Information Visualization (2004) 3, 49–59

www.palgrave-journals.com/ivs

Visualization of high-dimensional data with relational perspective map

James Xinzhi Li[1]

[1]*Edgehill Dr. NW Calgary, Alberta, Canada*

Correspondence:
James X. Li, 252 Edgehill Dr. NW, Calgary, Alberta, Canada T3A 2W8.
Tel: +1 403 547 9630;
E-mail: JamesXLi@VisuMap.net

Abstract
This paper introduces a method called relational perspective map (RPM) to visualize distance information in high-dimensional spaces. Like conventional multidimensional scaling, the RPM algorithm aims to produce proximity preserving 2-dimensional (2-D) maps. The main idea of the RPM algorithm is to simulate a multiparticle system on a closed surface: whereas the repulsive forces between the particles reflect the distance information, the closed surface holds the whole system in balance and prevents the resulting map from degeneracy. A special feature of RPM algorithm is its ability to partition a complex dataset into pieces and map them onto a 2-D space without overlapping. Compared to other multidimensional scaling methods, RPM is able to reveal more local details of complex datasets. This paper demonstrates the properties of RPM maps with four examples and provides extensive comparison to other multidimensional scaling methods, such as Sammon Mapping and Curvilinear Principle Analysis.
Information Visualization (2004) **3,** 49–59. doi:10.1057/palgrave.ivs.9500051

Keywords: multidimensional scaling; dimensionality reduction

Introduction

Many fields in sciences and technologies make extensive use of high-dimensional data, which are presented in the form of numerical tables or abstract distance matrices. Visualizing these data through 2-dimensional (2-D) maps is often an efficient way to discover regularities and extract information. This is especially true when dealing with new types of data where conventional analysis tools such as statistical modeling have difficulties because of lack of initial models. Data visualization, on the other hand, presents data directly to the human eyes, whose extremely high capability of pattern recognition is still unmatched by other analytical techniques.

Whereas the general goal of data visualization is to extract useful information for human eyes, visualization of high-dimensional data particularly faces two main challenges. First, it has to map the dataset onto a 2-D space while preserving distance information of the original dataset as much as possible. Second, it has to partition complex datasets into less complex pieces and present them in a non-overlapping manner so that people can easily discern regularities or irregularities.

In this paper we introduce a method, called the relational perspective map (RPM), which visualizes high-dimensional data through 2-D maps. The RPM method tries to preserve distance information similar to many known dimensionality-reducing mapping methods. But more importantly, the RPM method also shows the ability to visualize data in a non-overlapping manner so that it reveals short-range distance information better than other known mapping methods.

Received: 9 August 2003
Revised: 14 September 2003
Accepted: 15 September 2003

The starting point of the RPM algorithm is a set of abstract data points s_i, $i=1, \ldots, N$ with a distance matrix δ_{ij}, $i,j=1, \ldots, N$. The RPM algorithm maps data points s_i into image points t_i in a 2-D space in such a way that visual distances between the image points, denoted by d_{ij}, resemble the distances δ_{ij}. The image points t_i, $i=, \ldots, N$ therefore provide a 2-D visualization of distance information of the initial dataset. For the sake of our discussion, we call δ_{ij} and d_{ij}, respectively, *relational and image distance matrices*.

As illustrated in Figure 1, the RPM algorithm first maps data points onto the surface of a torus, then the flat rectangle by a vertical and a horizontal cut. The second step is more or less straightforward, thus the RPM algorithm focuses on how to map the dataset onto the torus surface so that the configuration of the image points reflects the distances information in the original dataset.

In order to find an appropriate configuration, the RPM algorithm considers the image points together with the torus as a *force-directed multiparticle system*: the image points are considered as particles that can move freely on the surface of the torus, but can not escape the surface. The particles exert repulsive forces on each other so that, guided by the forces, the particles rearrange themselves to a configuration that visualizes the relational distances δ_{ij}.

In particular, the RPM algorithm uses Eq. (1) as the total potential energy to characterize a configuration.

$$E_p := \sum_{i<j} \frac{\delta_{ij}}{p d_{ij}^p} \text{ with } E_0 := -\sum_{i<j} \delta_{ij} \ln(d_{ij}). \quad (1)$$

The parameter p in Eq. (1) is called the *rigidity* whose value is a real number between -1.0 and $+\infty$. Following the usual physical formalism, the forces between the particles are characterized by

$$f_{ij} := \frac{\partial E_p}{\partial d_{ij}} = -\frac{\delta_{ij}}{d_{ij}^{p+1}}. \quad (2)$$

From a physical point of view, the RPM algorithm simulates the multiparticle system described above by allowing the particles to move along the repulsive forces and therefore minimizes the potential energy. Eq. (2) states that the repulsive force between two image points is proportional to their relational distance, therefore data points with a larger relational distance between them will be mapped to further apart positions on the torus surface. Assuming $p=0$, Eq. (2) also states that the repulsive force between two image points is reversely proportional to the image distance between them. This means that more closely located image points have larger contribution to the total potential energy and therefore carry more information about the original dataset. In general, the parameter p provides a means to control how fast the repulsive force decreases with increasing image distances, and therefore controls how strong the system biases towards information represented by closely located image points.

Related works

In a broad sense, the RPM method can be considered as a technique of multidimensional scaling (MDS[1,2]) that, as a family of techniques, aims to produce multidimensional geometric representation of data. RPM resembles conventional MDS in problem setting, but differs in approach. To put it in a simplified way, compared to the energy function Eq. (1) used in this study, the usual MDS tries to minimize the so-called stress function as described in below

$$E = \sum_{i<j} (\delta_{ij} - d_{ij})^2. \quad (3)$$

By minimizing Eq. (3), the MDS algorithm forces image distance d_{ij} to approach the relational distance δ_{ij}, which is also called dissimilarity or proximity in traditional MDS literatures.

When we consider the optimization process of MDS as a force-directed dynamic system, the image points exert two kinds of forces on each other depending on the image and relation distances: the attractive force when the image distance is larger than the relational distance; or the repulsive force when the image distance is smaller than the relational distances. In contrast to this kind of dual force system, the RPM model only employs repulsive force but resorts to topological means (closed surface) to prevent the system from degeneracy.

A well-known problem of conversional MDS is that the stress function aggregates uniformly over all distances so that the long-range relational distances often dominate

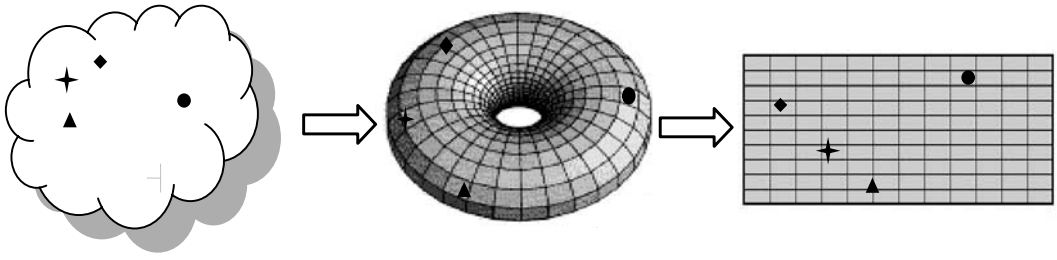

Figure 1 Model of the RPM method.

the global structure of MDS maps. If a dataset is structured in the way that it can not be mapped smoothly to the 2-D plane, the MDS method generally scarifies short-range distance information. This often leads to overlapping of image points in resulted maps as will be demonstrated later in this paper by examples.

Some variations of the stress function have been proposed that reduce the dominating effects of the long-range distances. One of the widely used methods is the Sammon mapping,[3] which uses the Newton–Raphson method (which will be discussed later in more detail) to minimize stress function

$$E = \sum_{i<j} \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}}. \tag{4}$$

The denominator δ_{ij} in Eq. (4) obviously reduces the effect of long-range relational distances. However, this modification is often not sufficient to deal with a complex dataset as will be demonstrated in this paper by examples.

Another approach, the maximum likelihood MDS,[4] proposed the use of function Eq. (5) to limit the effects of long distances by logarithmical transformation.

$$E = \sum_{i<j} [\log(\delta_{ij}) - \log(d_{ij})]^2. \tag{5}$$

The curvilinear component analysis (CCA) algorithm[5] employed a more direct and effective means to reduce the dominating effects of the long-range distances; it extends traditional MDS by using the function Eq. (6) as the stress function.

$$\begin{aligned} E &= \sum_{i<j} (\delta_{ij} - d_{ij})^2 F(d_{ij}, \lambda_t) \text{ with} \\ F(d_{ij}, \lambda_t) &= \begin{cases} 1 & \text{if } d_{ij} \le \lambda_t, \\ 0 & \text{if } d_{ij} > \lambda_t, \end{cases} \end{aligned} \tag{6}$$

where λ_t is a time-dependent parameter that gradually goes from the largest possible image distance to zero during the optimization process. It is obvious that, in terms of force-directed system, the addition factor $F(d_{ij},\lambda_t)$ gradually shrinks the interaction range between image points to zero. The CCA method has been shown to be effective in unfolding data with a complex structure and reveal details in a non-overlapping way. Compared with CCA, RPM seems to be able to carry out more global dataset partition so that local details are better reflected in resulting maps. We will compare CCA with RPM with several examples in this paper.

More recently, the Isomap method[6] and curvilinear distance analysis (CDA)[7,8] have been introduced as nonlinear visualization methods for datasets from high-dimensional space. These two methods extend the traditional MDS by using the geodesic distance instead of the usual Euclidian distance. Intuitively, to apply the geodesic distance, we assume the underlying dataset forms a kind of a geometrical surface (e.g. 2- or 3-D manifolds), and then we define the geodesic distance between two points as the shortest distance following a curve connecting the two points on the surface. Since the geodesic distance is invariant under a large class of deformations, the conversion from Euclidean distance to geodesic distance practically extracts deformation invariant distance information from a dataset. Both Isomap and CDA have been shown to be very effective in visualizing dataset with manifold alike characteristics (e.g. 2-D surface), but for a dataset with higher intrinsic dimensionality or fractal structure they resort, more or less, to the ability of the traditional MDS algorithm. Since the outgoing point of the RPM algorithm is an arbitrary distance matrix, we can apply the RPM algorithm with geodesic distance matrix without modifications.

Most recently, a fast MDS method has been proposed,[9] which combines several hybrid approaches and substantially speedsup the calculation of MDS maps. Although the method aimed to speed up a particular MDS method (which is based on the force-directed model as FDP mentioned below), the elaborate sampling and interpolation techniques might be adapted to the RPM method to achieve higher performance.

The concept of self-organizing map (SOM) developed by T. Kohonen[10] represents a large set of generic mapping methods for high-dimensional data. Based on neural network models, SOM provides an effective tool to cluster a large set of data points and form the so-called self-organizing feature maps. However, unlike MDS, self-organizing feature maps normally do not preserve distance information, but only the topological information, that is the neighboring relationships between the data points.

In practice, the neural network in the SOM algorithm is often structured as a torus instead of a rectangle mesh. This technique aims to avoid possible degeneracy at boundaries. The RPM method shares with SOM in the use of this technique. However, torus plays a more essential role in the RPM algorithm than in SOM: applying RPM algorithm on a rectangle image space would mostly result in strongly degenerated maps.

The method of force-directed placement (FDP)[11] has been developed to construct graphic networks to visualize objects with given relationship structures. The FDP method is based on physical models that make explicit use of attractive and repulsive forces between objects. Similar to the FDP method, we described the RPM algorithm as a force-directed dynamic system, instead of using more commonly used terms such as gradient and stress function as used in some MDS literatures. By doing this, we hope that our description becomes more intuitive.

From a more theoretical aspect, the physical model underlying the RPM algorithm resembles the Thomson Problem,[12] which aims at finding the equilibrium configurations of point charges on a spherical surface. As far as we know, there is no published work that uses this kind of model as a technique to visualize data.

We have adapted the Newton–Raphson (NR) method in the RPM algorithm to minimize the energy function Eq. (1). Like most multivariate optimization problems,

being caught into a local minimum is a main concern in the RPM algorithm. To alleviate this concern, we have used the method of simulated annealing[13] (SA) to verify the minimum found by the RPM algorithm. SA works by simulating the annealing process from material science in which annealing describes the process to eliminate lattice defects in crystals by first heating the crystal and then gradually cooling it to room temperature. Theoretically, if an annealing process is conducted slowly enough, the crystal obtained should have all defects removed. As a general-purpose stochastic optimization method, SA is known as the last-resort method for bad-conditioned optimization problems. Although SA is normally not very efficient, it is often capable of finding good quasi-global optimum. Apart from being used as a reference method in this paper, SA also provides concepts such as temperature and dynamics that offer a different aspect to analyze the RPM optimization process.

The RPM algorithm

Recall that the RPM algorithm aims to minimize the energy function Eq. (1) to find a minimum energy configuration $\{t_i | i = 1, \ldots, N\}$. We have informally stated that t_i are image points of a mapping that maps $\{s_i | i = 1, \ldots, N\}$ on the torus surface. In order to derive a practical algorithm, we first give a more formal specification of the torus and this mapping. Then we will adapt the NR method to find a minimum energy configuration.

Let $T := [0,w] \times [0,h] \subset \mathrm{R}^2$ denotes the rectangle plane of width w and height h in the 2-D Cartesian coordinator system. Let $S := \{s_i | i = 1, \ldots, N\}$ be an abstract set of points. A *torus mapping* is understood as a mapping of the following form:

$$\varphi : S \to T; \quad s_i \mapsto t_i := (x_i, y_i). \tag{7}$$

Next, we define a distance function on T to reflect the torus topology. Let $t_i := (x_i, y_i)$, $t_j := (x_j, y_j)$ be two points from T, the distance between t_i and t_j is defined as follows:

$$\begin{aligned} d(t_i, t_j) := & \min\{|x_i - x_j|, \ w - |x_i - x_j|\} \\ & + \min\{|y_i - y_j|, \ h - |y_i - y_j|\}. \end{aligned} \tag{8}$$

With the above distance function, the opposite edges of the rectangle T are actually stuck together, so that it becomes topologically equivalent to a torus as depicted in Figure 1. Intuitively, d is the city-block distance metric on the torus surface; it defines the distance between two points as the shortest walking distance between the two points while restricting the walking directions to the horizontal or vertical direction.

Having defined T and d, we can now say that the torus surface used in the RPM algorithm is actually the metric space (T, d). The distance d_{ij} in the Eqs. (1) and (2) is actually the abbreviation for $d(t_i, t_j)$. The goal of RPM algorithm is thus to find a torus mapping φ of form (7) that minimizes Eq. (1).

The RPM algorithm adapts the NR method to minimize the energy function Eq. (1). The NR method is a widely used method to find the root of differentiable functions. More detailed discussion about this method can be found, for instance, in.[14] Generally speaking, for a given single-variate function $f(x)$ the NR method suggests the following iterative formula to find a root of the function:

$$x^{(m+1)} = x^{(m)} - \frac{f(x^{(m)})}{f'(x^{(m)})}. \tag{9}$$

An optimum point of $f(x)$ can then be found by finding a root of $f'(x)$ by the following formula:

$$x^{(m+1)} = x^{(m)} - \frac{f'(x^{(m)})}{f''(x^{(m)})}. \tag{10}$$

Thus, in order to apply the NR method to our optimization problem, we need to calculate the first- and second-order partial derivatives of E with respect to all variables x_i and y_i. In the following, we will just consider the calculation of derivatives with respect to x_i; the calculation with respect to y_i is completely analogous. With respect to x_i we have the following equations:

$$\frac{\partial E}{\partial x_i} = \sum_{k<j} \frac{\partial E}{\partial d_{kj}} \frac{\partial d_{kj}}{\partial x_i} = \sum_{k<j} -\frac{\delta_{kj}}{d_{kj}^{p+1}} \frac{\partial d_{kj}}{\partial x_i}. \tag{11}$$

In the above form, the symbol $\sum_{k<j}$ stands for the summation over all $k = 1, \ldots, N$ and $j = 1, \ldots, N$ with $k < j$. Since d_{kj} depends on x_i only if i either equals k or j, we have $\partial d_{kj}/\partial x_i = 0$ for $i \neq j$ and k. Also, since both δ_{ij} and d_{ij} are assumed to be symmetric we have $\delta_{ij} = \delta_{ji}$ and $d_{ij} = d_{ji}$ for all i, j. With these relationships the above equation can be further simplified to

$$\begin{aligned} \frac{\partial E}{\partial x_i} &= \sum_{i<j} -\frac{\delta_{ij}}{d_{ij}^{p+1}} \frac{\partial d_{ij}}{\partial x_i} + \sum_{k<i} -\frac{\delta_{ki}}{d_{ki}^{p+1}} \frac{\partial d_{ki}}{\partial x_i} \\ &= \sum_{k \neq i} -\frac{\delta_{ik}}{d_{ik}^{p+1}} \frac{\partial d_{ik}}{\partial x_k}. \end{aligned} \tag{12}$$

Let $h_{ik} := \partial d_{ik}/\partial x_k$, and let f_{ik} be as been defined in Eq. (2), we have then

$$\frac{\partial E}{\partial x_i} = \sum_{k \neq i} h_{ik} f_{ik}. \tag{13}$$

Referring to Eq. (8), h_{ik} can be calculated as follows:

$$\begin{aligned} h_{ik} = \frac{\partial d_{ik}}{\partial x_i} &= \begin{cases} \partial(|x_i - x_k|)/\partial x_i & \text{if } |x_i - x_k| < w/2, \\ \partial(w - |x_i - x_k|)/\partial x_i & \text{if } |x_i - x_k| > w/2, \end{cases} \\ &= \begin{cases} +1 & \text{if } |x_i - x_k| < w/2 \text{ and } x_i > x_k, \\ -1 & \text{if } |x_i - x_k| < w/2 \text{ and } x_i < x_k, \\ -1 & \text{if } |x_i - x_k| > w/2 \text{ and } x_i > x_k, \\ +1 & \text{if } |x_i - x_k| > w/2 \text{ and } x_i < x_k. \end{cases} \end{aligned} \tag{14}$$

We note that h_{ik} in Eq. (14) is undefined for the case $|x_i - x_k| = w/2$, we will address this issue latter. In plain text, f_{ik} stands for the repulsive force between the image point t_i and t_k; h_{ik} is the operator that bridges the torus topology and the Cartesian representation of the torus: h_{ik} equals +1 if increasing x_i will increase the image distance d_{ik} and h_{ik} equals -1 if increasing x_i will decrease the image distance d_{ik}. More intuitively, Figure 2 illustrates how the operator h works: in the figure, we have three points from the torus surface represented as points

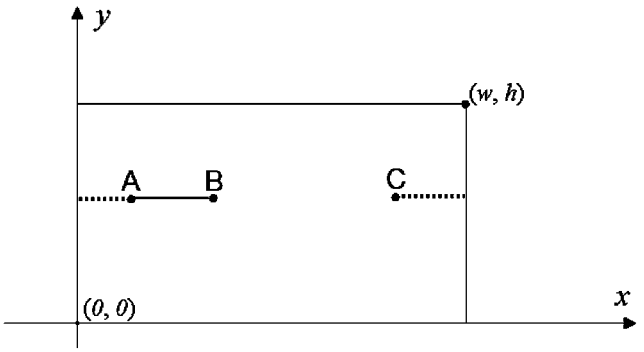

Figure 2 Shortest way between points with respect to the torus topology.

A, B and C in the Cartesian coordinator system. The shortest way from A to B is simply the straight line between them. The shortest way from A to C is, however, the dashed line that wraps around the torus. Thus, if point A moves to the right somewhat, the distance between A and B will decrease, but the distance between A and C will increase. This behavior is characterized by the h operator as $h_{AB}=1$ and $h_{AC}=-1$.

The second-order partial derivative of E is calculated as follows:

$$\begin{aligned}\frac{\partial^2 E}{\partial x_i^2} &= \sum_{k\neq i}\frac{\partial(f_{ik}h_{ik})}{\partial x_i} = \sum_{k\neq i} h_{ik}\frac{\partial f_{ik}}{\partial x_i} + f_{ik}\frac{\partial h_{ik}}{\partial x_i} = \sum_{k\neq i} h_{ik}\frac{\partial f_{ik}}{\partial x_i}\\ &= \sum_{k\neq i} h_{ik}\frac{\partial f_{ik}}{\partial d_{ik}}\frac{\partial d_{ik}}{\partial x_i} = \sum_{k\neq i} h_{ik}(p+1)\frac{\delta_{ik}}{d_{ik}^{p+2}}h_{ik}\\ &= (p+1)\sum_{k\neq i} h_{ik}^2\frac{\delta_{ik}}{d_{ik}^{p+2}}\\ &= (p+1)\sum_{k\neq i}\frac{f_{ik}}{d_{ik}}.\end{aligned} \tag{15}$$

Having worked out the first- and second-order derivatives we can now apply the NR method to our energy function E. Following Eq. (10) we have Eq. (16) as an iterative formula to find the minimum energy configuration.

$$x_i^{(m+1)} = x_i^{(m)} - \frac{\partial E/\partial x}{\partial^2 E/\partial x^2} = x_i^{(m)} - \frac{1}{(p+1)}\frac{\sum_{i\neq k} h_{ik}f_{ik}}{\sum_{k\neq i} f_{ik}/d_{ik}}. \tag{16}$$

For the RPM algorithm, we, however, use a modified variation of the above formula as shown below

$$x_i^{(m+1)} = x_i^{(m)} - c^{(m)}\frac{\sum_{k\neq i} h_{ik}f_{ik}}{\sum_{k\neq i} f_{ik}/d_{ik}}. \tag{17}$$

In Eq. (17), we have only replaced the constant *1/(p+1)* in Eq. (16) with a parameter $c^{(m)}$ that is called the learning speed at step m. $c^{(m)}$ should approach zero as m increases. For our experiments, we used the following simple form to set $c^{(m)}$:

$$c(m) = ra^m, \tag{18}$$

where r is the initial learning speed, a is a number between 0.0 and 1.0. Both r and a are determined empirically. The reason for using a vanishing learning speed is that the standard NR procedure does not converge to a minimum for our problem, but oscillates in an equilibrium state. Generally, the closer a approaches 1.0, the longer the optimization process and the better the resulting minimum energy configuration.

In summary, the RPM algorithm works as follows:

1. Select a mapping φ that maps $\{s_i | i=1, \ldots, N\}$ to randomly selected points $\{(x_i^{(0)}, y_i^{(0)}) \in T | i=1, \ldots, N\}$. Set $m=0$.
2. Calculate $x_i^{(m+1)}$ according to Eq. (17). Calculate $y_i^{(m+1)}$ analogously.
3. Stop the simulation if the total change $\sum_{i=1}^{N}\left|x_i^{(m+1)} - x_i^{(m)}\right| + \left|y_i^{(m+1)} - y_i^{(m)}\right|$ is smaller than a pre-set constant, say 0.0001.
4. Set $m := m+1$; Go to step 2.

In the above description we have, for the sake of simplicity, neglected some issues. The first is that the first-order derivative in Eq. (13) is not defined for the case $|x_i - x_j| = w/2$ as h_{ik} is not defined in this case. In an implementation of the RPM algorithm, we can set h_{ik} to +1 or −1 for the case $|x_i - x_j| = w/2$, either way the resulting minimum energy configuration would not be affected significantly. The reason for this is that the learning speed $c^{(m)}$ approaches zero as the iterative process continues, so that the changes made to the image points will gradually approach zero. On the other hand, the discontinuity of Eq. (13) seems to be a source of instability in the optimization process, and this necessitates the use of the gradually vanishing learning speed $c^{(m)}$.

One main impact of the high instability during the optimization process is that the algorithm lost its second-order convergence rate as it is the case by the standard NR method.[14] The convergence rate of RPM is directly tied to how fast the learning speed $c^{(m)}$ vanishes. On the other hand, the high instability seems to provide necessary dynamics to help the optimization process avoid local minimums. Like the temperature parameter in simulated annealing algorithm, the learning speed has a direct impact on the resulting minimum: the slower the learning speed vanishes, the better the chance to find the global minimum.

Another question that rises naturally is about image distance defined in Eq. (8) which is derived from the city-block distance. Why did we not derive the image distance from the usual Euclidian distance? To answer this question, we considered the following more general parameterized image distance:

$$\begin{aligned}d_r(t_i, t_j) :=& (\min\{|x_i - x_j|,\ w - |x_i - x_j|\}^r\\ &+ \min\{|y_i - y_j|,\ h - |y_i - y_j|\}^r)^{1/r}.\end{aligned} \tag{19}$$

This image distance is adapted from the Minkovsky distance, which becomes the Euclidian distance for $r=2.0$ and becomes city-block distance for $r=1.0$. We

have not adopted the NR method to solve this generalized minimization problem as it will become substantially more complicated. Instead, we used the SA algorithm to solve the minimization problem. Our experiments have found that, in most cases, only when r is close to 1.0 the resulting minimum energy configurations are not degenerated. Thus, the city-block distance not only greatly simplifies the calculation but also seems to be necessary to obtain useful maps. Unfortunately, we do not have a theoretical explanation for this rather surprising phenomenon.

It should also be noted here that the energy function defined in Eq. (1) has a negative value for $p<0.0$. This characteristic obviously contradicts the usual energy concept in physics, but it does not impact the RPM algorithm as the energy function is only used to formalize an optimization problem.

Example 1: the spherical dataset

Our first example is a syntactic dataset that contains about 1000 data points that are evenly distributed on a sphere as depicted in Figure 3a. The relational distances are the Euclidian distances between the points. The main challenge of this dataset is that the spherical structure can not be mapped smoothly to a flat plane in a non-overlapping way. Any non-overlapping map of the spherical dataset must have some kind of discontinuous distortion.

Figure 3b shows the Sammom mapping of the spherical dataset. We note that this map resembles very much the linear projection of Figure 3a and it is basically an overlapping of two discs. This map also shows a typical property of Sammon mappings: two closely related points are mapped to closely located positions but, on the other hand, two far apart points are not necessarily mapped to two distant positions. From a mathematical point of view, Sammon mapping realizes a non-linear, but continuous function.

The CCA map as shown in Figure 3c achieved a non-overlapping map of the spherical surface. Intuitively, during the optimization process the CCA algorithm opened a hole on the spherical surface and then gradually deformed or unfolded it to a flat map. We see that, unlike the Sammon mapping, CCA favors the short-distance information in the way that closely located image points normally correspond to closely located data points. However, distant image points do not always correspond to distant data points and in particular, the image points at boundary suffer quite some distortions.

The RPM map of the spherical dataset is shown in Figure 3d. We note that RPM maps have to be understood as *maps on the torus surface*. This means the opposite edges of the map have to be considered as stuck together. Thus, Figure 3d actually shows two equal-sized discs on the torus surface: one depicted in the center of the map, the other one on the four corners. Intuitively, the RPM algorithm cut the sphere into two equal halves and then

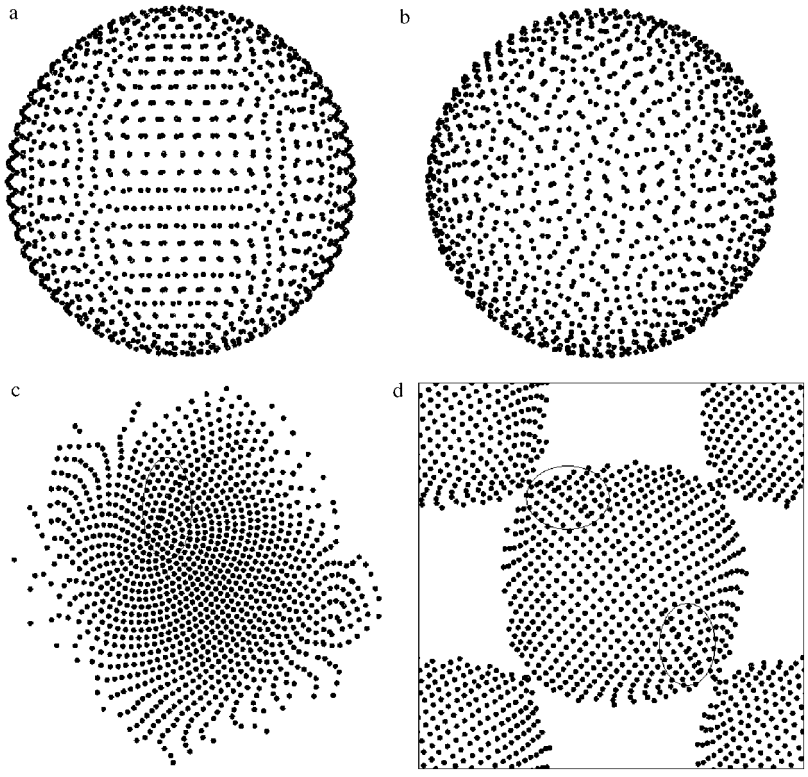

Figure 3 Different maps of the spherical dataset: (a) linear projection; (b) Sammon mapping; (c) CCA map; (d) RPM map.

deformed it into two discs. Compared to the CCA map, the RPM map suffers less local distortion since these image points are more evenly distributed. On the other hand, the RPM map made more radical global distortions as it partitioned the sphere into two halves instead of opening a hole.

With a close look at Figure 3d, we can also notice another interesting capability of the RPM map: the two circled regions show some irregular arrangements of some points. These irregularities are not caused by randomness of the optimization process. They are indeed present in the original dataset! Although this test dataset was generated with the best effort to make the points evenly distributed on the sphere, it was not perfectly done and some points do have irregular distances to their neighbors. These irregularities are illustrated in Figure 4, which shows the linear projection of two subsets of the spherical dataset. This feature demonstrates that RPM is capable of preserving very fine details within a dataset. We notice that the Sammon mapping in Figure 3b failed to reveal such details because of extensive overlapping. The CCA map revealed one irregular region as indicated by the circle in Figure 3c, but failed to reveal the second irregular region because of the extensive distortion occurring in the boundary area.

The fact that the RPM map's opposite edges have to be understood as stuck together is to certain extent contra-intuitive. This drawback can be significantly compensated by interactive exploring support of software programs. The program used in this paper,[15] for instance, implements a tiling technique that tiles multiple instances of the same RPM map together and creates an impression of an endless map. Based on the tiled representation, the user can then easily focus investigation on any sub-regions. Figure 5 depicts one example of such a tiled representation of Figure 3d.

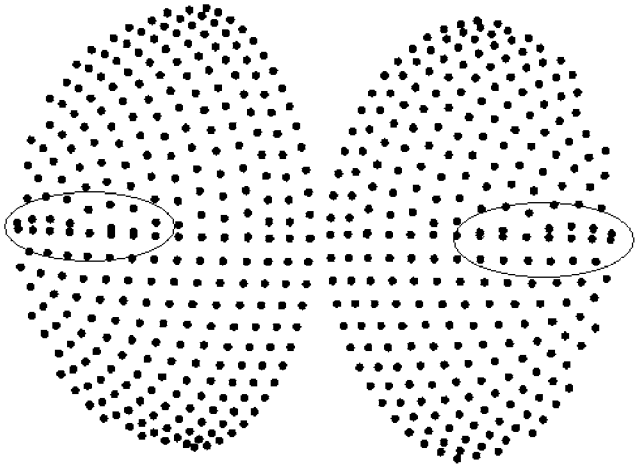

Figure 4 Two irregular regions in the spherical dataset.

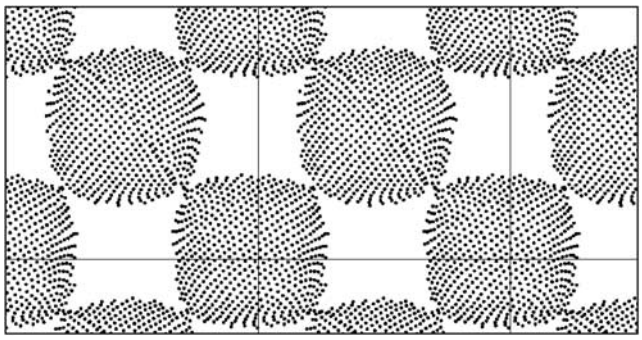

Figure 5 Tiling of multiple instances of the RPM map in Figure 3d.

Example 2: two crossing squares

The next example is also a syntactic dataset; it comprises about 1000 points that form two crossing squares in 3-D space as depicted in Figure 6a. Similar to example 1, this dataset cannot be smoothly mapped to a 2-D plane without overlapping. However it differs from Example 1 in that its geometric shape is not a convex object.

Figure 6b shows the Sammon mapping of the two crossing squares. Similar as in example 1, this map is again basically a linear projection in which the two squares overlap over each other. Figure 6c shows the corresponding CCA map. In this case, the CCA algorithm has fragmented the dataset into several, more or less irregular, pieces and then flattened them to the plane.

The RPM map of this dataset is shown in Figure 6d. The RPM algorithm partitioned the dataset into four major pieces along the intersection line between the two squares, then deformed and combined them together along the intersection line. We note that the RPM map reflects remarkably well the symmetrical structure of the dataset and, partially, the connectivity information along the intersection line.

Example 3: rigidity and fragmentation

Recall that Eq. (2) implies that the rigidity p controls how fast the repulsive force decreases with increasing image distance: larger p causes the repulsive force to decrease more rapidly, and therefore more rapidly reduces the effect of long-range distance. Since long-range distances define the global structure of the dataset, very large p may lead to unnecessary fragmentations of the dataset. To illustrate this phenomenon, we consider in this example a dataset with 400 data points that form two interlaced rings in 3-D space as depicted in Figure 7a. This dataset has been used in[16] as an example of a difficult problem for the CCA algorithm since the dataset can not be embedded into a plane without breaking its topology. However, this mapping problem is not particularly hard for the RPM algorithm as the two rings can be embedded smoothly onto a torus.

Figure 7b shows the RPM map of the two rings created with rigidity $P = 0.0$. The map contains two rings on the

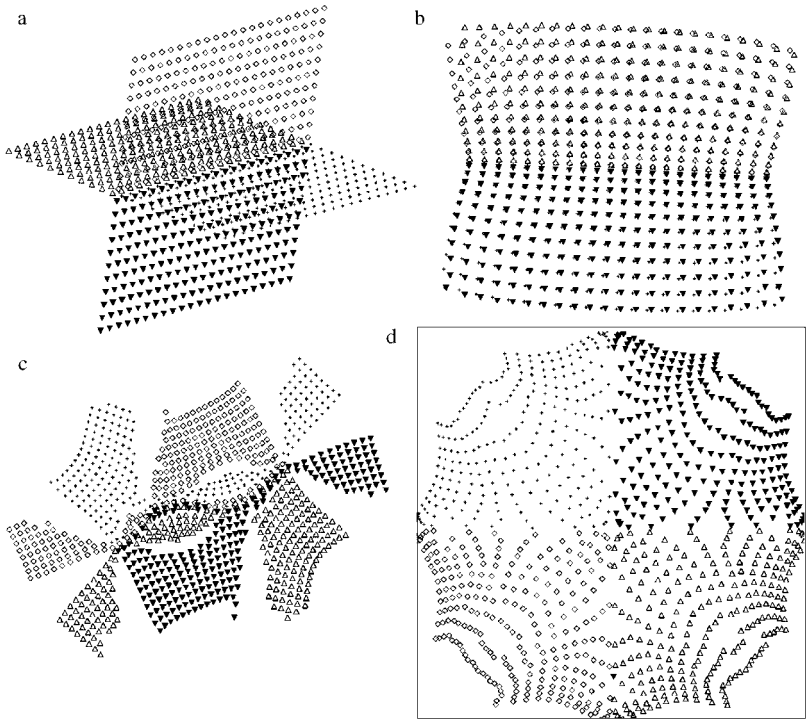

Figure 6 Maps of two crossing squares: (a) simple linear projection; (b) Sammon mapping; (c) CCA map; (d) RPM map.

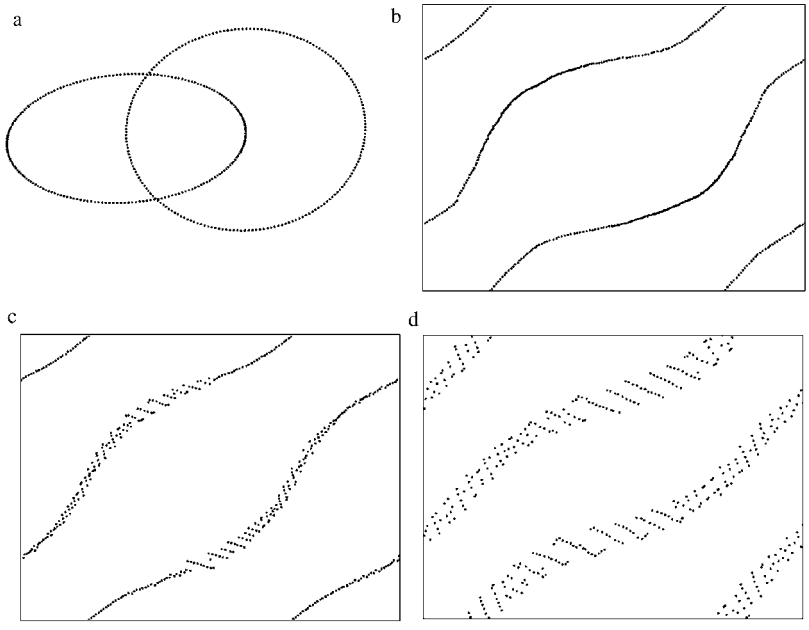

Figure 7 RPM maps of two interlaced rings: (a) linear projection; (b) RPM map with rigidity, $P = 0.0$; (c) RPM map with rigidity, $P = 0.5$; (d) RPM map with rigidity, $P = 1.0$.

torus that correctly reflects the uniform density of the original dataset. For rigidity $P = 0.5$, Figure 7c shows that the two rings have suffered some fragmentation at two regions. For rigidity, $P = 1.0$, we see in Figure 7d that the fragmentation has spread over the whole dataset.

Whereas the rigidity 0.0 is the preferred choice in most cases, in some rare cases higher rigidity seems to be more appropriate as it may lead to a better approach for local structures. On the other hand, if we want to emphasise the long-distance information, that is, the global struc-

ture, we can use low rigidity, for example, −0.5 or even lower values.

In general, we have found during our experiments that, when high rigidity is used, the RPM map tends to be broken unnecessarily into pieces, especially when the simulation is cooled too fast (i.e. $c^{(m)}$ in Eq. (17) vanishes too fast). One measure to avoid these unnecessary fragmentations is to start the simulation process with a low rigidity, say −0.75, and then gradually increase the rigidity to the targeted value. In this way, the RPM algorithm first approaches the global structures, and then gradually changes the emphasis to local structures. According to our experiences, this strategy resulted in better RPM maps in most cases.

Example 4: the S&P 500 stock performance

Our last example is a real-world example. The dataset contains 500 data points and each stands for the stock price histogram of an S&P 500 index company. The relational distances between the data points are calculated as follows: we first obtained the weekly average prices of each stock for the year 2002 from public available sources so that we have an array of 52 numerical values for each stock. We then scaled these arrays individually so that all arrays start from the value 1.0. The relational distances between the stocks are then defined as the Euclidian distance between their scaled arrays.

The mapping problem in this example is thus to map a dataset from the 52-D space to the 2-D space. Figure 8 shows the RPM map generated with rigidity 0.0. The different glyphs for the stocks are to indicate their relative locations in the map. The main property of this map is that closely located stocks should have similar histograms for the year 2002. To show this property, we have plotted histograms of stocks from three selected regions as indicated by three circles from the map: the lower left group represents the extremely bad-performing stocks. The middle group represents the average-performing stocks, where the black dot represents the S&P 500 index itself. The upper-right group represents the stocks which out-performed the market.

For comparison, we created a Sammon mapping for the S&P 500 dataset as depicted in Figure 9. We notice in this map that a large amount of data points are concentrated in the center of the map, whereas a few data points are scattered around and occupy relatively large areas compared to the corresponding RPM map. There could be two reasons for the high concentration in the map's center. First, some stocks are very volatile and therefore have large distances to the rest of the stocks. Sammon mapping algorithm tried to preserve these long-range distances by allocating more space to them. Consequently, less volatile stocks cover less space. For instance, stocks represented by '+'glyph are relatively volatile stocks, and they occupy collectively more space compared to those in the corresponding RPM map. Second, the less volatile stocks have quite similar distances to each other so that they form a kind of high-dimensional simplex. Similar to the spherical dataset, Sammon mapping approaches these data points by projecting them to the same region so that many points overlap each other.

This example demonstrates that, although Sammon mapping might be preferable for capturing long-range distance information and providing high-level overviews, the RPM map is more preferable for revealing details owing to its ability to limit the effects of long-range distances and to partition complex structures.

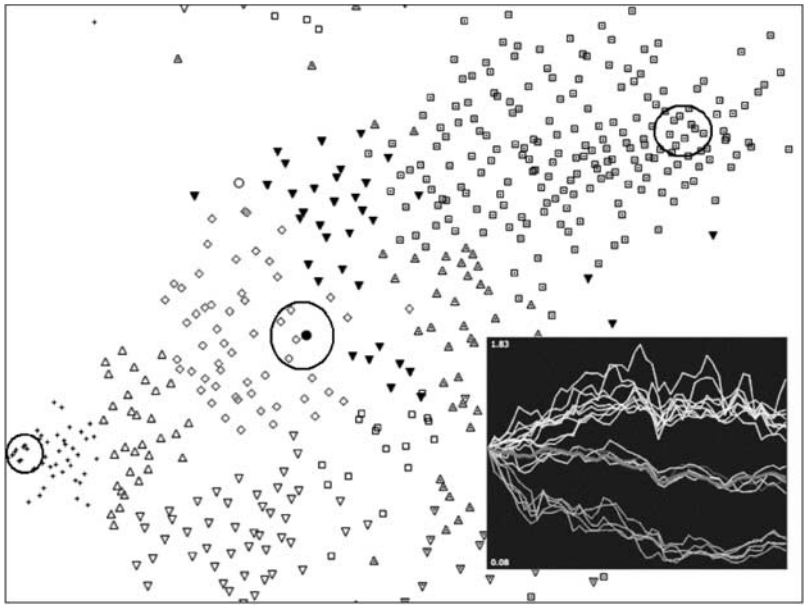

Figure 8 RPM map of the S&P 500 dataset. Each data point represents a stock of the S&P index stock.

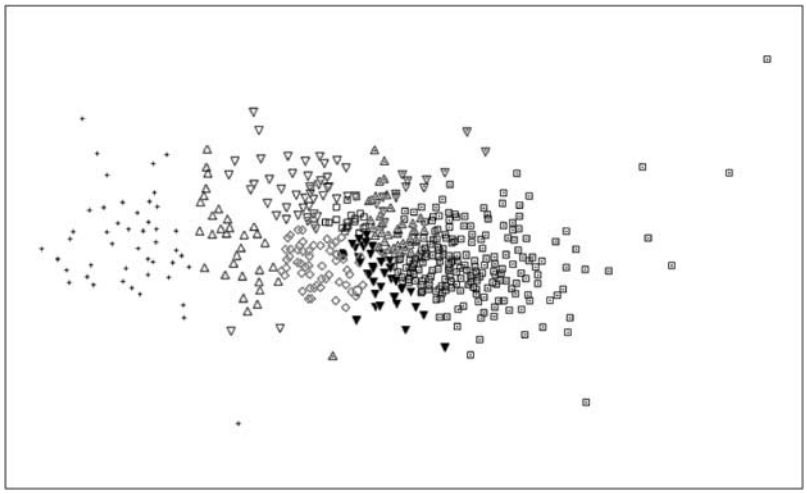

Figure 9 Sammon mapping of the S&P 500 dataset.

General discussion

An important question for a massively multivariate optimization problem is whether a solution is a global optimum or just a local minimum. Does the RPM algorithm always find the global minimum? If not, how far is the solution from the global minimum? We do not have a general answer to this question yet. To obtain some experimental evidence, we used SA as a reference method to solve the same optimization problems and compared the results with those obtained from the RPM algorithm. We found that in most cases, the minimum energy found by the SA algorithm is very close to those found by the RPM algorithm, often within the range of $\pm 0.1\%$. Also, the obtained maps from the two algorithms are visually quite alike. However, the SA algorithm is more than an order of magnitude slower than the RPM algorithm in our tests.

Although the computational speed is not a primary goal of this report, the actual simulation time is of interest for real-world application. We have implemented the algorithm in C# language and run the test on a 1.9 GHz personal computer. For a dataset with 1000 data points, it takes 1–2 min to generate an RPM map. For datasets with 5000 data points, the optimization process can take a couple of hours. In general, since the RPM optimization problem can have up to $O(N^2)$ independent parameters, its computation complexity is expected to be at least $O(N^2)$ for a dataset with N data points. The dynamical behaviors observed during our experiments seem to suggest that there is still large potential for enhancement. Thus, finding a faster optimization algorithm for the RPM method could be a promising task for the future study. Currently, for datasets with more than 10,000 data points, we recommend the use of some clustering methods, such as SOM, to reduce the dataset size before applying RPM algorithm.

The software program used to produce examples in this paper is available free for non-commercial use. This program and more other test datasets can be downloaded from the web site at http://www.visumap.net.

The RPM method as presented in this paper can be used generally where other known MDS methods such as Sammon mapping and CCA can be used. Compared to other known methods, our experiments showed that RPM is more preferable to reveal details in an easily discernable way because of its ability to partition complex dataset and present them in a non-overlapping manner. On the other hand, because of its bias towards short-range distances, RPM normally does not preserve long-range distance information as well as some conventional MDS methods such as Sammon mapping.

In this paper we demonstrated the characteristics of RPM maps mainly through concrete examples. There are many theoretical and practical questions about the RPM map which could be interesting subjects for future investigations. For instance, RPM maps in examples 1 and 2 show some natural partitions of simple structures. Can we find a more formal way to describe such partitions? How can we predict possible partitions under different conditions? How can we validate RPM maps? The distances used in examples of this paper are all Euclidian distances. How does the RPM map behave under other distance metrics?

Furthermore, conceptually RPM algorithm can use any closed surface as its image space. The reason why we chose the torus surface in our study is its simplicity with respect to a conventional computation model. It would be interesting to study RPM maps generated on other closed surfaces such as the sphere or even 3-D manifolds.

Conclusion

The RPM algorithm presented in this paper combines traditional multidimensional scaling method with topological constraints. Compared with other known dimensionality-reducing mapping methods, the RPM algorithm takes a significantly different approach to deal with compromises that often have to be made when approach-

ing both global and local distance information. The RPM algorithm is especially characterized by its ability to achieve a good non-overlapping approximation for local distance information through partitioning. The RPM map therefore offers a new way to explore high-dimensional data.

Acknowledgments

The author wishes to thank Dr. Lida Yang from Shell Canada Ltd. for reviewing the manuscript and for her kind support during the research and the development of the software.

References

1 Cox T, Cox M. *Multidimensional Scaling*. Heinemann: 1994.
2 Borg I, Groenen P. *Modern Multidimensional Scaling*. Spring: 1997.
3 Sammon JW. *A nonlinear mapping algorithm for data structure analysis*. IEEE Transaction on Computers 1969; **C18**: 401–409.
4 Ramsay JO. *Maximum likelihood estimation in multidimensional scaling*. Psychometricka 1977; **42**: 241–266.
5 Demartines P, Hérault J. *Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets*. IEEE Transaction on Neural Networks 1997; **8**: 148–154.
6 Tenenbaum JB, Silva de V, Langford JC. *A global geometric framework for nonlinear dimensionality reduction*. Science 2000; **290**: 2319–2323.
7 Lee JA, Lendasse A, Donckers N, Verleysen M. *A robust nonlinear projection method*. In: Proceedings of ESANN' 2000, 8th European Symposition on Artificial Neural Networks (Bruges, Belgium, 2000), M. D-Facto public, 13–20.
8 Lee JA, Lendasse A, Verleysen B. *Curvlinear distance analysis versus Isomap*. In: Proceedings of European Symposium on Artifical Neural Networks (Bruges, Belgium, 2002); d-side publications, 185–192.
9 Morrison A, Ross C, Chalmers M. *Fast multidimensional scaling, springs and interpolation*. Information Visualization 2003; **2**: 68–77.
10 Kohonen T. *Self-organized formation of topologically correct feature maps*. Biological Cybernetics 1982; **43**: 59–69.
11 Fruchterman TM, Rheingold EM. *Graph drawing by force-directed placement*. Software – Practice and Experience 1991; **21**: 1129–1164.
12 Ashby N, Brittin WE. *Thomson's problem*. American Journal of Physics 1986; **54**: 776–777.
13 Laarhoven van PJM, Aarts EHL. *Simulated Annealing: Theory and Application*. Dordrect: Reidel, 1987.
14 Press WH, Teukolsky SA, Vetterling WT, Flannery BP. *Numerical Recipes in C++ (2nd Edition)*. Cambridge University Press: Cambridge, 2002, 366–372.
15 VisuMap Technologies, VisuMap version 1.4 (standard edition). *The software program used to produce examples in this paper*. This program and associated datasets can be downloaded at http://www.visu-map.net (accessed August 9, 2003).
16 Hérault J, Guérin-Dugué A, Villemain P. *Searching for the embedded manifolds in high-dimensional data, problems and unsolved questions*. European Conference on Artificial Neural Networks (Bruges, Belgium, 2002).