which corresponds to a control point. For this purpose, we need to prepare a *fence F*.

In the curve fitting case, the fence

$$F = \{f_0, f_1, f_2, \ldots, f_l, f_{l+1}\} \tag{43}$$

is constructed based on the curvature sequence (30), the parameters, and the index set of the initial control points (31).

Specifically, the fence F (43) consists of a series of *boundary markers* $f_i, i = 0,1,\ldots,l+1$, where $f_0 = t_0, f_{l+1} = t_n$. The boundary marker f_i is a parameter value in the interval $(t_{j_{i-1}}, t_{j_i})$, where $t_{j_{i-1}}$ and t_{j_i} are the parameters of the data points $\boldsymbol{P}_{j_{i-1}}$ and $\boldsymbol{P}_{j_i}$, respectively. The marker f_i is determined as follows. Suppose the curvatures between $k_{j_{i-1}}$ and k_{j_i} in Eq. (30) are

$$k_s = k_{j_{i-1}}, k_{s+1}, k_{s+2}, \ldots, k_e = k_{j_i},$$

and the parameters of the points between $\boldsymbol{P}_{j_{i-1}}$ and $\boldsymbol{P}_{j_i}$ are

$$t_s = t_{j_{i-1}}, t_{s+1}, t_{s+2}, \ldots, t_e = t_{j_i}.$$

We first normalize them as

$$\bar{k}_i = \frac{k_i - k_{min}}{k_{max} - k_{min}}, \quad \bar{t}_i = \frac{t_i - t_s}{t_e - t_s}, \ i = s, s+1, \ldots, e-1, e,$$

where k_{min} and k_{max} are the minimum and maximum of the curvatures $k_i, i = s, s+1, \ldots, e-1, e$. Then, define,

$$\lambda_{s,w} = \sum_{j=s}^{w-1} e^{\bar{k}_j} \bar{t}_j.$$

Finally, we select a curvature k_w which minimizes the difference of $\lambda_{s,w}$ and $\lambda_{w,e}$, that is,

$$\min_{w} |\lambda_{s,w} - \lambda_{w,e}|, \tag{44}$$

where $s < w < e$. Thus, the corresponding parameter t_w is selected as f_i, i.e., $f_i = t_w$. In addition, if the index j_i is adjacent to j_{i-1}, i.e., $j_{i-1} = j_i - 1$, we define

$$f_i = \frac{t_{j_i-1} + t_{j_i}}{2}.$$

In this way, the data points whose parameters t_j satisfy $f_i \le t_j < f_{i+1}, i = 0,1,\ldots,l$ can be classified into the ith group G_i, corresponding to the ith control point. Differently slightly, the parameters of the data points in the last group G_l should satisfy $f_l \le t_j \le f_{l+1}$.

On the other hand, in the patch case, the aforementioned method is applied, respectively, on the row curvature sequence K_{row} (32) based on the row indices (34) of the initial control points, generating the *row fence*,

$$F_{row} = \{f_g^u, g = 0,1,\ldots,l_u+1\}, \tag{45}$$

and on the column curvature sequence K_{col} (33) based on the column indices (35), producing the *column fence*,

$$F_{col} = \{f_h^v, h = 0,1,\ldots,l_v+1\}. \tag{46}$$

Similarly, the data points whose parameters (u_i, v_j) satisfy $f_g^u \le u_i < f_{g+1}^u$, and $f_h^v \le v_j < f_{h+1}^v$ are classified into the group G_{gh}, corresponding to the control point with index (g, h). Specifically, in the last column, the parameters of the data points fulfill $f_{l_v} \le v_j \le f_{l_v+1}$; in the last row, they satisfy $f_{l_u} \le u_i \le f_{l_u+1}$.

Finally, the local property of PIA allows the users to select some data points to interpolate. If one data point is chosen to interpolate, the group containing the data point will retain just itself, while other data points in the group are distributed into its nearby groups in order. In Fig. 2, the 43 data points are fitted by a cubic B-spline curve with 13 control points, where one data point (in blue) is selected to interpolate.

Difference vector construction: In the curve fitting case, the difference vector for each data point and the difference vector for each control point are defined in Eqs. (8) and (9), respectively.

Similarly, in the patch fitting case, Eqs. (21) and (22) are the difference vectors for data point and control point, respectively.

Validation check: To ensure the decrease of the norm of the difference vector $\boldsymbol{\Delta}_i^k$ (9) for the control point leads to the reduction of that of the difference vector $\boldsymbol{\delta}_j^k$ (8) for the data point, the difference vector $\boldsymbol{\delta}_j^k$ in the group G_i must take *positive* effect in generating the difference vector $\boldsymbol{\Delta}_i^k$. The positive effect concerns two ingredients, the direction and length of the difference vector $\boldsymbol{\delta}_j^k$ in the group G_i. Specifically, the smaller the angle between $\boldsymbol{\delta}_j^k$ and $\boldsymbol{\Delta}_i^k$, the better; the closer the length of the vector $\boldsymbol{\delta}_j^k$ to that of $\boldsymbol{\Delta}_i^k$, the better. To measure the above two ingredients, we employ the following condition:

$$\frac{\|\sum_{\boldsymbol{P}_j \in G_i} \boldsymbol{\delta}_j^k\|}{\sum_{\boldsymbol{P}_j \in G_i} \|\boldsymbol{\delta}_j^k\|} > \eta, \tag{47}$$

where η is a pre-defined threshold. If all of the difference vectors $\boldsymbol{\delta}_j^k$ in the group G_i satisfy Eq. (47), we say that the corresponding control point $\boldsymbol{C}_i^k$ is *valid*; otherwise, it is invalid. In general, the larger value of η requires more control points and groups to meet the condition (47), which will be demonstrated in Section 5. If not specified, we take $\eta = 0.6$ in this paper.

In the patch fitting case, the validation check for each control point is the same as that in the curve fitting case as stated above.

In our implementation, we only check the validation after the initial control point construction.

Knot insertion: As aforementioned, each control point $\boldsymbol{C}_i^0$ corresponds to a data point group G_i, where the parameters of the data points lie in the interval $[f_i, f_{i+1})$.

In the validation check, if the initial control point $\boldsymbol{C}_i^0$ is invalid, the two boundary markers f_i and f_{i+1} are inserted into the corresponding parameter sequence (37), if they are not in the parameter sequence (37) previously. Next, the new knot sequence is generated by the formula (38).

Afterwards, the fence F (43) is updated as follows. First, we delete the markers f_i and f_{i+1} from F; then, insert four new markers into F by the minimum difference rule (44), each in the interval

$$(t_{j_{i-1}}, f_i),\ (f_i, t_{j_i}),\ (t_{j_i}, f_{i+1}) \text{ and } (f_{i+1}, t_{j_{i+1}}),$$

respectively. Finally, the groups $\{G_i\}$ should also be updated correspondingly.

In the iterations, if the fitting error e_c for the control point is less than a pre-defined threshold t_c, or the fitting error e_d in the current iteration is greater than e_d in the last iteration, we search a data point $\boldsymbol{P}_k$, which satisfies

(1) the fitting error $\|\boldsymbol{\delta}_k\|$ (8) to the data point $\boldsymbol{P}_k$ is the biggest, and,
(2) its corresponding parameter t_k is not in the parameter sequence (37);

then, insert its parameter t_k into the parameter sequence (37); finally, generate the new knot sequence by the formula (38). On the other hand, we also insert t_k to the current curve, and get the new control points of the refined curve [24]. The new control points and the new knot sequence produce the new curve for the next iteration.

After that, the fence F (43) and the groups $\{G_i\}$ should also be updated correspondingly. Suppose the inserted parameter t_k lies in between t_{j_i} and $t_{j_{i+1}}$. Then, the two boundary markers between t_{j_i} and t_k, between t_k and $t_{j_{i+1}}$, respectively, should be calculated by the minimum difference rule (44), and the data points with parameters between t_{j_i} and $t_{j_{i+1}}$ should be classified again based on the new boundary markers.

In the patch fitting case, the parameter insertion and knot adjustment, new patch generation, fence and data point group update are similar to the curve case.

After each knot insertion, the number of the control points will increase, and so on. When the number of the control points equals that of the data points, the extended PIA becomes the classical PIA, which guarantees the convergence of the incremental data fitting algorithm by the EPIA format (Algorithm 1).

Last but not the least, the EPIA format can be implemented in parallel totally, thus improving the computational efficiency greatly.

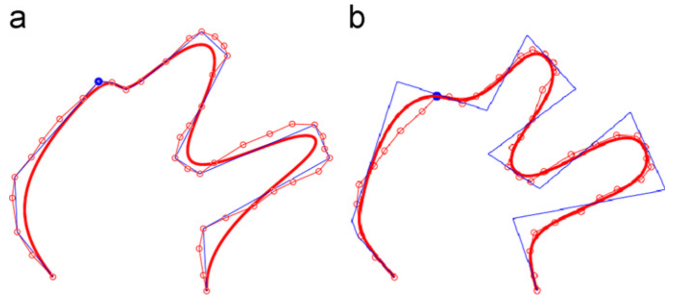

Fig. 2. By the EPIA format, the fitting curve can interpolate a specified data point (in blue solid dot). The data points are represented by the red circle; the control polygon is in blue; the cubic B-spline curve is in red. (a) The initial control polygon and curve. (b) After 10 iterations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Results and discussion

The incremental data fitting algorithm by EPIA format (Algorithm 1) has been implemented in parallel by the *parallel computing toolbox* in *Matlab*, and run on the PC with 3.25 G memory, and four core 2.83 G core 2 Quad CPU. Some empirical results are presented in Figs. 2–6.

As stated above, one of the advantages of the data fitting method by EPIA is that, it can control the fitting precision to each data point individually. For example, we can let the fitting curve interpolate a specified data point $\boldsymbol{P}_j$. To this end, the specified data point should be taken as an initial control point $\boldsymbol{C}_i^0$, and its corresponding data point group G_i should contain just the data point $\boldsymbol{P}_j$ itself. Fig. 2 shows such an example, where a cubic B-spline curve with 13 control points is employed to fit 43 data points, and the data point in blue is selected to interpolate. Fig. 2(a) is the initial control points and initial curve, where the fitting precision to the data point in blue is 0.1265. After 10 iterations, the fitting precision to the blue point improves to 0.0097 (Fig. 2(b)).

As another advantage of the incremental data fitting algorithm by the EPIA format, the new fitting procedure can start from the last fitting result after knot adjustment. Fig. 3(a) is a data point sequence with 269 points, which are extracted from a font image. Fig. 3(b) is the initial control polygon (with 37 control points) and initial cubic B-spline curve generated by the method developed in Section 4. After 15 iterations (Fig. 3(c)), a new knot is inserted in the B-spline curve (Fig. 3(d)). The incremental data fitting algorithm by EPIA starts the new fitting procedure just from the refined B-spline curve (Fig. 3(d)) after knot adjustment. However, if we employ the least square fitting technique here, the new fitting procedure should start from scratch, re-fitting all of the data points by solving a linear system. Fig. 3(e) and (f) is the fitting results after 50 and 70 iterations, respectively. Fig. 3(g) is the final fitting curve with 66 control points after 70 iterations.

Figs. 3 and 4 compare the effects of the threshold η (47) in the validation check (see Section 4). In Fig. 3, $\eta = 0.6$, and the number of the initial control points is 37 (Fig. 3(b)). After 70 iterations, the fitting cubic B-spline curve has 66 control points, with fitting precision $e_d = 0.003501$ (Fig. 3(f)). In Fig. 4, $\eta = 0.9$, and the

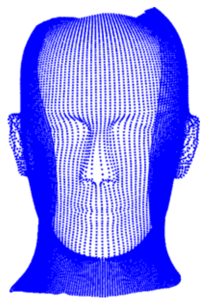
Fig. 5. The data array for fitting.

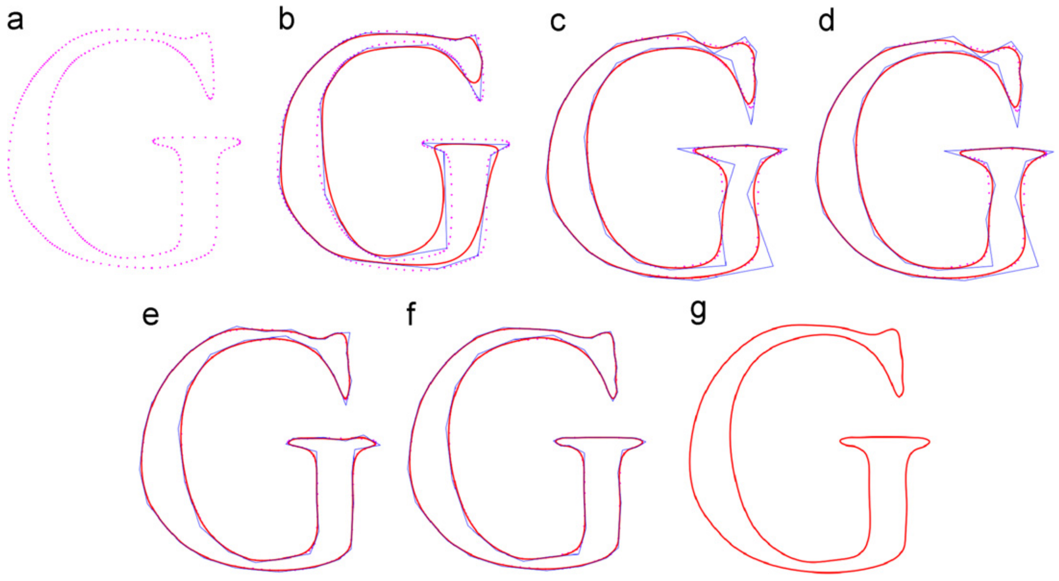

Fig. 3. The incremental data fitting method by the EPIA format starts the new fitting procedure from the last fitting result after knot adjustment. The fitting curve is in red, and the control polygon is in blue. (a) The data points. (b) Initial curve. (c) After 15 iterations. (d) After knot insertion. (e) After 50 iterations. (f) After 70 iterations. (g) The final fitting curve after 70 iterations. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

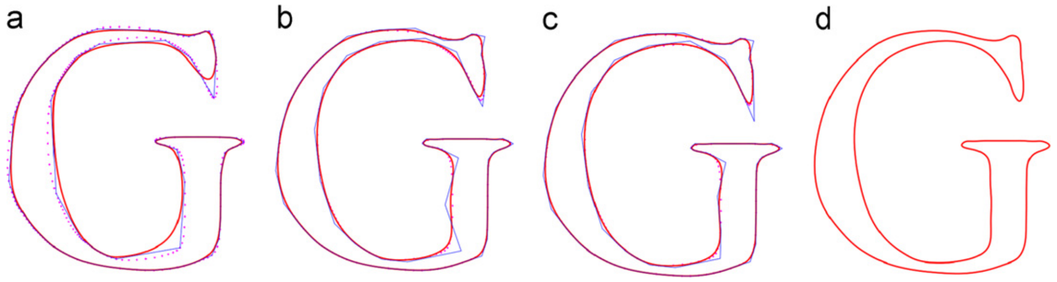

Fig. 4. When $\eta = 0.9$ (Eq. (47)), 83 control points are required to satisfy the validation condition, with initial fitting precision $e_d = 0.07398$. After 40 iterations, the fitting curve has 92 control points, with fitting precision $e_d = 0.003494$. (a) Initial curve. (b) After 10 iterations. (c) After 20 iterations. (d) After 40 iterations.

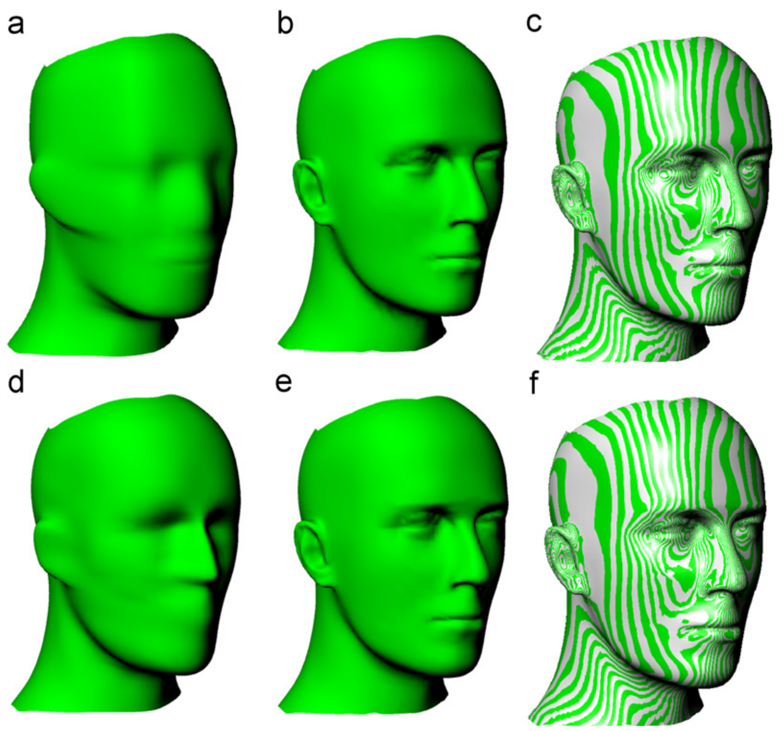

Fig. 6. Fit a data array (Fig. 5) by the incremental data fitting algorithm by EPIA (Algorithm 1), and the method in Ref. [27], respectively. (a) The initial patch of EPIA. (b) The patch after 100 iterations of EPIA. (c) Zebra on the patch in (b). (d) The initial patch of the method in Ref. [27]. (e) The patch after 100 iterations of the method in [27] (f) Zebra on the patch in (e).

Table 1
The empirical data on the examples in Figs. 3 and 4.

$\eta=0.6$ (Fig. 3)			$\eta=0.9$ (Fig. 4)		
#ite.	#control points	e_d	#ite.	#control points	e_d
0	37	0.074793	0	83	0.073984
15	42	0.016525	10	85	0.014109
50	59	0.005326	20	87	0.009085
70	66	0.003501	40	92	0.003494

#ite.: the iteration times.
#control points.: the number of the control points.
e_d: the fitting precision (25).

Table 2
The empirical data on the example in Fig. 6.

Method	Initial patch		After 100 iterations		Time
	#ctrl	e_d	#ctrl	e_d	
EPIA	31×66	0.053347	68×90	0.002576	4.6903
Method in [27]	20×31	0.021707	87×67	0.002743	10.7226

#ctrl: the number of the control points.
e_d: the fitting precision (25).
time: the time cost by 100 iterations, in seconds.

Table 3
The bounding boxes of the models in Figs. 2–6.

Model	Bounding box
Model in Fig. 2	$[1.5786,6.3836]\times[1.0252,8.8742]$
g-Model in Figs. 3 and 4	$[0.1313,0.9454]\times[0.0786,0.9172]$
Model in Fig. 6	$[0.0129,0.6164]\times[0.0048,0.6983]\times[0.0382,0.9587]$

number of the initial control points is 83 (Fig. 4(a)). After 40 iterations, the cubic B-spline curve has 92 control points, with fitting precision $e_d=0.003494$ (Fig. 4(d)). It shows that, the larger the η is, the more control points are required to satisfy the validation condition (47), and the generated initial curve is more faithful to the shape of the data point sequence. The empirical data on the examples in Figs. 3 and 4 are listed in Table 1.

Fig. 6 presents an example where the data array with 121×161 data points (Fig. 5) is fitted by the incremental data fitting algorithm by EPIA (Algorithm 1) and the method presented in Ref. [27], respectively. With the EPIA method, the initial bi-cubic B-spline patch (Fig. 6(a)) has 31×66 control points, with fitting precision $e_d=0.053347$. After 100 iterations, the number of the control points increases to 68×90, and the fitting precision improves to $e_d=0.002576$ (Fig. 6(b)). With the method in Ref. [27], the initial bi-cubic B-spline patch (Fig. 6(c)) has 20×31 control points, with fitting precision $e_d=0.021707$. After 100 iterations, the number of the control points increases to 87×67, and the fitting precision improves to $e_d=0.002743$. Note that, with the EPIA method, the control points of the initial B-spline patch (Fig. 6(a)) are directly chosen from the original data points, while with the method in Ref. [27], the initial patch (Fig. 6(c)) is generated by the least square fitting, so the fitting precision of patch in Fig. 6(c) is less than that of the patch in Fig. 6(a). For comparison of the patch quality, the zebra on the two patches generated by the two methods are illustrated in Fig. 6(e) and (f), respectively. Moreover, the empirical data of this example are summarized in Table 2. It should be pointed out that, the time cost by the EPIA method can be made faster, if more parallel tasks can be invoked.

Finally, to measure the fitting precision, we list in Table 3 the bounding boxes of the three data models in Figs. 2–6.

6. Conclusion

In this paper, we develop an extended progressive-iteration approximation format (EPIA), where the number of the control points is less than that of the data points, and show its convergence. Moreover, we propose an incremental data fitting algorithm by the extended PIA format. The EPIA format performs iteratively by foot point computation, difference vector construction, and new control point generation, till the fitting precision is reached. Since the number of control points is less than that of the data points in the EPIA, it is possible to fit the large scale data points. Furthermore, the three main operations in each iteration of EPIA are all independent itself, so they can be carried out in parallel, thus improving the computational efficiency greatly. Additionally, the storage requirement of EPIA deceases from $O(n^2)$ to $O(n)$, where n is the number of the control points. By the EPIA format, the incremental data fitting algorithm can control the fitting precision to each data point individually, and the new data fitting procedure can start from the last fitting result after knot adjustment. Moreover, after getting the parameters of the unorganized point set, the EPIA method can be employed to fit the unorganized points after minor modifications. Finally, some examples are illustrated in this paper, which are all implemented by the *parallel computing toolbox* in *Matlab*, and run on a PC with a four-core CPU.

Acknowledgments

We thank the reviewers for their valuable comments. This paper is supported by Natural Science Foundation of China (Nos. 60970150, 60933008), and Zhejiang Provincial Natural Science Foundation of China (No. Y1090416).

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.cag.2011.07.003.

References

[1] Qi D, Tian Z, Zhang Y, Feng J. The method of numeric polish in curve fitting. Acta Mathematica Sinica 1975;18(3):173–84.
[2] de Boor C. How does Agee's smoothing method work? In: Proceedings of the 1979 army numerical analysis and computers conference, ARO Report 79-3. Army Research Office, 1979. p. 299–302.
[3] Lin H, Wang G, Dong C. Constructing iterative non-uniform B-spline curve and surface to fit data points. Science in China, Series F 2004;47(3):315–31.
[4] Lin H, Bao H, Wang G. Totally positive bases and progressive iteration approximation. Computers and Mathematics with Applications 2005;50(3–4):575–86.
[5] Delgado J, Peña JM. Progressive iterative approximation and bases with the fastest convergence rates. Computer Aided Geometric Design 2007;24(1):10–8.
[6] Shi L, Wang R. An iterative algorithm of NURBS interpolation and approximation. Journal of Mathematical Research and Exposition 2006;26(4):735–43.
[7] Martin T, Cohen E, Kirby RM. Volumetric parameterization and trivariate B-spline fitting using harmonic functions. Computer Aided Geometric Design 2009;26(6):648–64.
[8] Lu L. Weighted progressive iteration approximation and convergence analysis. Computer Aided Geometric Design 2010;27(2):129–37.
[9] Lin H. Local progressive-iterative approximation format for blending curves and patches. Computer Aided Geometric Design 2010;27(4):322–39.
[10] Maekawa T, Matsumoto Y, Namiki K. Interpolation by geometric algorithm. Computer-Aided Design 2007;39:313–23.
[11] Gofuku S, Tamura S, Maekawa T. Point-tangent/point-normal B-spline curve interpolation by geometric algorithms. Computer-Aided Design 2009;41: 412–22.
[12] Lin H. The convergence of the geometric interpolation algorithm. Computer-Aided Design 2010;42(6):505–8.
[13] Nishiyama Y, Morioka M, Maekawa T. Loop subdivision surface fitting by geometric algorithms. In: Igarashi T, Max N, Sillion F, editors. Poster proceedings of pacific graphics 2008; 2008.
[14] Pottmann H, Leopoldseder S, Hofer M. Approximation with active B-spline curves and surfaces. In: Proceedings of the 10th pacific conference on computer graphics and applications, 2002. IEEE; 2002. p. 8–25.
[15] Wang W, Pottmann H, Liu Y. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. ACM Transactions on Graphics 2006;25:214–38.
[16] Cheng F, Fan F, Lai S, Huang C, Wang J, Yong J. Progressive interpolation using loop subdivision surfaces. In: Chen F, Jüttler, editors. Proceedings of geometric modeling and processing (GMP). Lecture notes in computer science, vol. 4975, 2008. p. 526–33.
[17] Cheng F, Fan F, Lai S, Huang C, Wang J, Yong J. Loop subdivision surface based progressive interpolation. Journal of Computer Science and Technology 2009;24(1):39–46.
[18] Fan F, Cheng F, Lai S. Subdivision based interpolation with shape control. Computer Aided Design & Applications 2008;5(1–4):539–47.
[19] Chen Z, Luo X, Tan L, Ye B, Chen J. Progressive interpolation based on Catmull–Clark subdivision surfaces. Pacific Graphic 2008, Computer Graphics Forum 2008;27(7):1823–7.
[20] Ando T. Totally positive matrices. Linear Algebra Applications 1987;90:165–219.
[21] Karlin S. Total Positivity, vol. I. Standord, CA: Standford University Press; 1968.
[22] Chen JL, Chen XH. Special matrices. Beijing: Tsinghua University Press; 2000.
[23] Ray B, Pandyan R. Acord—an adaptive corner detector for planar curves. Pattern Recognition 2003;36(3):703–8.
[24] Piegl L, Tiller W. The NURBS book. Springer Verlag; 1997.
[25] Li W, Xu S, Zhao G, Goh L. Adaptive knot placement in B-spline curve approximation. Computer-Aided Design 2005;37(8):791–7.
[26] Park H, Lee J. B-spline curve fitting based on adaptive curve refinement using dominant points. Computer-Aided Design 2007;39(6):439–51.
[27] Park H. B-spline surface fitting based on adaptive knot placement using dominant columns. Computer-Aided Design 2011;43(3):258–64.