Computing 75, 359–379 (2005)
Digital Object Identifier (DOI) 10.1007/s00607-005-0122-6

Computing
Printed in Austria

Elastic Principal Graphs and Manifolds and their Practical Applications

A. Gorban, Leicester, and **A. Zinovyev**, Paris

Received January 14, 2005
Published online: May 18, 2005
© Springer-Verlag 2005

Abstract

Principal manifolds serve as useful tool for many practical applications. These manifolds are defined as lines or surfaces passing through "the middle" of data distribution. We propose an algorithm for fast construction of grid approximations of principal manifolds with given topology. It is based on analogy of principal manifold and elastic membrane. First advantage of this method is a form of the functional to be minimized which becomes quadratic at the step of the vertices position refinement. This makes the algorithm very effective, especially for parallel implementations. Another advantage is that the same algorithmic kernel is applied to construct principal manifolds of different dimensions and topologies. We demonstrate how flexibility of the approach allows numerous adaptive strategies like principal graph constructing, etc. The algorithm is implemented as a C++ package *elmap* and as a part of stand-alone data visualization tool *VidaExpert*, available on the web. We describe the approach and provide several examples of its application with speed performance characteristics.

AMS Subject Classifications: 62H25, 62-07, 62-09, 68P05.

Keywords: Principal manifolds, elastic functional, data analysis, data visualization, surface modeling.

1. Introduction

Principal manifolds were introduced by Hastie and Stueltze in 1984, 1989 as lines or surfaces passing through "the middle" of the data distribution [22], [23]. This intuitive definition was supported by mathematical notion of self-consistency: every point of the principal manifold is a conditional mean of all points that are projected into this point. In the case of datasets only one or zero data points are projected in a typical point of the principal manifold, thus, one has to introduce smoothers that become an essential part of the principal manifold construction algorithms.

Since the pioneering work of Hastie, many modifications and alternative definitions of principal manifolds have appeared in the literature. Theoretically, existence of self-consistent principal manifolds is not guaranteed for arbitrary probability distributions. Many alternative definitions were introduced (see, for example, [25]) in order to improve the situation and to allow the construction of principal curves (manifolds) for a distribution of points with several finite first moments. A promising approach is based on analogy of principal manifold and elastic membrane. The

idea of using the elastic energy functional for principal manifold construction in the context of neural network methodology was proposed in mid 1990s (see [9], [13] and bibliography there). This idea was developed and tested on practical applications in [12], [14], [16]–[20], [41]–[43]. Another computationally effective and robust algorithmic kernel for principal curve construction, called the polygonal algorithm, was proposed by Kégl et al. [27]. A variant of this strategy for constructing principal graphs was also formulated in the context of the skeletonization of hand-written digits [26]. An interesting approach we would also like to mention is the construction of principal manifolds in a piece-wise manner by fitting unconnected line segments [38].

Probably, most scientific and industrial applications of principal manifold methodology were implemented using the Kohonen Self-Organizing Maps (SOM) approach developed in the theory of neural networks [24]. These applications are too numerous to be mentioned here. We only mention that SOMs, indeed, can provide principal manifold approximations (for example, see [31], [32]) and are computationally effective. The disadvantage of this approach is that it is entirely based on heuristics; also it was shown that in the SOM strategy there does not exist any objective function that is minimized by the training process [8].

In this paper, we introduce a computationally effective framework for principal manifold construction. Our approach [17], [18], [41] combines ideas developed in [9], [13]–[15] with the approach of Kégl [25], and takes some details from the SOM approach as well. We use grid approximations to the principal manifold, defining manifold in a finite number of points. To describe elastic properties we utilize mesh of springs. The topology of the manifold can be fixed or modified during the process of construction.

Following metaphor of elasticity, we introduce two smoothness penalty terms, which are quadratic at the vertex optimization step. This allows using standard minimization of quadratic functionals (i.e., solving a system of linear algebraic equations with a sparse matrix), which is considerably more computationally effective than gradient optimization of more complicated function, introduced by Kégl.

Minimization of a positive definite quadratic functional can be provided by the sequential one-dimensional minimization for every space coordinate (cyclic). If for a set of coordinates $\{x_i\}_{i\in J}$ terms $x_i x_j$ $(i, j \in J, i \neq j)$ do not present in the functional, then for these coordinates the functional can be minimized independently. The quadratic functional we formulate has a sparse structure, it gives us the possibility to use parallel minimization that is expected to be particularly effective in the case of multidimensional data.

Another feature of our approach is a universal and flexible way to describe grid. A grid approximation to a principal manifold is defined as a connected graph of nodes placed in data space and having a "natural" node placement in a low-dimensional space. The same algorithmic kernel is used to optimize the embedded graph with respect to the dataset. Thus, the same algorithm, given an initial definition of the grid, provides construction of principal manifolds with different dimensions and topologies.

Our algorithm is implemented as a C++ package *elmap* [7] and as a stand-alone application *VidaExpert* for multidimensional data visualization [39]. Some of the applications of the approach to the data visualization were reported in series of works [12], [14], [16]–[20], [41]–[43].

2. Outline of the Method

We define an *elastic net* as a connected unordered graph $G(Y,\boldsymbol{E})$, where $Y = \{y^{(i)},\ i = 1..p\}$ denotes the collection of graph nodes, and $E = \{E^{(i)}, i = 1..s\}$ is the collection of graph edges. We combine some of the incident edges in pairs $R^{(i)} = \{E^{(i)}, E^{(k)}\}$ and denote by $\boldsymbol{R} = \{R^{(i)}, i = 1..r\}$ the collection of *elementary ribs*.

Every edge $E^{(i)}$ has a beginning node $E^{(i)}(0)$ and an ending node $E^{(i)}(1)$. An elementary rib is a pair of incident edges. It has a beginning node $R^{(i)}(1)$, an ending node $R^{(i)}(2)$ and a central node $R^{(i)}(0)$ (see Fig. 1).

Introducing edges is equivalent to introducing connectivity on the graph; this connectivity defines a topology of the principal manifold to be constructed, along with its dimension. Ribs together with edges are used to define a smoothness penalty function, defining in such a way a "natural" form of the graph. Edges connect pairs of nodes, ribs connect triples (or, connect two nodes through another one).

Figure 2 illustrates some examples of the graphs practically used. The first is a simple polyline, the second is a planar rectangular grid, the third is a planar hexagonal grid and the forth is a non-planar graph with nodes arranged on a sphere (spherical grid), then a 3D cubical grid, torus and hemisphere. Elementary ribs at these graphs are incident edges touching with a blunt angle.

We underline here that the grids presented on Fig. 2 correspond to manifolds of different topology and dimension. The grid embedded in data space is optimized with respect to the data point positions.

In optimization criterion we use the standard mean squared point-to-node distance as a main term, and two penalty terms, which are useful to interpret in terms of physical elastic properties of the grid.

For the graph G we define the energy U that includes energies of every node, edge and rib:

$$U = U^{(Y)} + U^{(E)} + U^{(R)}. \tag{1}$$

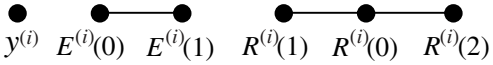

Fig. 1. Node, edge and rib

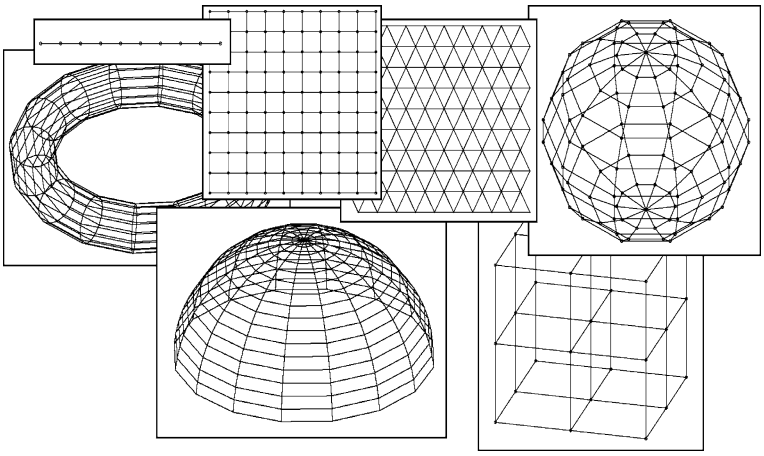

Fig. 2. Elastic nets used in practice

Let us divide the data points into subcollections $K^{(i)}, i = 1 \dots p$. The set K^i contains the data points for which the node y^i is the closest one:

$$K^{(i)} = \left\{ x^{(j)} : \left\| x^{(j)} - y^{(i)} \right\| \leq \left\| x^{(j)} - y^{(m)} \right\|, \quad \text{for all } m = 1, \dots, p \right\}.$$

Let us also assign a weight w_j to every point. We define

$$U^{(Y)} = \frac{1}{\sum\limits_{x^{(j)}} w_j} \sum_{i=1}^{p} \sum_{x^{(j)} \in K^{(i)}} w_j \left\| x^{(j)} - y^{(i)} \right\|^2, \tag{2}$$

$$U^{(E)} = \sum_{i=1}^{s} \lambda_i \left\| E^{(i)}(1) - E^{(i)}(0) \right\|^2, \tag{3}$$

$$U^{(R)} = \sum_{i=1}^{r} \mu_i \left\| R^{(i)}(1) + R^{(i)}(0) - 2R^{(i)}(0) \right\|^2. \tag{4}$$

The $U^{(Y)}$ term is the usual average weighted square of distances between $y^{(i)}$ and data points in $K^{(i)}$; $U^{(E)}$ is the analogue of energy of elastic stretching and $U^{(R)}$ is the analogue of energy of elastic bending of the net. We can imagine that every node is connected by elastic bonds to the closest data points and simultaneously to the adjacent nodes (see Fig. 3).

Values λ_i and μ_j are coefficients of stretching elasticity of every edge $E^{(i)}$ and of bending elasticity of every rib $R^{(j)}$. In the simplest case we have

$$\lambda_1 = \lambda_2 = \dots = \lambda_s = \lambda(s), \quad \mu_1 = \mu_2 = \dots = \mu_r = \mu(r).$$

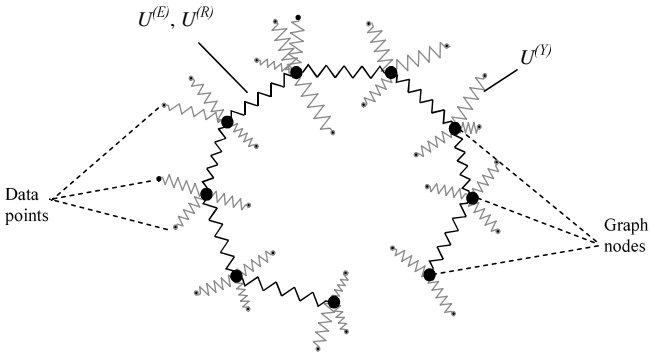

Fig. 3. Energy of elastic net

To obtain $\lambda(s)$ and $\mu(r)$ dependences we simplify the task and consider the case of a regular, evenly stretched and evenly bended grid. Let us consider a lattice of nodes of "internal" dimension d ($d = 1$ in the case of a polyline, $d = 2$ in case of a rectangular grid, $d = 3$ in the case of a cubical grid and so on). Let the "volume" of the lattice be equal to V. Then the edge length equals $(V/s)^{1/d}$. Having in mind that for typical regular grids $r \approx s$, we can calculate the smoothening parts of the functional: $U^{(E)} \sim \lambda s^{\frac{d-2}{d}}$, $U^{(R)} \sim \mu r^{\frac{d-2}{d}}$. Then in the case where we want $U^{(R)}$, $U^{(E)}$ be independent on the grid "resolution",

$$\lambda = \lambda_0 s^{\frac{2-d}{d}}, \quad \mu = \mu_0 r^{\frac{2-d}{d}}, \tag{5}$$

where λ_0, μ_0 are elasticity parameters. This calculation is not applicable, of course, for the general case of any graph. The dimension in this case can not be easily defined and, in practical applications, the λ_i, μ_i are often made different in different parts of a graph according to some adaptation strategy (see below).

The elastic net approximates the cloud of data points and has regular properties. Minimization of the $U^{(Y)}$ term provides approximation, the $U^{(E)}$ penalizes the total length (or, indirectly, "square", "volume", etc.) of the grid and $U^{(R)}$ is a smoother term, preventing the grid from folding and twisting.

In order to perform the vertex optimization step we derive the system of algebraic linear equations to be solved. Let us consider the situation when our collection of data points is already separated in $K^{(i)}$, $i = 1 \ldots p$.

Let us denote

$$\Delta(x, y) = \begin{cases} 1, x = y \\ 0, x \neq y, \end{cases}$$

$$\Delta E^{ij} \equiv \Delta(E^{(i)}(0), y^{(j)}) - \Delta(E^{(i)}(1), y^{(j)}),$$

$$\Delta R^{ij} \equiv \Delta(R^{(i)}(2), y^{(j)}) + \Delta(R^{(i)}(1), y^{(j)}) - 2\Delta(R^{(i)}(0), y^{(j)}).$$

That is, $\Delta E^{ij} = 1$ if $y^j = E^{(i)}(0)$, $\Delta E^{ij} = -1$ if $y^j = E^{(i)}(0)$, and $\Delta E^{ij} = 0$ for all other y^j; $\Delta R^{ij} = 1$ if $y^j = R^{(i)}(1)$ or $y^j = R^{(i)}(2)$, $\Delta R^{ij} = -2$ if $y^j = R^{(i)}(0)$, and $\Delta R^{ij} = 0$ for all other y^j. After a short calculation we obtain the system of p *linear* equations to find new positions of nodes in multi-dimensional space $\{y^i, i = 1 \dots p\}$:

$$\sum_{k=1}^{p} a_{jk} y^{(k)} = \frac{1}{\sum\limits_{x^{(i)}} w_i} \sum_{x^{(i)} \in K_j} w_i x^{(i)},$$

where

$$a_{jk} = \frac{n_j \delta_{jk}}{\sum\limits_{x^{(i)}} w_i} + e_{jk} + r_{jk}, \quad j = 1 \dots p, \tag{6}$$

$$\delta_{jk} = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

and $n_j = \sum\limits_{x^{(i)} \in K^{(j)}} w_i$, $e_{jk} = \sum\limits_{i=1}^{s} \lambda_i \Delta E^{ij} \Delta E^{ik}$, $r_{jk} = \sum\limits_{i=1}^{r} \mu_i \Delta R^{ij} \Delta R^{ik}$. The values of e_{jk} and r_{jk} depend only on the structure of the grid. If the structure does not change then they are constant. Thus only the diagonal elements of the matrix (6) depend on the data set. The a matrix has sparse structure for a typical grid used in practice. In the Appendix, we define this structure, giving an algorithm for calculating only nonzero elements of the matrix.

To minimize the energy of the graph U we use the following two-step iterative algorithm:

(1) Initialize the grid of nodes in data space.
(2) Given the nodes placement, separate the collection of data points into subcollections $K^{(i)}$, $i = 1 \dots p$.
(3) Given this separation, minimize the graph energy U and calculate new positions of nodes.
(4) Go back to step 2.

It is evident that this algorithm converges to a final placement of nodes of the grid (energy U is a non-decreasing value, and the number of divisions of data points into $K^{(i)}$ is finite). Moreover, theoretically the number of iterations of the algorithm before converging is finite. In practice this number may be too large; therefore we interrupt the process of minimization if change of U becomes less than a small value ε or after a fixed number of iterations.

3. Optimization Strategies

We can only guarantee that the algorithm described at the end of the previous section leads to a local minima of the functional. Obtaining a solution close to the global minimum can be a non-trivial task, especially in case where the initial position of

the grid is very different from the expected (or unknown) optimal solution. In many practical situations the "softening" strategy can be used to obtain solutions with low energy levels robustly. This strategy starts with "rigid" grids (small length, small bending and large λ, μ coefficients) at the beginning of the learning process and finishes with soft grids (small λ, μ values), Fig. 4. Thus, the training goes in several epochs, each epoch with its own grid rigidness. The process of "softening" is one of numerous heuristics that pretend to find the global minimum of energy U or rather close configuration.

Nevertheless, for some artificial distributions (like spiral point distribution, used as a test in many papers on principal curves construction) "softening" starting from any linear configuration of nodes does not lead to the expected solution. In this case, adaptive strategies, like "growing curve" (analogue of what was used by Kégl in his polygonal algorithm [27] or "growing surface" can be used to obtain suitable configuration of nodes. This configuration does not have to be optimal, in the adaptation process one can still use the grids more rigid than it is needed for good approximation (thus, providing more robust ways of doing this), finishing the optimization at the next stage with a softer grid (see *spiral* example in the examples section).

4. Adaptive Strategies

The method described above allows us to construct different adaptive strategies by playing with (a) individual λ_i and μ_j weights; (b) the grid connection topology; (c) the number of nodes.

This is a way of extending the approach significantly making it suitable for practical applications. The *elmap* package with implementation of the method described above supports several adaptive strategies that will be described in this section.

First of all, let us define a basic operation on the grid, which allows inserting new nodes. Let us denote by **N**, **S**, **R** the sets of all nodes, edges and ribs respectively. Let us denote by $\mathbf{C}(i)$ the set of all nodes which are connected to the i-th node by an edge. If one has to insert a new node in the middle of an edge I, connecting two nodes k and l, then the following operations have to be accomplished:

(1) Delete from **R** those ribs which contain node k or node l;
(2) Delete the edge I from **S**;

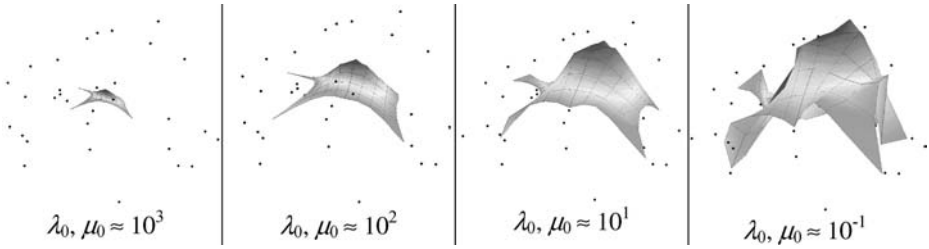

Fig. 4. Training elastic net in several epochs (softening)

(3) Put a new node m in $\mathbf{N}$;
(4) Put in $\mathbf{S}$ two new edges connecting k and m, m and l;
(5) Put in $\mathbf{R}$ new ribs, connecting m, k and all $i \in \mathbf{C}(k)$, and m, l and all $i \in \mathbf{C}(l)$.

At steps 4 and 5 one has to assign new weights to the edges and ribs. This choice depends on the task to be solved. If one constructs a "growing" grid, then these weights must be chosen the same as they were at the deleted ones. If one constructs a refinement of an already constructed grid, one must choose these weights to be twice bigger than they were at the deleted ones.

The *grow-type strategy* is applicable mainly to grids with planar topology (linear, rectangular, cubic grids). It consists of an iterative determining of those grid parts, which have the largest "load" and doubling the number of nodes in this part of the grid. The load can be defined in different ways. One natural way is to calculate the number of points that are projected onto the nodes. For linear grids the grow-type strategy consists of

(1) Initializing the grid; it must contain at least two nodes and one edge;
(2) Determining the edge which has the largest load, by summing the number of data points (or the sum of their weights) projected to both ends of every edge;
(3) Inserting a new node in the middle of the edge, following the operations described above;
(4) Optimizing the positions of the nodes.

One stops this process usually when a certain number of nodes in the grid is reached (see, for example, [28]). This number is connected with the total amount of points. In the *elmap* package this is an explicit parameter of the method, allowing the user to implement his own stopping criterion. Because of this stopping condition the computational complexity is not proportional to the number of data points and, for example, grows like $n^{5/3}$ in the case of the Polygonal Line algorithm. Another form of the stopping condition is when the mean-square error (MSE) does not change more than a small number ε after several insertion/optimization operations.

We should mention here also *growing lump* and *growing flag* strategies used in physical and chemical applications [10], [11]. In growing lump strategy we add new nodes uniformly at the boundary of the grid using a linear extrapolation of the grid embedding. Then the optimization step follows, and, after that, again the step of growing could be done.

For the growing flag one uses sufficiently regular grids, in which many points are situated on the coordinate lines, planes, etc. First, we build a one-dimensional grid (as a one-dimensional growing lump, for example). Then we add a new coordinate and start growing in new direction by adding lines. After that, we can add the third coordinate, and so on.

The *break*-type adaptive strategy changes individual rib weights in order to adapt the grid to those regions of data space where the "curvature" of data distribution has a break or is very different from the average. It is particular useful in applications of

principal curves for contour extraction (see Fig. 7). For this purpose the following steps are performed:

(1) Collect statistics for the distances from every node i to the mean point of the datapoints that are projected into this node:

$$r_j = \left\| y_j - \left(\sum_{x^{(i)} \in K_j} w_i \right)^{-1} \sum_{x^{(i)} \in K_j} w_i x^{(i)} \right\|.$$

(2) Calculate mean and standard deviation for some power of $r : m = \overline{r^\alpha}$, $s = \sigma_{r^\alpha}$; where $\alpha > 1$ is a parameter which in our experiments is chosen to be 4.
(3) Determine those nodes for which $r_j > m + \beta s$, where $\beta > 0$ is another parameter, equal 2 in our experiments.
(4) For every node k determined at the previous step one finds those ribs that have k as their central point and change their weight for $\mu_j^{(new)} = \mu_j^{(old)} \cdot \frac{m}{r_j^\alpha}$.
(5) Optimize the node positions.
(6) Repeat this process a certain number of times.

Principal graph strategy, implemented in the *elmap* package allows performing clustering of curvilinear data features along principal curves. Two example applications of this approach are satellite image analysis [2] or hand-written symbol skeletonization [26] (see also Figs. 8 and 9). First, notice that the grid we constructed does not have to be a connected graph. The system matrix (6) is not singular if for every connected component of the graph there are data points that are projected onto one of its nodes. This allows using the same algorithmic kernel to optimize node positions of unconnected graph. Notice also that if the sets of edges and ribs are empty, then this algorithm acts exactly like standard K-means clustering.

To construct a "skeleton" for two-dimensional point distribution, we apply a variant of local linear principal component analysis first, then connect local components into several connected parts and optimize the node positions after. This procedure is robust and efficient in applications to clustering along curvilinear features and it was implemented as a part of *elmap* package. The following steps are performed:

(1) Make a "grid" from a number of unconnected nodes (sets of edges and ribs are empty at this stage). Optimize the node positions (i.e., do K-means clustering). The number of nodes is chosen to be a certain proportion of the number of data points. In our experiments we used 5% of the total number of data points. At every iteration of the K-means algorithm, the "empty" nodes (those for which there is no data point having this node as this closest one) change their position randomly. After a certain number of K-means iterations, empty nodes (or nodes with only one datapoint as well) are removed from the set of all nodes.
(2) For every node of the grid in position y^i, the local first principal direction is calculated. By local we mean that the principal direction is calculated inside the cluster of datapoints corresponding to the node i. Then this node is substituted by two new nodes in positions $y^{(new1)} = y^i + \alpha s \boldsymbol{n}$, $y^{(new2)} = y^i - \alpha s \boldsymbol{n}$, where $\boldsymbol{n}$ is the unit vector in the principal direction, s is the standard deviation of data

points belonging to the node i, α is a parameter, which we set to 1. These two nodes are connected by an edge (see Fig. 9b).

(3) A collection of edges and ribs is generated, following this simple rule: every node is connected to the node which is closest to this node but not already connected at the step 2. The edges with length much bigger than the average are pruned. Every new edge generates two ribs consisting of a new edge and one of the edges made at step 2.

(4) Weights of the ribs are calculated. A rib is assigned a weight equal to $|\cos(\alpha)|$, where α is an intersection angle of two edges contained in this rib, if $\alpha \geq \frac{\pi}{2}$. Otherwise it is zero (or, equally, the rib is eliminated).

(5) The node positions are optimized.

One possible way to improve the resulting graph further is to apply graph simplification rules, analogously to how it was done in [26]. The idea of this algorithm is close to the k-segments algorithm of Verbeek [38] and, indeed, one possible option is to use k-segment clustering instead of K-means clustering on the first step of the algorithm.

The adaptive strategies: "grow", "break" and the principal graphs can be combined and applied one after another. For example, the principal graph strategy can be followed by break-type weight adaptation or by grow-type grid adaptation.

5. Projecting

In the process of the grid construction we use projection of data into the closest node. This allows us to improve the speed at the data projection step without loosing too much when the grid resolution is good enough. The effect of an estimation bias, connected with this type of projection, was observed in [25]. In our approach the bias is indirectly reduced by utilizing the $U^{(E)}$ smoother term that makes the grid almost isometric (having the same form, the grid will have lesser energy with equal edge lengths). For presentation of data points or for data compression, other projectors can be applied. A natural way to do it is to introduce a set of simplexes on the grid (line segments for one-dimensional grids, triangles for two-dimensional grids, and tetrahedrons for the 3D grids). Then one performs orthogonal projection onto this set. In order to not calculate all distances to all simplexes, one can apply a simplified version of the projector: find the closest node of the grid and then consider only those simplexes that contain this node. This type of projection is used in the *elmap* package and demonstrated by the example on Fig. 9.

Since the grid has penalty on its length (and, for higher dimensions, indirectly, area, volume), the result of the optimization procedure is a bounded manifold, embedded in the cloud of data points. Because of this, if the penalty coefficient is big, many points can have projection on the boundary of the manifold. This can be undesirable, for example, in data visualization applications. To avoid this effect, we introduced in the *elmap* package the possibility to make a linear extrapolation of the bounded rectangular manifold (extending it by continuity in different directions). Other, more complicated extrapolations can be performed as well, like using Carleman's formulas (see [1], [5], [10], [11], [14], [15]).

6. Principal Manifold as Elastic Membrane

Let us discuss in more detail the central idea of this paper: using metaphor of elastic membrane in principal manifold construction algorithm. The system represented on Fig. 3 can be modeled as elastic membrane with external forces applied to the nodes. In this section we consider the question of correspondence between our spring network system and realistic physical systems (evidently, we make comparison in 3D).

Spring meshes are widely used to create physical models of elastic media (for example, [3]). The advantages, comparing with the continuous approaches like Finite Elements Method (FEM), are evident: computational speed, flexibility, possibility to solve the inverse elasticity problem easily [37].

Modeling elastic media by spring networks has a number of applications in computer graphics, where, for example, there is a need to create realistic models of soft tissues (human skin, as an example). In [37], it was shown that it is not generally possible to model elastic behavior of a membrane using spring meshes with simple scalar springs. In [40], the authors introduced complex system of penalizing terms to take into account angles between scalar springs as well as shear elasticity terms. This allowed to improve the results of modeling and develop applications in subdivision surface design.

In a recent paper [21], it was demonstrated that there is an exact correspondence between the FEM approach and spring networks where elastic behavior of every spring is defined by 6×6 matrix

$$K^S = \begin{pmatrix} k^s & -k^s \\ -k^{sT} & k^{sT} \end{pmatrix},$$

where k^s is a 3×3 matrix describing the elastic behavior of spring s with one of the two ends fixed. In particular, to model 2D-elastic membrane by a regular close-packed triangular lattice spring model, one takes the springs with the following stiffness matrix (in the coordinate frame where the spring is oriented along the x-axis)

$$k^s = \frac{1}{2\sqrt{3}} \begin{pmatrix} 3\lambda' + 5\mu' & 0 \\ 0 & \mu' - \lambda' \end{pmatrix}, \tag{7}$$

where λ' and μ' are Lamé constants of the initial membrane. Simple scalar springs can be utilized only in the particular case $\lambda' = \mu'$.

Let us slightly reformulate our problem to make it more close to the standard notations in the elasticity theory. We introduce the $m \times p$-dimensional vector of displacements, stacking all coordinates for every node:

$$u = \{u_1^{(1)}; u_2^{(1)}; ..; u_m^{(1)}; ...; u_1^{(p)}; u_2^{(p)}; ..; u_m^{(p)}\}^T,$$

where m is dimension, p is the number of nodes, $u_i^{(k)}$ is the i-th component of the k-th node displacement. The absolute positions of nodes are $y^{(k)} = \tilde{y}^{(k)} + u^{(k)}$, where $\tilde{y}^{(k)}$ are equilibrium (relaxed) positions. Then our minimization problem can be stated in the following generalized form:

$$u^T E u + D(u; x) \to \min, \tag{8}$$

where E is a symmetric $(m \times p) \times (m \times p)$ element stiffness matrix. This matrix reflects elastic properties of the spring network and has the following properties: (1) it is sparse; (2) it is invariant with respect to translations of the whole system (as a result, for any band of m consecutive rows corresponding to a given node k, the sum of the $m \times m$ off-diagonal blocks should always be equaled to the corresponding diagonal block taken with the opposite sign). The $D(u; x)$ term describes how well the set of data x is approximated by the spring network with the node displacement vector u. It can be interpreted as the energy of external forces applied to the nodes of the system. To minimize (8) we solve the problem of finding equilibrium between elastic internal forces of the system (defined by E) and external forces:

$$Eu = f, \quad f = -\frac{1}{2}\frac{\partial}{\partial u} D(u; x). \tag{9}$$

In the method introduced above, we propose to assemble the matrix E with use of simple scalar springs plus ribs to introduce bending elasticity. The matrix is assembled very similar to how it is described in the Appendix. There is one important point: the springs (edges) have zero rest lengths, it means that equilibrium node positions are all in zero: $\tilde{y}^{(k)} = 0, \ k = 1..p$. The system behavior then can be described as "super-elastic". From the point of view of data analysis it means that we do not impose any pre-defined shape on the data cloud structure.

Let us look at the structure of E for a simple configuration of nodes, see Fig. 5. Edges give local connections, whereas the ribs produce terms that describe connection of two nodes through another (in a rib two ending nodes are connected through the central one). These non-local connections are marked on Fig. 5 by gray circles. This observation tells that generally our stiffness matrix differs in its structure from the one used in the FEM approach (where all connections are local). The same is true for the system of terms used in [40]: for example, the term penalizing angle deviations introduces non-local connections in the corresponding stiffness matrix.

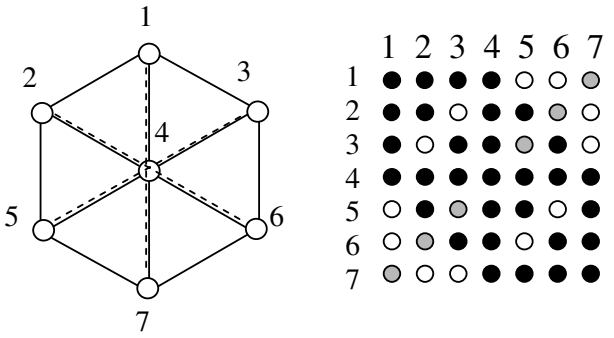

Fig. 5. The stiffness matrix structure for one particular elastic graph. Dashed lines denote ribs. Black circles correspond to local connections of nodes. Gray circles correspond to non-local connections (inside ribs), through one node

For $D(u;x)$ we use the usual mean square distance measure, see (2): $D(u;x) = U^{(Y)}$. The force applied to the j-th node equals

$$f_j = \frac{n^{(j)}}{N}\left(\tilde{x}^{(j)} - u^{(j)}\right) \quad , \tag{10}$$

where

$$\tilde{x}^{(j)} = \frac{\sum\limits_{x^{(i)} \in K^{(j)}} w_i x^{(i)}}{n^{(j)}}, \quad n^{(j)} = \sum_{x^{(i)} \in K^{(j)}} w_i, \quad N = \sum_{x^{(i)}} w_i.$$

It is proportional to the vector connecting the jth node and the weighted average $\tilde{x}^{(j)}$ of the data points in $K^{(j)}$ (i.e., the average of the points that surround the j-th node: see (2) for definition of $K^{(j)}$). The proportionality factor is simply the relative size of $K^{(j)}$. The linear structure of (10) allows to move u in the left part of Eq. (9). Thus the problem is linear.

Now let us show how we can benefit from the definition (9) of the problem. First, we can introduce a pre-defined equilibrium shape of the manifold: this initial shape will be elastically deformed to fit the data. This approach corresponds to introducing a model into the data. After we assemble a physically realistic stiffness matrix E constructed following the recipe from [21]. In a particular but very practical case of a regular close-packed triangular lattice spring model we assemble E using individual spring matrices in the form (7).

Secondly, one can try to change the form (10) of the external forces applied to the system. In this way one can utilize other, more sophisticated approximation measures: for example, taking the outliers into account.

Third, in three-dimensional applications one can benefit from existing solvers for finding equilibrium form of elastic membranes. They can be utilized to solve the problems analogous to the one shown on Fig. 7. For multi-dimensional data point distributions one has to adapt the engines, but this adaptation is mostly formal.

Finally, there is a possibility of a hybrid approach: we utilize first "super-elastic" energy functional (1) to find the initial approximation. Then we "fix" the result and define it as the equilibrium. After we utilize physical elastic functional to find elastic deformation of the equilibrium form to fit the data.

7. Examples

On Fig. 6 we present two examples of 2D-datasets provided by Kégl[1].

The first dataset called *spiral* is one of the standard in the principal curve literature ways to show that one's approach has better performance than the initial algorithm

[1] http://www.iro.umontreal.ca/ ~ Kegl/research/pcurves/implementations/Samples/

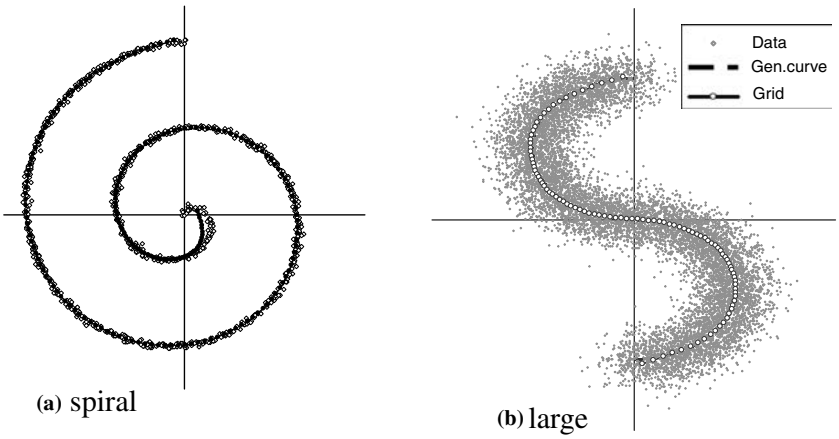

Fig. 6. Two-dimensional examples of principal curves construction

provided by Hastie and Stuelze. As we have already mentioned, this is a bad case for optimization strategies, which start from linear distribution of nodes and try to optimize all the nodes together in one loop. But the adaptive "growing curve" strategy, though being by order of magnitude slower than the "softening", finds the solution quite stably, with exception for the region in the neighborhood of zero, where the spiral has very different (comparing to the average) curvature.

Second dataset, called *large* is a simple case, despite the fact that it has comparatively large sample size (10000 points). The nature of this simplicity lies in the fact that the initial first principal component based approximation is already effective; the distribution is in fact *quasi-linear*, since the principal curve can be unambiguously orthogonally projected onto a line. On Fig. 6b, it is shown that the generating curve, which was used to generate this dataset, has been discovered almost perfectly and in a very short time. To give the idea of speed, we mention that in the case of the simplest optimization (one epoch with fixed grid rigidness, which is suitable in the case of a good initial approximation) the algorithm we described gives the principal curve, approximated by 100 nodes in less than 0.5 seconds on a computer with an Athlon 1800 MHz processor. Application of a softening strategy with 4 epochs gives the principal curve in approximately 1.5 seconds on the same computer.

The third example illustrates modeling of surfaces in 3D. An interesting challenge is to model *molecular surfaces* of complex biological molecules like proteins using principal manifold approach. We extracted the Van-der-Waals molecular surface, using slightly modified Rasmol source code [34] (available from the authors by request) for a simple fragment of DNA. The topology of the surface is expected to be spherical. We should notice that since it is impossible to make the lengths of all edges equal for the sphere-like grid, in the *elmap* package some corrections are performed for edge and rib weights during the grid initialization (shorter edges are given with larger weights proportionally and the same for the ribs). As a result one gets a smooth principal manifold with a spherical topology approximating rather a complicated

set of points. This also allows us to introduce a global spherical coordinate system on the molecular surface. The advantage of this method is its ability to deal not only with star-like shapes as the spherical harmonic functions approach does (see, for example, [4]) but also to model complex forms with cavities as well as non-spherical forms. The result of applying the principal manifold construction by elmap package is shown on Fig. 7.

The forth example demonstrates extracting curvilinear features from images with the *elmap* package. Figure 8 demonstrates how "principal graph" strategy is used for contour extraction. Figure 9 shows how "principal graph" strategy is used for skeletonization of hand-written symbols.

Our final, fifth example illustrates an application of the principal manifold method in multidimensional *data visualization* and dimension reduction. As in the case of molecular surface modeling, we take an example of a dataset from bioinformatics. The genome of C. eleganse (small worm with only one-hundred cells) contains approximately 17000 genes, each of them can be characterized by its codon usage (there are 64 codons, i.e., triplets of 4 genetic letters, this gives a 64-dimensional vector of their frequencies), dinucleotide and nucleotide usage (this gives additional 20 dimensions). The resulting dataset has 17083 points with 84 dimensions. PCA view of the dataset is shown in Fig. 10a. To make noise-filtering, the dataset was projected first into 25-dimensional space spanned by the first 25 principal vectors. In this space, using our *elmap* package, we constructed a two dimensional principal surface, approximated by 1296 nodes. The datapoints were projected onto the manifold by projecting onto the closest point of the manifold (as proposed above). Using a 3-epoch optimization strategy, provided in the sample initialization file for the *elmap* package, it takes 300 seconds to do this on a computer with Athlon 1800 MHz processor. The initial mean-square error (MSE), obtained by a principal plane approximation was 4.59. The resulting manifold provides MSE about 3.60; what is at 22% better than approximation by the principal plane (this value is relatively big, bearing in mind that we approximate a 25-dimensional dataset). The resulting image of projections is shown in Fig. 10b. By changing point forms/sizes we marked two signals that are clearly seen on this plot. More detailed analysis shows that indeed these two groups of points (genes) have very special positions in

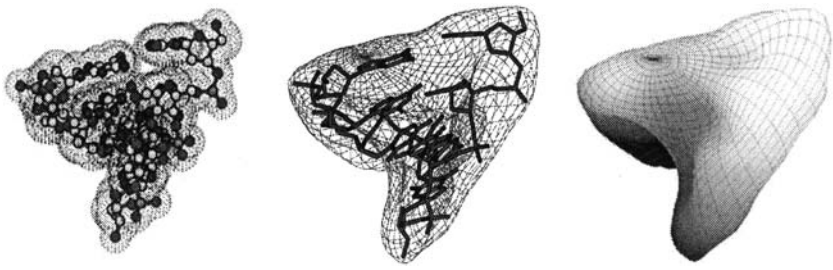

Fig. 7. Construction of principal surface with spherical topology for a distribution of points on Van-der-Waals molecular surface of a biological molecule

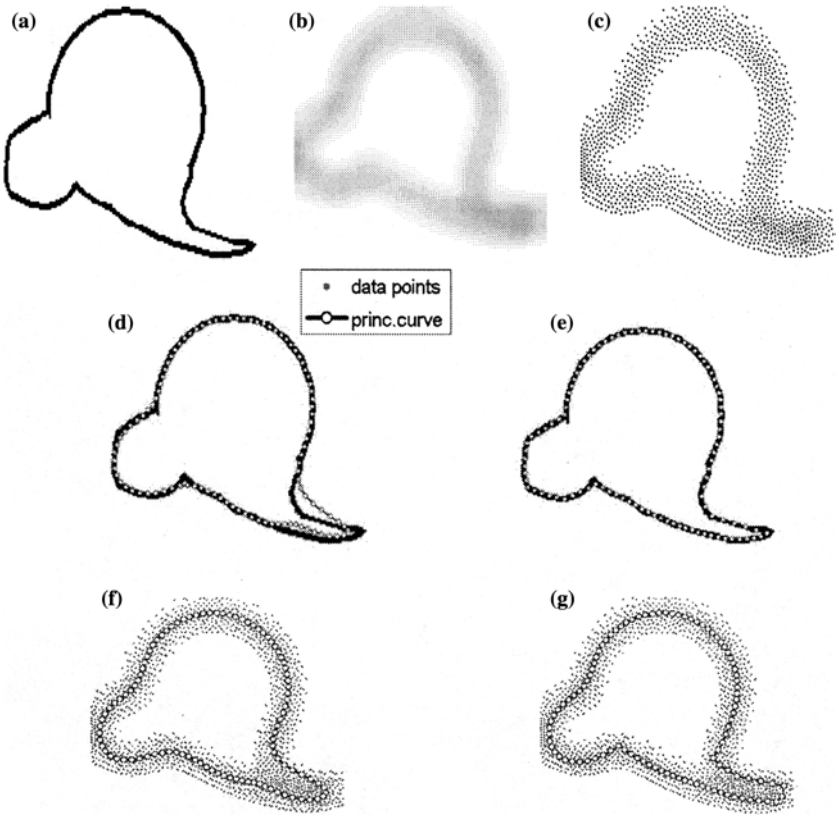

Fig. 8. Contour extraction with closed principal curve. (**a**) initial countour; (**b**) blurred contour; (**c**) Floyd-Steinberg error diffusion color image binarization; (**d, f**) fitting closed principal curve with constant "elasticity", regions of higher curvature can not be fitted equally well; (**e, g**) fitting closed principal curve with adaptive elasticity ("break" adaptation strategy)

the dataspace (i.e., codons and dinucleotide compositions) with respect to the main cluster of data. The principal manifold we constructed can be utilized for displaying different functions defined in the dataspace. In Fig. 10c, visualization of a simple non-parametric estimation of the density distribution is shown. One can see that in general the nonlinear manifold captures more essential features of the dataset than the PCA plot.

8. Method Implementation

In the implementation of the algorithm we used the SparseLib [6] library together with IML++ library to store the matrix and to solve the system of linear equations. We used the BLAS kernel provided by the authors of SparseLib without any platform-specific optimization. This combination showed rather good performance

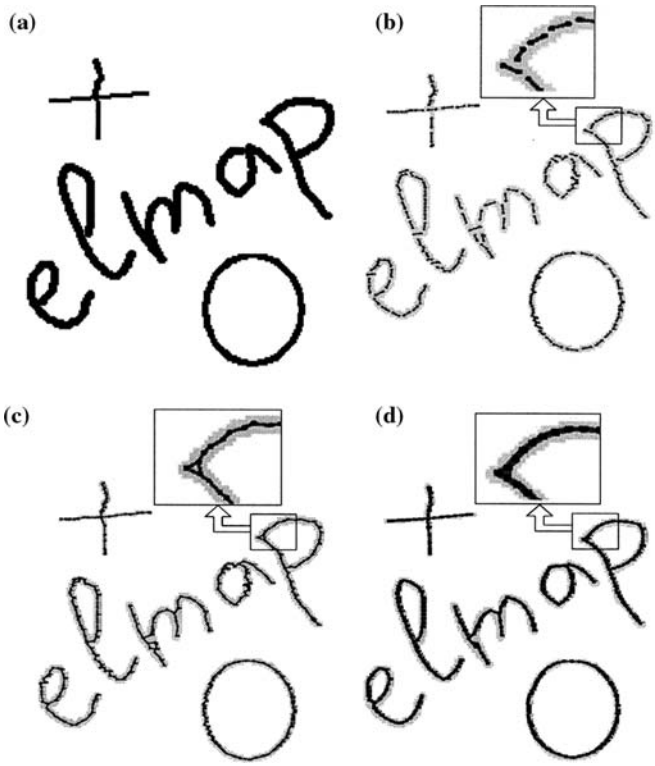

Fig. 9. Skeletonization using principal curves: (**a**) initial image; (**b**) calculation of local principal components; (**c**) connecting the graph; (**d**) graph vertices optimization with principal manifolds algorithm

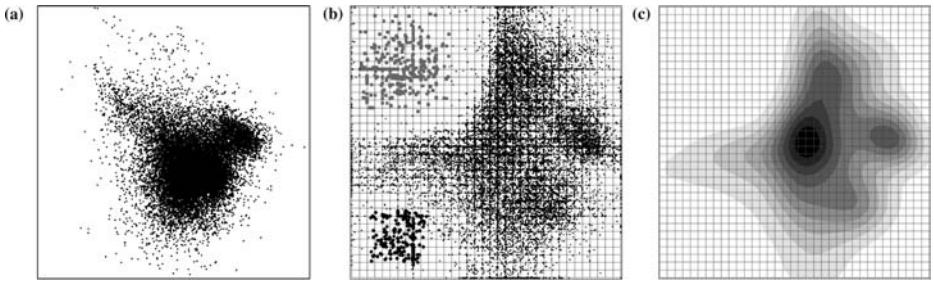

Fig. 10. Visualization of a big dataset in 84-dimensional space. (**a**) PCA view; (**b**) projection onto the manifold constructed; two strong signals are marked by changing point sizes/forms; (**c**) principal manifold as a screen for displaying points density distribution

characteristics, still being easily portable, i.e., written, using ANSI standards. The *elmap* package together with a stand-alone data visualization tool *VidaExpert* are available online [7], [39].

9. Discussion

We introduced a new algorithmic kernel for calculating grid approximations for principal manifolds of different topologies and dimensions. The main advantages of this method are speed and good performance. The optimization criterion we formulated has a particularly simple form and natural physical interpretation. Together with the usual mean square node-to-point distance term our minimized functional contains two penalizing terms: $U^{(E)}$ and $U^{(R)}$, both quadratic with respect to the grid node positions. As one can see from (3) and (4) they are similar to the sum of squared grid approximations of the first and second derivatives, in the directions, guided by natural choice of ribs[2]. The $U^{(E)}$ term penalizes the total length (or area, volume) of the principal manifold and, indirectly, makes the grid regular by penalizing non-equidistant distribution of nodes along the grid. The $U^{(R)}$ term is a smoothing factor. It penalizes the nonlinearity of the ribs embedding into the Euclidean space. This term is quadratic, it gives us benefits in comparison with the cosine function as in the algorithm of Kégl [27], for example.

Attractive characteristics of the method such as its universality, speed and inherited parallelism open new fields to the applications of principal manifolds, especially for the analysis of huge datasets with hundreds of thousands of points with dimensionality of the order of hundreds. The algorithm we described with its C++ implementation provide a way to construct a principal manifold for these datasets approximated by a number of nodes of the order of 10000 in a reasonable time.

In applications of principal manifolds to 3D-surface modeling, one can find similar "physics-based" new methods in surface modeling in computer graphics (see, for example [30], [40]). The method of constructing the elastic energy functional considered here can be compared with the approach described in [40]. Our functional contains only restricted subset of elastic energies proposed there; we utilize such a "physics-based" model, which allows quadratic description, thus leads to quadratic optimization problem. In this way we significantly speed-up the optimization step. Also one can consider use of physically realistic energy functionals and pre-defined equilibrium forms as described above. In general, our point is to construct computationally effective approximation method rather than closely imitate realistic behavior (though it is also possible): this is particularly true for multidimensional applications where the notion of "physical realism" does not make sense.

One important application of principal manifolds is dimension reduction and data visualization. In this field they compete with multidimensional scaling methods and the recently introduced advanced algorithms of dimension reduction, such as locally linear embedding (LLE) [33] and ISOMAP [36] algorithms. The difference between the two approaches is that the later ones seek new point coordinates directly and do not use any intermediate geometrical objects. This has several advantages, in

[2] The differences should be divided by node-to-node distances in order to be true derivative approximations, but in this case the quadratic structure of the term would be violated. We suppose that the grid is regular with almost equal node-to-node distances, then the dependence of coefficients λ_i, μ_j on the total number of nodes contains this factor.

particular that a) there is a unique solution for the problem (the methods are not iterative in their nature, there is no problem of grid initialization) and b) there is no problem of choosing a good way to project points onto a nonlinear manifold. Another advantage is that the methods are not limited by several first dimensions in dimension reduction (it is difficult in practice to manipulate nonlinear manifolds of dimension more than three).

Principal manifold can serve as a nonlinear low-dimensional screen to project data. It gives additional benefits to users. First, the manifold approximates data and can be used itself, without applying projection, to visualize different functions defined in data space (for example, density estimation). Also the manifold as an intermediate that fixes the structure of a learning dataset, can be used in visualization of data points that were not used in the learning process, for example, for visualization of dataflow "on the fly". Constructing manifolds does not use a point-to-point distance matrix that is particularly useful for large datasets. Also using principal manifolds is expected to be more robust to additive noise than the methods based on the local properties of point-to-point distances. To conclude this short comparison, LLE and ISOMAP methods are more suitable if the low-dimensional structure in multi-dimensional data space is complicated but is expected to exist, and if the data points are situated rather tightly on it. Principal manifolds are more applicable for the visualization of real-life noisy observations, appearing in economics, biology, medicine and other sciences, and for constructing data screens showing not only the data but different related functions defined in data space.

Appendix

Constructing the Sparse Matrix

Matrix (6) has p^2 elements (where p is a number of grid nodes), but for typical grids only kp of them are nonzero, where $k \ll p$. Here we provide a simple procedure to fill only nonzero elements of the matrix, thus, define its sparse structure.

For the e^{jk} matrix:

(1) All e^{jk} values are initialized by zero;
(2) If for an edge E^i with weight λ_i, the beginning node is y^{k1} and the ending node is y^{k2}, then we update the e^{jk} values:

$$e^{k_1k_1} = e^{k_1k_1} + \lambda_i, e^{k_2k_2} = e^{k_2k_2} + \lambda_i, e^{k_1k_2} = e^{k_1k_2} - \lambda_i, e^{k_2k_1} = e^{k_2k_1} - \lambda_i.$$

(3) Steps 1–2 are repeated for every edge.

For the r^{jk} matrix:

(1) All r^{jk} values are initialized by zeros;
(2) If for a rib R^i with weight μ_i, the beginning node is y^{k1}, the middle node is y^{k2} and the ending node is y^{k3}, then we update the r^{jk} values:

$$r^{k_1k_1} = r^{k_1k_1} + \mu_i, r^{k_2k_2} = r^{k_2k_2} + 4\mu_i, r^{k_3k_3} = r^{k_3k_3} + \mu_i$$

$$r^{k_1k_2} = r^{k_1k_2} - 2\mu_i, r^{k_2k_1} = r^{k_2k_1} - 2\mu_i,$$

$$r^{k_2k_3} = r^{k_2k_3} - 2\mu_i, r^{k_3k_2} = r^{k_3k_2} - 2\mu_i,$$

$$r^{k_1k_3} = r^{k_1k_3} + \mu_i, r^{k_3k_1} = r^{k_3k_1} + \mu_i.$$

(3) Steps 1–2 are repeated for every rib.

References

[1] Aizenberg, L.: Carleman's formulas in complex analysis: theory and applications. Math. Appl., *244*, Kluwer 1993.

[2] Banfield, J. D., Raftery, A. E.: Ice flow identification in satellite images using mathematical morphology and clustering about principal curves. J. Am. Stat. Assoc. *87*(417), 7–16 (1992).

[3] Born, M., Huang, K.: Dynamical theory of crystal lattices. Oxford: Oxford University Press 1954.

[4] Cai, W., Shao, X., Maigret, B.: Protein-ligand recognition using spherical harmonic molecular surfaces: towards a fast and efficient filter for large virtual throughput screening. J. Mol. Graph. Model *20*(4), 313–28 (2002).

[5] Dergachev, V. A., Gorban, A. N., Rossiev, A. A., Karimova, L. M., Kuandykov, E. B., Makarenko, N. G., Steier, P.: The filling of gaps in geophysical time series by artificial neural networks. Radiocarbon *43*(2A), 365–371 (2001).

[6] Dongarra, J., Lumsdaine, A., Pozo, R., Remington, K.: A sparse matrix library in C++ for high performance architectures. In: Proc. 2nd Object Oriented Numerics Conf., pp. 214–218 (1994).

[7] Elmap: C++ package available online: http://www.ihes.fr/∼zinovyev/vidaexpert/elmap.

[8] Erwin, E., Obermayer, K., Schulten, K.: Self-organizing maps: ordering, convergence properties and energy functions. Biol. Cybern. *67*, 47–55 (1992).

[9] Gorban, A. N. (ed.): Methods of neuroinformatics (in Russian). Krasnoyarsk State University Press, p. 205, 1998.

[10] Gorban, A. N., Karlin, I. V., Zinovyev, A. Yu.: Invariant grids for reaction kinetics. Physica A *333*, 106–154 (2004). Preprint online: http://www.ihes.fr/PREPRINTS/P03/Resu/resu-P03–42.html.

[11] Gorban, A. N., Karlin, I. V., Zinovyev, A. Yu.: Constructive methods of invariant manifolds for kinetic problems. Phys. Reports *396*(4–6), 197–403 (2004). Preprint online: http://arxiv.org/abs/cond-mat/0311017.

[12] Gorban, A. N., Pitenko, A. A., Zinov'ev, A. Y., Wunsch, D. C.: Vizualization of any data using elastic map method. Smart Eng. Syst. Des. *11*, 363–368 (2001).

[13] Gorban, A. N., Rossiev, A. A.: Neural network iterative method of principal curves for data with gaps. J. Comp. Sys. Sci. Int. *38*(5), 825–831 (1999).

[14] Gorban, A., Rossiev, A., Makarenko, N., Kuandykov, Y., Dergachev, V.: Recovering data gaps through neural network methods. Int. J. Geomagnetism Aeronomy *3*(2), 191–197 (2002).

[15] Gorban, A. N., Rossiev, A. A., Wunsch, D. C. II: Neural network modeling of data with gaps: Method of principal curves, Carleman's formula, and other. In: USA–NIS Neurocomputing Opportunities Workshop, Washington, July 1999 (Associated with IJCNN'99). Preprint online: http://arXiv.org/abs/cond-mat/0305508.

[16] Gorban, A. N., Zinovyev, A. Yu.: Visualization of data by method of elastic maps and its applications in genomics, economics and sociology. Preprint of Institut des Hautes Etudes Scientiques, M/01/36, 2001. http://www.ihes.fr/PREPRINTS/M01/Resu/resu-M01-36.html

[17] Gorban, A. N., Zinovyev, A. Yu.: Method of elastic maps and its applications in data visualization and data modeling. Int. J. Comput. Anticipatory Syst. CHAOS *12*, 353–369 (2001).

[18] Gorban, A. N., Zinovyev, A. Yu., Pitenko, A. A.: Visualization of data using method of elastic maps (in Russian). Informatsionnie Technologii *6*, 26–35 (2000).

[19] Gorban, A. N., Zinovyev, A. Yu., Pitenko, A. A.: Visualization of data. Method of elastic maps (in Russian). Neurocomputers *4*, 19–30 (2002).

[20] Gorban, A. N., Zinovyev, A. Yu., Wunsch, D. C.: Application of the method of elastic maps in analysis of genetic texts. In: Proc. Int. Joint Conf. on Neural Networks (IJCNN), Portland, July 20–24, 2003.

[21] Gusev, A.: Finite element mapping for spring network representations of the mechanics of solids. Phys. Rev. Lett. *93*(2), 034302 (2004).
[22] Hastie, T.: Principal curves and surfaces. PhD Thesis, Stanford University, 1984.
[23] Hastie, T., Stuetzle, W.: Principal curves. J. Am. Stat. Assoc. *84*(406), 502–516 (1989).
[24] Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. *43*, 59–69 (1982).
[25] Kégl, B.: Principal curves: learning, design, and applications. PhD Thesis, Concordia University, Canada, 1999.
[26] Kégl, B., Krzyzak, A.: Piecewise linear skeletonization using principal curves. IEEE Trans. Pattern Anal. Machine Intell. *24*(1), 59–74 (2002).
[27] Kégl, B., Krzyzak, A., Linder, T., Zeger, K.: A polygonal line algorithm for constructing principal curves. Neural Inf. Processing Sys. 501–507 (1999).
[28] Kégl, B., Krzyzak, A., Linder, T., Zeger, K.: Learning and design of principal curves. IEEE Trans. Pattern Anal. Machine Intell. *22*(2), 281–297 (2000).
[29] LeBlanc, M., Tibshirani, R.: Adaptive principal surfaces. J. Amer. Stat. Assoc. *89*, 53–64 (1994).
[30] Mandal, C., Qin, H., Vemuri, B. C.: A novel FEM-based dynamic framework for subdivision surfaces. Comp. Aided Des. *32* , 479–497 (2000).
[31] Mulier, F., Cherkassky, V.: Self-organization as an iterative kernel smoothing process. Neural Comput. *7*, 1165–1177 (1995).
[32] Ritter, H., Martinetz, T., Schulten, K.: Neural computation and self-organizing maps: an introduction. Reading, MA: Addison-Wesley 1992.
[33] Roweis, S., Saul, L. K.: Nonlinear dimensionality reduction by locally linear embedding. Science *290*, 2323–2326 (2000).
[34] Sayle, R., Bissell, A.: RasMol: a program for fast realistic rendering of molecular structures with shadows. In: Proc. 10th Eurographics UK'92 Conf., University of Edinburgh, Scotland, 1992.
[35] Stanford, D., Raftery, A. E.: Principal curve clustering with noise. IEEE Trans. Pattern Anal. Machine Intell. *22*(6), 601–609 (2000).
[36] Tenenbaum, J. B., De Silva, V., Langford, J. C.: A global geometric framework for nonlinear dimensionality reduction. Science *290*, 2319–2323 (2000).
[37] Van Gelder, A., Wilhelms, J.: Simulation of elastic membranes and soft tissue with triangulated spring meshes. Technical Report: UCSC-CRL-97–12, 1997.
[38] Verbeek, J. J., Vlassis, N., Krose, B.: A k-segments algorithm for finding principal curves. Technical report, 2000. Online: http://citeseer.nj.nec.com/article/verbeek00ksegments.html.
[39] VidaExpert: Stand-alone application for multidimensional data visualization. Available online: http://www.ihes.fr/∼zinovyev/vidaexpert/vidaexpert.htm.
[40] Xie, H., Qin, H.: A physics-based framework for subdivision surface design with automatic rules control. In: Proc. 10th Pacific Conf. on Computer Graphics and Applications (Pacific Graphics 2002), IEEE Press, 304–315, 2002.
[41] Zinovyev, A.: Visualization of multidimensional data. Krasnoyarsk State University Press Publ., 2000.
[42] Zinovyev, A. Yu., Gorban, A. N., Popova, T. G.: Self-organizing approach for automated gene identification. Open Sys. Inf. Dyn. *10*(4), 321–333 (2003).
[43] Zinovyev, A. Yu., Pitenko, A. A., Popova, T. G.: Practical applications of the method of elastic maps (in Russian). Neurocomputers *4*, 31–39 (2002).

A. Gorban
University of Leicester
University Road
Leicester, LE1 7RH
UK
e-mail: ag153@le.ac.uk

A. Zinovyev
Institut Curie
26, rue d'Ulm
Paris, 75248
France
e-mail: andrey.zinovyev@curie.fr

Verleger: Springer-Verlag GmbH, Sachsenplatz 4–6, 1201 Wien, Austria. – Herausgeber: Prof. Dr. Wolfgang Hackbusch, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22–26, 04103 Leipzig, Germany. – Satz und Umbruch: Scientific Publishing Services (P) Ltd., Chennai, India. – Offsetdruck: Manz Crossmedia, 1051 Wien, Austria. – Verlagsort: Wien. – Herstellungsort: Wien. – Printed in Austria.