Pattern Recognition 46 (2013) 1638–1647

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

A global structure-based algorithm for detecting the principal graph from complex data

Hongyun Zhang [a,b,c,*], Witold Pedrycz [b,d], Duoqian Miao [a,c], Caiming Zhong [a,e]

[a] Department of Computer Science and Technology, Tongji University, Shanghai 201804, PR China
[b] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada T6G 2G7
[c] Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai 201804, PR China
[d] System Research Institute, Polish Academy of Sciences, Warsaw, Poland
[e] College of Science and Technology, Ningbo University, Ningbo 315211, PR China

ARTICLE INFO

Article history:
Received 1 March 2012
Received in revised form
10 October 2012
Accepted 11 November 2012
Available online 5 December 2012

Keywords:
Principal curve
Complex data
Global structure
Principal graph
Vertex-merge step
Improved fitting-smoothing phase

ABSTRACT

Principal curves arising as an essential construct in dimensionality reduction and pattern recognition have recently attracted much attention from theoretical as well as practical perspective. Existing methods usually employ the first principal component of the data as an initial estimate of principal curves. However, they may be ineffective when dealing with complex data with self-intersecting characteristics, high curvature, and significant dispersion. In this paper, a new method based on global structure is proposed to detect the principal graph—a set of principal curves from complex data. First, the global structure of the data, called an initial principal graph, is extracted based on a thinning technique, which captures the approximate topological features of the complex data. In terms of the characteristics of the data, vertex-merge step and the improved fitting-and-smoothing phase are then proposed to control the deviation of the principal graph and improve the process of optimizing the principal graph. Finally, the restructuring step introduced by Kégl is used to rectify imperfections of the principal graph. By using synthetic and real-world data sets, the proposed method is compared with other existing algorithms. Experimental results show the effectiveness of the global structure based method.

1. Introduction

Principal component analysis [1] is a well-known technique in multivariate analysis. It is used in dimensionality reduction, feature extraction, and image coding and enhancement. As a nonlinear generalization of principal component analysis, principal curves are defined as one-dimensional (1D) curves that pass through the "middle" of a set of p-dimensional data points, providing smooth and curvilinear summaries of p-dimensional data. These curves satisfy the self-consistency property, i.e., a point on the curve is an average of all data points that project onto it. Principal curves have received significant attention since Hastie and Stuetzle (hereafter HS) introduced the notion of principal curves to solve the problems in traditional machine learning and multivariate data analysis [2]. Considerable work has been reported on the applications of principal curves, such as high-dimensional data partition [3], shape detection [4,5], image skeletonization [6,7], speech recognition [8], noise robustness improvement of time warping methods [9], feature extraction, bill recognition [10–12], intelligent transportation analysis [13], and regression analysis [14].

HS first proposed the concept of principal curve and developed an algorithm for constructing principal curves. The HS's principal curves algorithm (HSPC) finds principal curves by iterating between projecting data onto the curve and estimating conditional expectations on projectors by the scatter smoother or the spline smoother [2]. Referring to the HS's algorithm, many researchers have offered improvements to the theory as well as algorithmic developments. To address the problem of model bias of the HSPC algorithm, Tibshirani introduced a semi-parametric principal curve model (hereafter TPC), in which an EM algorithm was used to estimate principal curves [15]. In 2000, Kégl et al. defined a polygonal curve with k segments and length L as principal curves to solve the problem of convergence of the HSPC algorithm (KPC). The KPC algorithm seeks principal curves by starting with the shortest segment of the first principal component line $f_{1,n}$ which contains all of the projected data points, and in each iteration of the algorithm, it increases the number of segments by one by adding a new vertex to the polygonal curve $f_{k,n}$ produced in the previous iteration. After adding a new vertex, the positions of all vertices are updated so that

* Corresponding author at: Department of Computer Science and Technology, Tongji University, No. 4800, Cao'an Road, Jiading District, Shanghai 201804, PR China. Tel.: +86 21 69589867/13917907676; fax: +86 21 69589359.
E-mail address: zhanghongyun@tongji.edu.cn (H. Zhang).

0031-3203/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2012.11.015

the value of a penalized distance function becomes minimized [16,22]. Delicado in 2001 defined principal curves as principal curves of oriented points to correct bias (DPC). The DPC algorithm finds the principal oriented points one by one and orderly links them to estimate principal curves [17,18]. Verbeek et al. defined *K*-segment principal curves (VPC). The VPC algorithm estimates principal curves by incrementally combining local line segments into the polygonal line to achieve an objective similar to that of Tibshirani's [19]. More recently, Einbeck et al. introduced local principal curves, which were based on the localization of principal component analysis (hereafter LPC). The LPC algorithm generates principal curves by connecting a series of local centers of mass of the data using interpolation or splines, which is similar to Delicado's algorithm [20]. Zhang et al. proposed Riemannian principal curves to address the problem of non-constant data distributions in 2010 (hereafter RPC). The RPC algorithm constructs principal curves by revisiting the projection of the samples onto the curve and incorporating Riemannian distances to reflect the middle of data distribution [21]. Ozertem and Erdogmus in 2011 introduced principal curves and surfaces from a new point of view. They expressed principal curves and surfaces in terms of gradient and the Hessian (hereafter OEPC). OEPC algorithm generates principal curves and surfaces by using subspace constrained mean shift (SCMS) based on kernel density estimation and Gaussian mixture models [23]. Gérard et al. in 2011 proposed parameter selection for principal curves. They considered the principal curve problem from an empirical risk minimization perspective and addressed the parameter selection issue using the point of view of model selection via penalization [24].

Despite the progress reported in the development of principal curve algorithms, there are still some issues that need to be resolved. For instance, the first principal component is often used as the initial estimate of the principal curve when lacking the prior knowledge in existing principal curves algorithms. However, for the complex data with high curvature, significant dispersion and self-intersecting, such as spiral-shaped data, fingerprint data, and alike, the first principal component cannot reflect topological features of these data, so good results cannot be achieved during implementation process of these algorithms. Though some recent PC algorithms do not depend on initial estimates and can achieve good results on data with loops, self-intersecting and bifurcations, such as the OEPC algorithm proposed by Ozertem and Erdogmus in 2011 [23], a good initial estimate may be helpful to improve the performance of the principal curves algorithm, especially for processing the complex data mentioned above. Therefore, we refer to the ideas of Granular Computing [25–30] and propose a global principal graph method (referred to as GPG), which is based on a global structure to detect the principal graph—a set of principal curves from the complex data. Instead of starting with a simple topology such as the first principal component, the GPG directly generate an initial principal graph, which captures the approximate topological features of the complex data by using thinning algorithm. However, the principal graph is not smooth and does not satisfy the self-consistency property. To remedy these drawbacks, we adopt the fitting-and-smoothing step introduced by Kégl [16] to optimize the principal graph by updating the positions of all vertices. During the process of optimizing the principal graph, we find that the complex data may cause the deviation of the principal graph and result in a low efficiency of the algorithm. To address these problems, a vertex-merge step and improved the fitting-and-smoothing step are proposed. Finally, the restructuring step introduced by Kégl [16] is used to further rectify imperfections of the principal graph.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of the concept of principal curves. In Section 3, the global structure based principal curve algorithm is described in detail. Section 4 evaluates and analyzes the performance of the proposed algorithm on synthetic and real-world data sets. Finally, Section 5 provides some conclusions of this study.

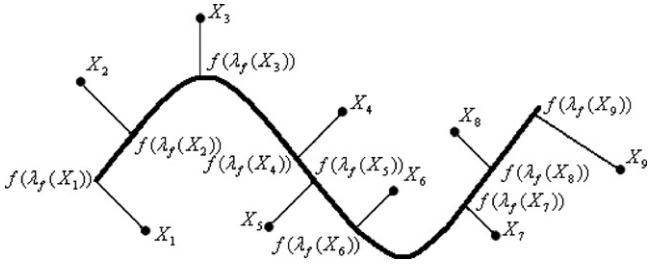

Fig. 1. Projecting points on a curve.

2. Principal curves-some preliminaries

In this section, we review some basic concepts of principal curves. For more details, the reader is referred to [2].

Hastie and Stuetzle generalized the self-consistency property of principal components and introduced the notion of principal curves. Let X denote a random vector in R^d, and $f(\lambda)=(f_1(\lambda),\ldots,f_d(\lambda))$ be a smooth curve in R^d parameterized by $\lambda\in R$. For any $X\in R^d$, let $\lambda_f(X)$ denote the largest parameter value λ for which the distance between X and $f(\lambda)$ is minimized. More formally, the projection index $\lambda_f(X)$ is defined by

$$\lambda_f(X)=\sup\{\lambda:\|X-f(\lambda)\|=\inf_{\tau}\|X-f(\tau)\|\} \quad (1)$$

where $\|.\|$ denotes the Euclidean norm in R^d. Accordingly, the projection point of X to f is $f(\lambda_f(X))$, see Fig. 1.

Definition 1. The smooth curve $f(\lambda)$ is a principal curve if and only if [2]:

(1) $f(\lambda)$ does not intersect itself,
(2) $f(\lambda)$ has finite length inside any bounded subset of R^d,
(3) $f(\lambda)$ is self-consistent, that is

$$f(\lambda)=E[X|\lambda_f(X)=\lambda] \quad \forall\lambda\in R, X\in R^d \quad (2)$$

Roughly speaking, principal curves are smooth one-dimensional (1D) curves that pass through the "middle" of a set of p-dimensional data points [2]. The goal is to provide smooth and low dimensional summaries of these data. Here, a 1D curve in a p-dimensional space is a vector f of p functions indexed by one single variable λ. The parameter λ is the arc length along the curve. The definition of a principal curve states that any point of a principal curve is the condition expectation of those points that project to this point, and a principal curve satisfies the property of self-consistency. Principal curves form a nonlinear generalization of principal component analysis. The theoretical foundations of these curves are a low-dimensional nonlinear manifold embedded in a high-dimensional space [2,3]. Fig. 2 shows a first principal component line and a principal curve. Compared with the corresponding principal component, two obvious advantages of a principal curve can be observed: a principal curve can retain more information about the data. Furthermore, it follows the data more closely and captures a geometric shape of data more accurately.

3. The global principal graph algorithm

Assume that a set of data $X_n=\{x_1,\ldots,x_n\}$ is given. We determine the smooth principal graph which passes through the

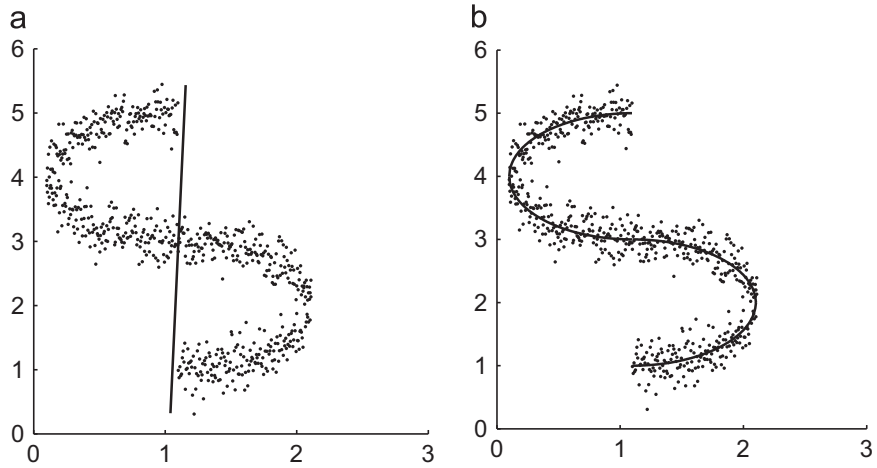

Fig. 2. The comparison between first principal component and principal curve: (a) first principal component; (b) principal curve.

"middle" of such cloud of data. The principal graph is constructed following the strategy outlined below:

(1) Initialize the data points to extract global structure called an initial principal graph G_{vs}^0;
(2) Merge the adjacent vertices of G_{vs}^0 to increase the ratio of graph vertex v_i number to the data point x_i number;
(3) Project the data points X_n and partition them into "the nearest neighbor regions" through the projection step;
(4) Fit and smooth the G_{vs}^0 by updating the positions of all vertices on a basis of the results of projection obtained during in the step of vertex optimization;
(5) Repeat steps (3) and (4) until a local minimum of the penalized distance function $E^n(G)$ has been achieved. In this way, principal curves G_{vs}^k is formed;
(6) Further rectify the structural imperfections of G_{vs}^k during the restructuring step.

In the following sections, we elaborate on these steps in more detail.

3.1. Extraction of the global structure

To overcome the ineffectiveness of the first principal component when it is used as an initial estimate of complex data, we initialize the complex data based on the thinning algorithm [31] to extract the global structure called here an initial principal graph G_{vs}^0, which captures the approximate topological features of the complex data. It roughly follows the medial axis of the complex data. G_{vs}^0 is defined by two sets V and S, where $V=\{v_1,\ldots,v_n\}$ is a set of *vertices*, and $S=\{(v_{i1},v_{j1}),\ldots,(v_{ik},v_{jk})\}=\{s_{i1,j1},\ldots,s_{ik,jk}\}$, $1\le i1,j1,\ldots,ik,jk\le m$ is a set of *edges*, while s_{ij} is a line segment that connects v_i and v_j. For the sake of simplicity, we write $S=\{S_1,\ldots,s_k\}$.

3.2. Merging the adjacent vertices

The initial principal graph G_{vs}^0 does not satisfy the self-consistency property [2] and it is not smooth meaning it contains a number of spurious branches and short loops. In light of this, G_{vs}^0 requires further refinements.

Note that G_{vs}^0 includes hundreds of vertices, and the distribution of the complex data is scattered. In other words, the ratio of the number of data point to the vertex number of G_{vs}^0, which is denoted as α, is low. Through observations coming from experiments, the value of α is usually less than 6. This indicates that only a small number of data points, on average, are involved in adjusting the vertices of G_{vs}^0, which causes the deviation of the G_{vs}^0 and the low efficiency of the algorithm during the process of optimizing the principal graph G_{vs}^0. In order to address these problems, a vertex-merge step based on the distance and curvature criterion has to be completed prior to the optimization of G_{vs}^0. The reason is as follows: (a) The vertex-merge step can effectively reduce the number of vertices, which need to be adjusted. As a result, the efficiency of the algorithm is improved. (b) Vertex-merge step increases the proportion of data points to principal graph vertices. More data points, on average, are involved in adjusting a vertex of G_{vs}^0, so that the deviation of principal graph G_{vs}^0 can be controlled to a certain degree. Vertex-merge algorithm based on the distance and curvature criterion (DCVM) can be summarized as follows:

Algorithm 1. Vertex-merge algorithm.

Input: $G_{vs}^0=(V,S)$, the initial principal graph of vertices to be merged
Output: G_{vs}^1, reduction of G_{vs}^0.

for every vertex $v_i\in G_{vs}^0$ **do**;
  Let v_l be the left adjacent vertex of v_i;
  Let v_r be the right adjacent vertex of v_i;
  Compute the Euclidean distance, $d(v_l,v_i)=\|v_l-v_i\|$ and $d(v_i,v_r)=\|v_i-v_r\|$;
  Compute the threshold d_t, $d_t=\frac{1}{m}\sum_{i=1}^{m} d(v_{i-1},v_i)$;
  if $d(v_l,v_i)+d(v_i,v_r)<k\times d_t$, **Then**
    connect v_l and v_r, and remove v_i;
  end if
  if $d(v_l,v_i)/d(v_i,v_r)>\alpha_u$ or $d(v_l,v_i)/d(v_i,v_r)<\alpha_l$, **Then**
    connect v_l and v_r, and remove v_i;
  end if
end for
for every vertex $v_i\in G_{vs}^0$ **do**;
  Compute the directional angle of v_l and v_i; $\theta_{v_iv_i}=\arctan[(v_i\cdot y-v_l\cdot y)/(v_i\cdot x-v_l\cdot x)]$
  Compute the directional angle of v_i and v_r; $\theta_{v_iv_r}=\arctan[(v_r\cdot y-v_i\cdot y)/(v_r\cdot x-v_i\cdot x)]$
  Let $\beta=\angle v_lv_iv_r$ be the included angle of $\rightarrow_{v_lv_i}$ and $\rightarrow_{v_iv_r}$;
  Compute β, $\beta=|\theta_{v_lv_i}-\theta_{v_iv_r}|$
  if $\beta>\pi$, **Then**
    $\beta=2\pi-\beta$
  end if

21: **if** $\pi-\beta<\theta$, **Then**
22: connect v_l and v_r, and remove v_i;
23: **end if**
24: **end for**

Note that in the algorithm k, α_u, α_l and θ are the parameters whose values have to be determined in an experimental fashion. This problem will be discussed in Section 4.1.

3.3. Fitting and smoothing the principal graph

After reducing the G_{vs}^0 by using vertex-merge algorithm based on the distance and curvature criterion (DCVM), we adopt the fitting-and-smoothing step introduced by Kégl's [16] to optimize G_{vs}^1. However, the fitting-smoothing step of the original algorithm is of low efficiency for complex data. To remedy this drawback, we improve the projection strategy and redefine the penalized distance function in the vertex optimization step to improve the efficiency of the algorithm. For the sake of simplicity, the principal graph is denoted as G in the following formulas. The main idea is as follows:

First, we project the data points onto the principal graph G_{vs}^1 and partition them into "the nearest neighbor regions" in this projection step. Then, the penalized distance function $E^n(G)$ is calculated according to the partition. Finally, we iterate over the vertices, and relocate each vertex by using a gradient (steepest descent) method while all the others are kept fixed, until a local minimum of $E^n(G)$ has been achieved in the vertex optimization step.

The improved projection step: For every data point x_i, scanning the whole principal graph and partitioning the data point x_i into "the nearest neighbor regions" according to which segment or vertex it projects onto in the original algorithm [16]. This step is the most time-consuming, since thousands of scans are required. Considering the characteristic of the complex data, the projection step of the original algorithm can be improved. We only scan certain areas of the graph around the data point x_i and partition it into the nearest neighborhood region instead of the whole graph. Width of the areas of the graph around a data point x_i could be adjusted experimentally, which is denoted as ω. The resulting partition is illustrated in Fig. 3.

The improved vertex optimization step: The penalized distance function $E(G)$ presented in the original algorithm [16] is defined as

$$E(G)=\Delta(G)+\lambda P(G) \tag{3}$$

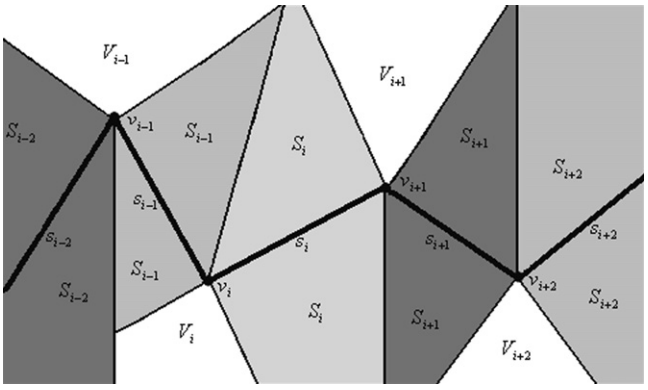

Fig. 3. A nearest-neighbor partition induced by the vertices and edges of a graph.

The first component $\Delta(G)$ is the average squared distance of points in X_n to the graph G defined by

$$\Delta(G)=\frac{1}{n}\sum_{i=1}^{n}\Delta(x_i,G) \tag{4}$$

where n is the number of the data points and $\Delta(x_i,G)$ is the Euclidean squared distance between a point x_i and the nearest point of the principal graph G to x_i. The second component $P(G)$ is a penalty of the total curvature of the graph defined by

$$P(G)=\frac{1}{m}\sum_{i=1}^{m}P_v(v_i) \tag{5}$$

where m is the number of vertices and $P_v(v_i)$ is the curvature penalty at vertex v_i.

Since a lot of triangle functions and piecewise functions are used in the original penalty function $P(G)$ [16], the related computing becomes really time-costing. We have figured out a way to solve this problem by replacing $P(G)$ with a new penalty function $D(G)$. Accordingly, we redefine the penalized distance function as follows:

$$E^n(G)=\Delta(G)+\lambda D(G) \tag{6}$$

The second component $D(G)$ is a penalty imposed on the total curvature of the graph defined by

$$D(G)=\frac{1}{m}\sum_{i=1}^{m}\sum_{x\in V_i\cup S_i}\Delta(x,v_i) \tag{7}$$

From the experimental perspective, the function $D(G)$ exhibits three advantages: (1) it makes the optimization converge fast; (2) it reduces the principal graph deviation; (3) since $D(G)$ only involves simple calculation of addition and average, it is more efficient than $P(G)$ which uses triangle functions.

The decision step: Decide if the adjusted principal graph meets the convergence condition, which means a local minimum of $E^n(G)$ has been achieved. If it is satisfied, output G_{vs}^k and go to Section 3.4, else go to the improved projection step.

Remark 1. It should be noted that λ is a penalty coefficient that trade-offs between the accuracy of the approximation and smoothness of the curves [16]. The smaller the value of $\Delta(G)$ is, the better G_{vs}^k fits the data; $D(G)$ is a penalty on the total curvature of the principal graph. The lower the value of $D(G)$, the smoother G_{vs}^k is.

3.4. Restructuring the principal graph

In the fitting-and-smoothing step, we relocate vertices and edges of the principal graph based on $E^n(G)$, but we do not modify the principal graph in a graph theoretical sense. In the step, we use geometric properties of the principal graph to modify the configuration of vertices and edges by deleting short paths and small loops to get more accurate principal graph. The step is the same as the restructuring step of the original algorithm [16].

3.5. The pseudo-code of GPG algorithm

The essence of the GPG algorithm is outlined as follows:

Algorithm 2. The principal graph algorithm based on global structure (GPG algorithm).

Input: the complex data set $X_n=\{x_1,\ldots,x_n\}$
Output: the principal graph G_{vs}^k.
01: extract the global structure G_{vs}^0 for X_n, by thinning algorithm;
02: reduce the G_{vs}^0 by DCVM algorithm. Set $k=1$;

do fitting and smoothing G_{vs}^k
  for every data point $x_i \in X_n$ **do**;
    for every vertex $v_i \in G_{vs}^k$ and $d(x_i,v_i) < \omega$ **do**
      compute the min $d(x_i,v_i)$, $G(\lambda_G(x_i))$, and $min = \min d(x_i,v_i)$
    end for
    for every vertex $v_i \in G_{vs}^k$ **do**
      if $d(x_i,v_i) = \|x_i - G(\lambda_G(x_i))\|$ and $d(x_i,v_i) = min$, **Then**
        add x_i to the nearest neighbor regions V_i of v_i;
      end if
    end for
    for every line segment $s_i \in G_{vs}^k$ and $d(x_i,s_i) < \omega$ **do**
      compute the min $d(x_i,s_i)$, $G(\lambda_G(x_i))$, and $min = \min d(x_i,s_i)$;
    end for
    for every line segment $s_i \in G_{vs}^k$ **do**
      if $d(x_i,s_i) = \|x_i - G(\lambda_G(x_i)\|$ and $d(x_i,s_i) = min$, **Then**
        add x_i to the nearest neighbor regions S_i of s_i;
      end if
    end for
  end for
  compute the average squared distance of points in X_n to the graph G_{vs}^k

$$\Delta(G) = (1/n)\sum_{i=1}^{n} \Delta(x_i,G)$$

  compute penalty function

$$D(G) = (1/m)\sum_{i=1}^{m} \sum_{x \in V_i \cup S_i} \Delta(x,v_i)$$

  compute the penalized distance function $E^n(G)$

$$E^n(G) = \Delta(G) + \lambda D(G)$$

  for every vertex $v_i \in G_{vs}^k$ **do**;
  compute the gradient $\nabla_{v_i} E^n(G)$ of $E^n(G)$ with respect to v_i;
    execute a line search in the direction of $-\nabla_{v_i} E^n(G)$ (steepest descent) and relocate v_i;
  end for
  $k = k+1$
until find a local minimum of $E^n(G)$
further delete short paths and small loops of G_{vs}^k

3.6. Comparison with Kégl's principal graph

The proposed algorithm comes as an extension and improvement of Kégl's principal graph. The differences between Kégl's method and our algorithm are, therefore, the following: (a) Our algorithm is more widely applicable than Kégl's method. Kégl's principal graph algorithm is an extension of the polygonal line algorithm for finding some smooth skeletons of hand-written character. Since complex data with self-intersecting characteristics, high curvature, and significant dispersion are different from character data, through observations coming from experiments, we find that Kégl's method is not suitable for detecting the principal graph from complex data such as fingerprint data, spiral data, and alike. However, the algorithm presented here can effectively extract the principal graph not only from the hand-written characters but also from other complex data. (b) Vertices-Merge step is added to further preprocess the initial principal graph G_{vs}^0. The distribution of the complex data is scattered, and scattered data cause the low efficiency of Kégl's algorithm and the deviation of the principal graph G_{vs}^k obtained Kégl's method. We propose vertex-merge step based on the distance and curvature criterion to address this problem. (c) Projection strategy and the objective function of our algorithm are different from those of Kégl's algorithm. When the fitting-smoothing step of Kégl's algorithm is adopted to optimize G_{vs}^1, it is of low efficiency for complex data due to the characteristic of the complex data. To remedy this drawback, we improve the projection strategy and redefine the penalized distance function to increase the efficiency.

4. Experimental results and analysis

In this section, we first discuss the parameter setting of the GPG algorithm. Then synthetic data sets and real images are used to demonstrate the performance of GPG algorithm and contrast this performance with the results produced by some traditional methods. Finally, the performance of the GPG algorithm and the impact of noise on the performance of algorithms are investigated.

4.1. Setting parameters

In the proposed GPG algorithms, the parameters k, α_u, α_l and θ are crucial for merging the adjacent vertices. Different values of parameters lead to different levels of reduction. The parameter ω determines whether certain area of the graph is scanned or not. However, it is difficult to select proper values for these parameters because they would be different for data sets with different density distributions. Since for given k, α_u, α_l, θ and ω, it is computationally simple to reduce G_{vs}^0 and project data points, we can inspect all possible values to find optimal result. Empirically, it was found that when we set $k \in [2.6,3.3]$, $\alpha_u \in [3.8,4.3]$, $\alpha_l \in [0.21,0.29]$, $\theta \in [0.32,0.38]$ and $\omega \in [60,70]$, respectively, the parameters form a sound option. According to these empirical intervals, in the reported experiments, the values of the five parameters are further adjusted until the algorithm achieves good performance.

4.2. Experimental results

We carry out two typical experiments on complex data sets. The first experiment tested the capability of the our algorithm for extracting principal curves from data distribution of synthetic data sets, and the second one tested the effectiveness of the proposed algorithm in the real images.

4.2.1. Synthetic datasets

The method has been tested on several synthetic datasets. We generated data sets distributed along some curves by the commonly used additive Gaussian noise, which is independently imposed on different dimensions of the corresponding true curves.

We constructed various shaped curves, such as circle, zigzag-shaped, spiral-shaped, *etc*. Then we compared the results of our algorithm with the results of Kégl et al. (hereafter KKLZ) principal curves, Delicado principal curves, HS principal curves and Ozertem et al. (hereafter OE) principal curves on these datasets. (The curves from KKLZ are obtained via the Principal Curves Java program from Balázs Kégl, available at http://www.iro.umontreal.ca/~kégl/research/pcurves/. The HS curves are obtained by Hastie's Splus function http://lib.stat.cmu.edu/S/principal.curve. Principal curves according to Delicado are determined with a C++ program, which is available at http://www-eio.upc.es/~delicado/PCOP/. The OE curves are obtained using the Principal Curves Matlab code coming from Deniz Erdogmus, available at http://indigo.ece.neu.edu/erdogmus/pubs.html.). We consider several different scenarios: first, $n=200$ data points are generated by means of an underlying circle of radius $r=1$, contaminated with large noise ($\sigma = 0.2$), we then consider three types of spirals: a simple spiral with large noise ($\sigma = 0.07$), a complex spiral with large noise ($\sigma = 0.06$), and a complex spiral with small noise

($\sigma = 0.01$), The first two spirals are composed of 1300 data points, and the third one includes 34,675 data points. Finally, we investigate a zigzag pattern with small ($\sigma = 0.02$) and large noise ($\sigma = 0.1$), where in both cases 800 data points were generated. The results are shown in Fig. 4. The experimental results indicate our principal curves algorithm performs better than the ones by Kégl, Delicado, HS and the OE algorithm.

By looking at the data with moderate noise shown in Fig. 4, respectively, one notices that most algorithms yield satisfactory results, except the HS algorithm. Delicado and KKLZ algorithms fail to handle the complex spiral, however, the result of KKLZ seems to better than that produced by the Delicado principal curves. The results of the OE algorithm are good, except for the peaks of the zigzag curve and the starting part of the spiral curve. In the two cases, the curves obtained by our proposed algorithm are nearly indistinguishable from the true curves. Regarding the noisy data, one can notice that the circle is reconstructed quite differently by all five algorithms, whereby only the proposed algorithm, OE, and Delicado form a closed curve. HS and Delicado have serious problems with the simple noisy spiral. The results of KKLZ and OE seem to be perfect in this case. The results of the proposed algorithm are quite good. HS, Delicado, and KKLZ algorithms fail in case of the complex noisy spiral. The curve obtained by the OE method succeeds to follow the spiral, with an exception for some artificial lines connecting the start of the spiral, however, the result of the algorithm developed in this paper seems to be better than that of OE. The noisy zigzag data are fitted best by KKLZ and the worst results are observed for the HS method. In Section 4.3 we will evaluate the performance of the principal curves in a quantitative way.

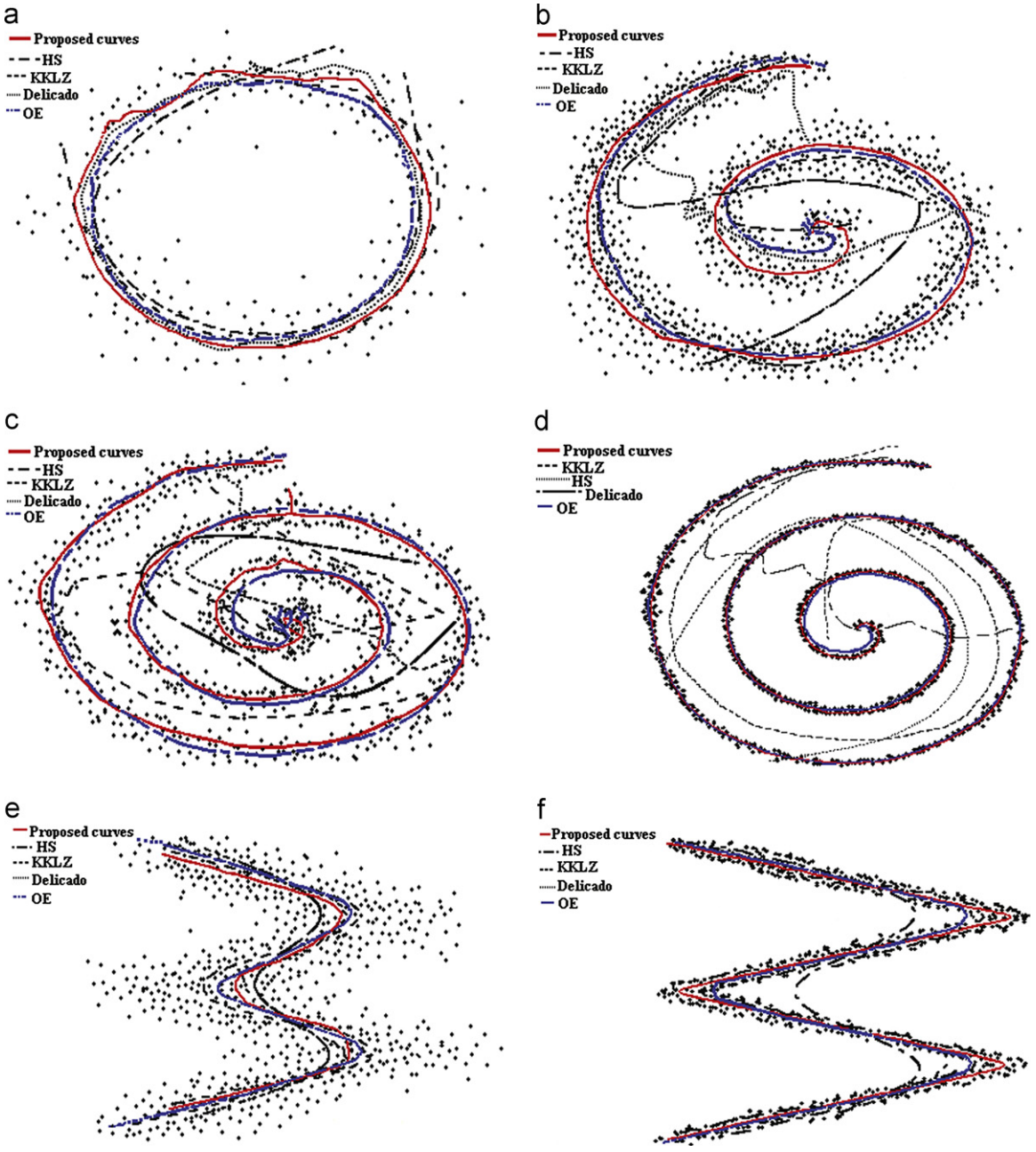

Fig. 4. Principal graphs obtained for synthetic datasets: (a) circle with large noise; (b) simple spiral with large noise; (c) complex spiral with large noise; (d) complex spiral with small noise; (e) zigzag with large noise; (f) zigzag with small noise.

4.2.2. Real-world images

The self-consistency property of principal curves is quite similar to the equidistance property of medial axis of shapes. If foreground pixels of a shape are represented by a two dimensional data set, then the principal curves of this data set are the approximation to its principal graph of shape.

The performance of the algorithm was tested on three suites of real images. The first one comprises the images of isolated handwritten characters captured by using a graphics tablet. The second one is the logo of the Tongji University. The last one is the fingerprint pictures from FVC2000 and FVC2002 fingerprint database [32,33]. Experimental results are shown in Fig. 5. The results confirm the effectiveness of our algorithm on real images whose data distribution follows quite complex pattern.

4.3. Performance analysis

We analyze the performance of the proposed method and the impact of noise (its standard deviation σ) on the performance of the algorithms.

4.3.1. The Quantitative comparison of performance of the algorithms

To further discuss and compare the results of the algorithms, there is a need for some criterion that evaluates the performance

Fig. 5. Principal graphs produced for real-world images: (a) handwritten characters; (b) logo image; (c) fingerprint images.

of a principal graph. This can be done by means of a quantitative measure as fitting deviation. Given a set of data points $X_n = \{x_1, \ldots, x_n\} \subset R_d$, for any $x_i \in X_n$, let $\Delta(x_i, f)$ is the Euclidean squared distance between a point x_i and the nearest point of the principal graph f to x_i. We suppose that f^t is true principal graph and f^e is the estimation of true principal graph by the algorithm called the estimating principal graph. Then fitting deviation (D_f) between estimating principal graph f^e and true principal graph f^t can be defined as

$$D_f = \left| 1 - \frac{\frac{1}{n}\sum_{i=1}^{n} \Delta(x_i, f^t)}{\frac{1}{n}\sum_{i=1}^{n} \Delta(x_i, f^e)} \right| \tag{8}$$

where $\Delta(x_i, f^t) = \min_\lambda \|x_i - f^t(\lambda)\|^2$ and $\Delta(x_i, f^e) = \min_\lambda \|x_i - f^e(\lambda)\|^2$. The smaller the value of D_f, the closer f^e fits f^t. We calculate the fitting deviation of the five algorithms obtained on the six datasets. The quantitative comparison of the performance of the algorithms is shown in Table 1. For zigzag affected by a low level of noise, the results of KKLZ and our method are comparable, however, for the complex noisy spiral (low level of noise), our method and the OE algorithm exhibit good performance. In this case, result produced by the method developed in this study is slightly better than the one formed by the OE. In this case, the HS algorithm and the Delicado's algorithm perform quite poorly. For the noisy data, the HS, KKLZ and Delicado algorithms demonstrate a quite poor performance for the complex spiral. In contrast, the OE comes with a comparatively good performance. For the simple

Table 1
The quantitative comparison for the performance of GPG, HSPC, KKLZPC, OEPC and DPC in the six datasets of Fig. 4.

Method	(a)	(b)	(c)	(d)	(e)	(f)
HS	0.257	0.726	0.812	0.781	0.269	0.243
KKLZ	0.213	0.145	0.713	0.479	0.135	0.094
Delicado	0.102	0.687	0.826	0.809	0.213	0.186
OE	0.093	0.128	0.288	0.157	0.151	0.139
Proposed method (GPG)	0.089	0.107	0.269	0.143	0.164	0.084

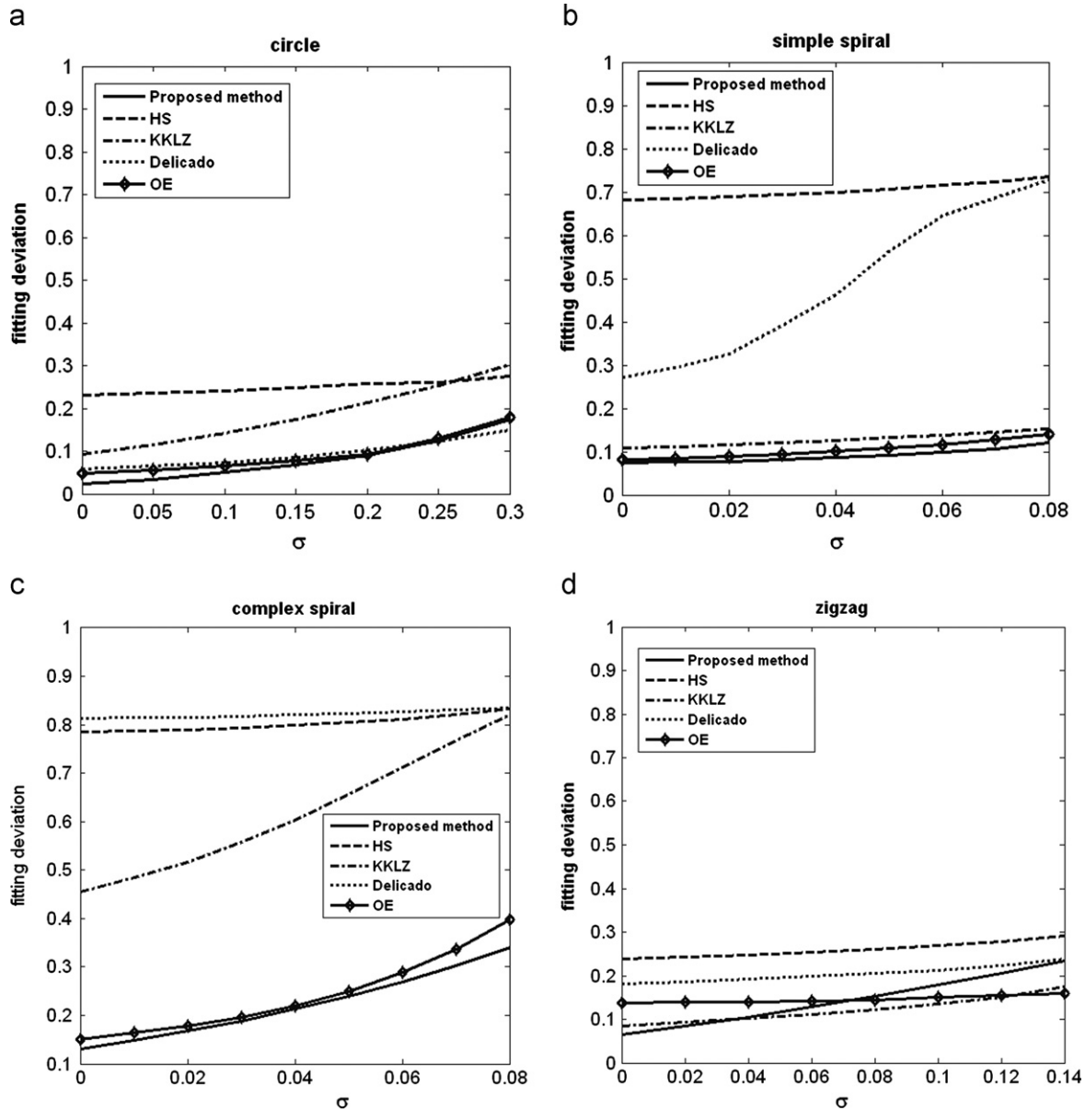

Fig. 6. The relationship between algorithm performance and noise σ in different dataset: (a) circle; (b) simple spiral; (c) complex spiral; (d) zigzag.

spiral, the results generated by the KKLZ and OE seem to be perfect. It is worth noting that our method is always among the best, but is slightly inferior to those of OE and KKLZ when used for the zigzag data.

Since the algorithm starts with some global structure that approximately captures topological features of the complex data instead of the first principal component, good results can be achieved during implementation process of our algorithm. However, our algorithm cannot work the best for zigzag with noise of high standard deviation. The reason is that the thinning algorithm is sensitive to noise σ, which lead to the generated initial principal graph G_{vs}^0 cannot effectively reflect the sharp peak of the zigzag data.

4.3.2. The impact of noise σ on the algorithm performance

The algorithm's sensitivity with regard to noise variations can be detected via modifying values of noise σ. Through four synthetic datasets, we analyse the impact of noise σ on the performance of the algorithm. The relationship between algorithm performance and noise σ is shown in Fig. 6. For a simple spiral, Delicado algorithm is the most sensitive to noise. However, KKLZ algorithm is most sensitive to noise σ for complex spiral and circle. Note that the larger the noise σ, the more sensitive our method. So when noise σ is too large, our algorithm performance is not the best, except for the complex spiral. The reason is that thinning algorithm is sensitive to noise σ, and large noise leads to loss of continuity and distortion of initial principal graph G_{vs}^0. Anyway, our method has obvious advantage in comparison with other several methods when dealing with the complex spiral.

4.3.3. Comparison of computation time

The algorithms were implemented on DELL E7200, Intel CORE 2, WinXP, 2.47 GHz. The average running time of the algorithm was 0.56 s on several simulated data sets, while that of HS, Delicado, and KKLZ algorithms took about 0.47, 0.51 and 0.72 s, respectively. The running time of PL algorithm proposed by KKLZ is almost 1.6 times that of our algorithm, yet the running time of PL algorithm is almost 2.2 times that of SCMS algorithm proposed by Ozertem and Erdogmus (hereafter OE) in 2011 on a simple noisy spiral ($\sigma = 0.08$) [23]. Therefore, the algorithm proposed in this paper is less efficient than HS, Delicado and OE. However, it is more efficient than the KKLZ algorithm. The reason is that the initialization step of the algorithm is time-consuming, but step 2 and step 3 of the proposed algorithm help reduce the running time.

5. Conclusions

Principal curves are nonlinear generalizations of principal components. Since the notion of principal curves was put forward, there have been many methods of how to find principal curves from data sets. However, for the complex data with self-intersecting, high curvature and dispersion, those existing methods may not perform well. On the basis of researching existing work on constructing principal curves, the paper proposed a novel method based on global structure to solve this problem. The basic idea was to initialize data by using thinning algorithm to generate a principal graph which captures the approximate topological features of complex data, and the algorithm optimized the principal graph by updating the positions of all vertices using the fitting-and-smoothing step introduced by Kégl's. During the process of optimizing the principal graph, we found that the complex data might cause the deviation of the principal graph and the low efficiency of algorithm. We proposed vertex-merge step and improved the projection strategy and the penalized distance function in the fitting-and-smoothing step to address the problems in terms of the characteristics of complex data. Our algorithm had been tested on synthetic datasets and applied to real images. Experimental results obtained for selected data confirmed the effectiveness of the proposed method on finding principal curves from complex data. Our future work will focus on if the proposed algorithm can be extended to obtain principal surfaces or even principal manifolds of higher dimensions by developing new concepts and techniques.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (grant No. 61075056, 60970061, 61103067 and 61175054).

References

[1] P. Baldi, K. Hornik, Neural networks and principal component analysis: learning from examples local minima, Neural Networks 2 (1) (1989) 53–58.
[2] T. Hastie, W. Stuetzle, Principal curves, Journal of the American Statistical Association 84 (406) (1989) 502–516.
[3] J.P. Zhang, X.D. Wang, U. Kruger, F.Y. Wang, Principal curve algorithms for partitioning high-dimensional data spaces, IEEE Transactions on Neural Networks 22 (3) (2011) 367–380.
[4] D.C. Stanford, A.E. Raftery, Finding curvilinear features in spatial point patterns: principal curve clustering with noise, IEEE Transactions Pattern Analysis and Machine Intelligence 22 (6) (2000) 601–609.
[5] H.N. Wang, Thomas C.M. Lee, Extraction of curvilinear features from noisy point patterns using principal curves, Pattern Recognition Letter 29 (16) (2008) 2078–2084.
[6] X.B. Liu, Y.D. Jia, A bottom-up algorithm for finding principal curves with applications to image skeletonization, Pattern Recognition 38 (7) (2005) 1079–1085.
[7] E. Bas, D. Erdogmus, Principal curves as skeletons of tubular objects: locally characterizing the structures of axons, Neuroinformatics 9 (2–3) (2011) 181–191.
[8] K. Reinhard, M. Niranjan, Parametric subspace modeling of speech transitions, Speech Communication 27 (1) (1999) 19–42.
[9] U. Ozertem, D. Erdogmus, Principal curve time warping, IEEE Transactions on Signal Processing 57 (6) (2009) 2041–2049.
[10] H.Y. Zhang, The Research of Off-line Handwritten Character Recognition Based on Principal Curves, Ph.D. Thesis, Tongji University, 2005.
[11] H.Y. Zhang, D.Q. Miao, C.M. Zhong, Modified principal curves based fingerprint minutiae extraction and pseudo minutiae detection, International Journal of Pattern Recognition and Artificial Intelligence 25 (8) (2011) 1243–1260.
[12] H.Y. Zhang, D.Q. Miao, D.X. Zhang, Analysis and extraction of structural features of off -line handwritten digits based on principal curves, Journal of Computer Research and Development 42 (8) (2005) 1344–1349.
[13] J.P. Zhang, D.W. Chen, U. Kruger, Adaptive constraint k-segment principal curves for intelligent transportation systems, IEEE Transactions on Intelligent Transportation Systems 9 (4) (2008) 666–677.
[14] E. Jochen, T. Gerhard, E. Ludger, Data Compression and Regression Based on Local Principal Curves, Advances in Data Analysis, Data Handling and Business Intelligence, Springer, Berlin, Heidelberg, 2009.
[15] R. Tibshirani, Principal curves revisited, Statistics and Computing 2 (3) (1992) 183–190.
[16] B. Kgl, A. Krzyzak, T. Linder, Learning and design of principal curves, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (3) (2000) 281–297.
[17] P. Delicado, Another look at principal curves and surface, Journal of Multivariate Analysis 7 (1) (2001) 84–116.
[18] P. Delicado, M. Huetra, Principal curves of oriented points: theoretical and computational improvements, Computational Statistics 18 (2) (2003) 293–315.
[19] J.J. Verbeek, N. Vlassis, B. Krose, A k-segments algorithm for finding principal curves, Pattern Recognition Letter 23 (10) (2002) 1009–1017.
[20] J. Einbeck, G. Tutz, L. Evers, Local principal curves, Statistics and Computing 15 (4) (2005) 301–313.
[21] J.P. Zhang, U. Krger, X.D. Wang, D.W. Chen, A Riemannian distance approach for constructing principal curves, Internal Journal of Neural System 20 (3) (2010) 209–218.
[22] B. Kegl, Principal Curves: Learning, Design, and Applications, Ph.D. Thesis, Concordia University, Canada, 1999.
[23] U. Ozertem, D. Erdogmus, Locally defined principal curves and surfaces, Journal of Machine Learning Research 12 (4) (2011) 1249–1286.
[24] B. Grard, F. Aurlie, Parameter selection for principal curves, IEEE Transactions on Information Theory 57 (12) (2011) 1534–1570.

[25] W. Pedrycz, M.L. Song, Analytic hierarchy process in group decision making and its optimization with an allocation of information granularity, IEEE Transactions on Fuzzy Systems 19 (3) (2011) 527–539.
[26] L.A. Zadeh, Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic, Fuzzy Sets and Systems 90 (1997) 111–117.
[27] W. Pedrycz, B.J. Park, S.K. Oh, The design of granular classifiers: a study in the synergy of interval calculus and fuzzy sets in pattern recognition, Pattern Recognition 41 (12) (2008) 3720–3735.
[28] W. Pedrycz, P. Rai, A multifaceted perspective at data analysis: a study in collaborative intelligent agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B 39 (4) (2009) 834–844.
[29] Y.Y. Yao, Interpreting concept learning in cognitive informatics and granular computing, IEEE Transactions on Systems, Man, and Cybernetics, Part B 39 (4) (2009) 855–866.
[30] W. Pedrycz, The design of cognitive maps: a study in synergy of granular computing and evolutionary optimization, Expert Systems with Applications 37 (10) (2010) 7288–7294.
[31] S. Zhang, K.S. Fu, A thinning algorithm for discrete binary images, in: Proceedings of the International Conference on Computers and Application, Beijing, China, 1984, pp. 879–886.
[32] FVC2000 web site: 〈http://bias.csr.unibo.it/fvc2000/download.asp〉.
[33] FVC2002 web site: 〈http://bias.csr.unibo.it/fvc2002/download.asp〉.

Hongyun Zhang is a Ph.D. holder, a lecturer of Department of Computer Science and Technology at Tongji University, Shanghai, China. She is currently a visiting scholar in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. Her research interests include principal curve, data mining, pattern recognition, and granular computing.

Witold Pedrycz is a professor and Canada Research Chair in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. He is also with the Systems Research Institute of the Polish Academy of Sciences. He is actively pursuing research in Computational Intelligence, fuzzy modeling, pattern recognition, knowledge discovery, neural networks, granular computing and software engineering. He has published vigorously in these areas. He is an author of 11 research monographs and numerous journal papers in highly reputable journals. Dr. Pedrycz has been a member of numerous program committees of international conferences in the area of Computational Intelligence, Granular Computing, fuzzy sets and neurocomputing. He currently serves as an Associate Editor of IEEE Transactions on Fuzzy Systems, IEEE Transactions on Neural Networks. He is also on editorial boards of over 10 international journals. Dr. Pedrycz is also an Editor-in-Chief of Information Sciences and IEEE Transactions on Systems, Man, and Cybernetics part A. He is the past president of IFSA and NAFIPS. He is a Fellow of the IEEE.

Duoqian Miao is a professor of Department of Computer Science and Technology at Tongji University, Shanghai, China. He has published more than 60 papers in international proceedings and journals. His research interests include soft computing, rough sets, pattern recognition, data mining, machine learning and granular computing.

Caiming Zhong is currently pursuing his Ph.D. in Computer Sciences at Tongji University, Shanghai, China. His research interests include cluster analysis, manifold learning and image segmentation.