# Two-Phase Mapping for Projecting Massive Data Sets

Fernando V. Paulovich, Cláudio T. Silva, *Senior Member, IEEE*, and L. Gustavo Nonato

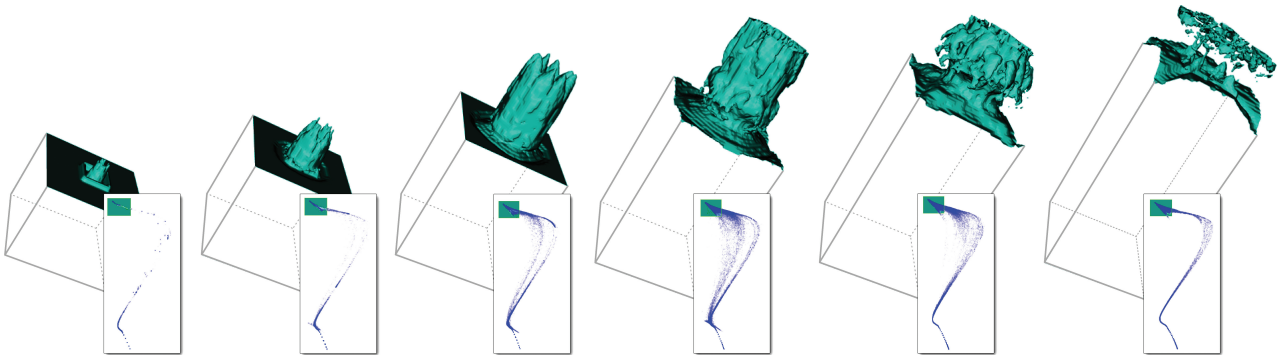


Fig. 1. Large time-varying multivariate volumetric data projection. The information contained within each simulation cell is represented as a tuple in a high-dimensional Cartesian space. The proposed PLMP technique projects the high-dimensional instances into the visual space (bottom-right) in a streaming way. An inspection window (green rectangles) is delimited on the visual space and instances projected inside the inspection window, which correspond to simulation cells with similar features (mainly characterized by high gradient pressure), are rendered in 3D. The streaming projection enables the analysis/visualization of voxels with similar features (projected close to each other in the visual space) on massive time-varying data.

**Abstract**— Most multidimensional projection techniques rely on distance (dissimilarity) information between data instances to embed high-dimensional data into a visual space. When data are endowed with Cartesian coordinates, an extra computational effort is necessary to compute the needed distances, making multidimensional projection prohibitive in applications dealing with interactivity and massive data. The novel multidimensional projection technique proposed in this work, called Part-Linear Multidimensional Projection (PLMP), has been tailored to handle multivariate data represented in Cartesian high-dimensional spaces, requiring only distance information between pairs of representative samples. This characteristic renders PLMP faster than previous methods when processing large data sets while still being competitive in terms of precision. Moreover, knowing the range of variation for data instances in the high-dimensional space, we can make PLMP a truly streaming data projection technique, a trait absent in previous methods.

**Index Terms**—Dimensionality Reduction; Projection Methods; Visual Data Mining; Streaming Technique.

## 1 Introduction

Visualization is currently experiencing a shift from classic approaches, handling small and isolated problems, to more sophisticated frameworks that are able to deal with massive complex data composed of multiple time varying attributes. Many of the new emerging visualization techniques typically represent multivariate data as an $m$-tuple which, in turn, is interpreted as coordinates in an $m$-dimensional Cartesian space. The motivation behind this kind of representation is that techniques such as parallel coordinates [16], scatterplots [12], and projection [32] can be directly applied to visualizing data from its Cartesian coordinates.

Multidimensional projection, in particular, has received significant attention due to its intrinsic ability to construct visual representations that respect proximity between instances of data. More specifically, multidimensional projection methods aim at mapping instances from the Cartesian $m$-dimensional space to a $p$-dimensional visual space, $p = \{2,3\}$, so as to preserve distances as much as possible. In more mathematical terms, projection methods work by minimizing a given energy, such as the normalized stress function:

$$\frac{\sum_{ij}(d_{ij} - \overline{d}_{ij})^2}{\sum_{ij} d_{ij}^2}, \tag{1}$$

which measures how different the distance $\overline{d}_{ij}$ between instances $i$ and $j$ in the $p$-dimensional space is from the original distance $d_{ij}$ in the $m$-dimensional space.

Multidimensional projection is a special case of a wider class of techniques called Multidimensional Scaling (MDS). MDS methods carry out the embedding into a visual $p$-dimensional space by considering distance measures (also called dissimilarities) between pairs of instances, building a visual representation through a minimization procedure that strongly relies on dissimilarities, avoiding explicit use of the Cartesian coordinates in a high-dimensional space. In fact, most multidimensional projection techniques are derived from MDS methods, relying on distance information to embed data into a visual space. Distance computation, however, requires extra computational effort when data is already endowed with a Cartesian representation, which can make multidimensional projection prohibitive in applications demanding interactivity and out-of-core processing of massive data.

This work proposes a novel multidimensional projection technique, called *Part-Linear Multidimensional Projection* (PLMP), which is tailored to handle multivariate data represented in Cartesian $m$-dimensional spaces. In contrast to previous methods, PLMP requires a reduced amount of distance information to carry out the embedding into a visual space, speeding up the projection process substantially. Data instances are projected through a linear mapping built from


- *F. V. Paulovich and L. G. Nonato are with ICMC - Universidade de São Paulo, São Carlos, SP, Brazil. E-mail: {paulovic,gnonato}@icmc.usp.br.*
- *C. T. Silva is with SCI - University of Utah, Salt Lake City, UT, USA. E-mail: csilva@cs.utah.edu.*


*Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.*
*For information on obtaining reprints of this article, please send email to: tvcg@computer.org.*

Cartesian coordinates of representative instances that should be a priori positioned in the visual space. As representative instances are positioned using a nonlinear scheme, the whole projection process is not purely linear, justifying "Part-Linear" in the name of our technique. Distance information might be only required to position representative instances, that is, distances between pairs of representatives are the only ones that might be required. In fact, if representative samples are manually positioned in the visual space then distance information is entirely unnecessary, a property not found in previous methods.

As we show in our extensive comparisons, though PLMP does not exactly minimize the stress function when linearly projecting instances, the precise (non-linear) positioning of representatives and the continuity provided by linear mappings ensure satisfactory projections, better than several MDS methods that rely on pure stress minimization. Moreover, the computational efficiency of PLMP is similar to the fastest existing techniques for small and medium size data sets, but has been shown experimentally to be at least one order of magnitude faster when processing large out-of-core data. The provided effectiveness renders PLMP quite attractive in many applications, such as the streaming projection scheme proposed in Section 6. We show that if the range of variation of each Cartesian coordinate in the high-dimensional space is known a priori then representatives can be manufactured rather than chosen from the data set. This fact used in conjunction with the proposed fast projection scheme enables to project streaming data, as illustrated in Figure 1. To the best of our knowledge this is the first multidimensional projection technique that is able to handle streaming data in the context of visualization, representing an important aspect of this work.

The main contributions of the paper are:

- PLMP: A new multidimensional projection technique that enables the embedding of high-dimensional data instances in a visual space while avoiding extensive computation of distances between data instances (Section 4). The mathematical formulation supporting our technique is novel, and it can be seen as a generalization of Principal Component Analysis (PCA) [18] (see Section 4.3).
- *Extensive Experimental Results and Comparisons*: In this paper, we present a comprehensive set of comparisons that both attests the effectiveness of our methodology and provides a panoramic view on how existing techniques compare to each other. We report results using 16 techniques. We believe that our comparisons, presented in Section 5, can be helpful when deciding which method should be considered for a specific application.
- *Streaming Projection*: In Section 6 we show how our technique is suitable for operating in a streaming mode. The basic that we use is to propose a mechanism to manufacture representative samples that makes PLMP a streaming data projection technique.

## 2 Related Work

Most projection methods have been originally proposed in the context of multidimensional scaling, thus operating from dissimilarities between data instances. In the particular case of projections, dissimilarities are not available and need to be computed. In the following we provide an overview of the main projection techniques, pointing out their computational complexity (given in terms of $n$, the number of instances), optimization mechanisms, and the overhead introduced by distance computations. We group techniques into three main categories: spectral decomposition, nonlinear optimization, and force-based schemes.

**Spectral decomposition:** Techniques based on spectral decomposition, also known as *classic scaling*, typically compute embedding coordinates for each data instance from eigenvectors associated to the larger eigenvalues of a double-centered transformation applied to the dissimilarity matrix (symmetric matrix containing the dissimilarity, or distance, between each pair of data instances). The very first approach proposed by Torgeson [37] made use of a costly $O(n^3)$ singular value decomposition to compute eigenvectors. Computational efficiency has been improved by using sparse dissimilarity matrices, more efficient methods for eigenvector computation, and multiscale matrix representation [3, 20]. However, such approaches still face quadratic complexity in practical applications, impairing their use with massive data.

More efficient approaches have been proposed, for instance, LLE [28] and Isomap [36], whose complexities are close to $O(n)$. LLE generates a sparse matrix from coefficients given by local linear fittings, carrying out the eigendecomposition through efficient numerical methods tailored to solve sparse eigenproblems. Isomap aims at building a distance matrix that approximates "geodesic" distances, carring out the dimensionality reduction from this matrix. LLE and Isomap (and their variants [10]) need $O(bn)$ distances, where $b$ is the maximum number of neighbors for each instance, in order to correctly accomplish local fittings. Distance computation and eigendecomposition negatively affect the performance of those algorithms (see Section 5), since in the worst case they are $O(n^2)$. A more efficient version of Isomap, called L-Isomap (Landmark Isomap), makes use of randomly chosen representative points (landmarks) to reduce the number of distance computation to $O(bkn\log(n))$, where $k$ is the number of landmark points [31].

Subquadratic complexity is achieved by Landmarks MDS [9] and Pivot MDS [4] techniques, which apply classical MDS to only a small subset of $k$ representative instances, projecting the remaining instances through interpolation. Besides being fast, these techniques require dissimilarities between each pair of representatives and between data instances and representatives, that is, $O(k^2+kn)$ dissimilarities have to be computed, thus avoiding the computation of the full dissimilarity matrix. Therefore, the larger the number of representatives the more costly those algorithms become. Our technique also makes use of a subset of representatives but requires dissimilarities only between representatives, having distance computation of $O(k^2)$ complexity.

Fastmap [13] is a technique with truly linear complexity. Although at first glance Fastmap does not seem to compute any eigendecomposition, a more detailed analysis shows its close relation with Nyström approximation of eigenvectors and eigenvalues of a matrix [27]. Fastmap only requires dissimilarities between each instance and two pivot elements per coordinate axis, making its distance computation $O(n)$. Comparisons reported in Section 5 show our technique is similar in performance to Fastmap for in-core data, while presenting a considerable speed-up when handling out-of-core data. Moreover, our technique generates projections with reduced stress.

**Nonlinear optimization:** Nonlinear optimization methods rely on gradient descent schemes to find a minimum for the stress function. First proposed by Kruskal [21], optimization methods usually have complexity above $O(n^2)$ [15, 29], although reasonable performance can be reached by using multigrid-based numerical solvers, as shown by Bronstein et al. [5]. Although convergence can be ensured [8], local minima is still an issue. Most optimization methods are formulated assuming a complete dissimilarity matrix, which adds an $O(n^2)$ computational effort in applications where dissimilarities are not available and have to be computed, such as for Cartesian data.

Aiming to reduce the high computational cost of optimization methods, Pekalska et al. [26] proposed a speed-up mechanism based on representative instances. Their algorithm first embeds a subset of $k$ representative instances using a gradient descent approach and then places the remaining instances using interpolation. In particular, a linear mapping is proposed as an interpolation mechanism, however, in contrast to our method, the linear mapping is computed from dissimilarities between instances and representatives. The computation of dissimilarities makes Pekalska's algorithm computationally costly, as we show in the comparisons.

**Force-based schemes:** Force-based methods are among the simplest multidimensional scaling techniques. The basic idea arose from work by Eades [11], which makes analogy between stress minimization and mass-spring system, defining forces from the residual between original and embedded distances. A naive implementation of a force-based scheme gives $O(n^3)$ complexity. Chalmers [7] proposed a method that makes use of neighborhood structure and representative samples which allows forces to be updated in $O(n)$ per iteration, resulting in a $O(n^2)$ algorithm. Chalmer's algorithm has been further improved,

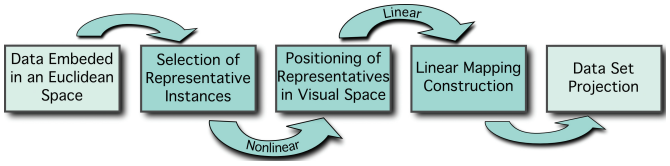



Fig. 2. PLMP pipeline: darker boxes represent the three main steps of the proposed technique.

reaching complexities $O(n^{5/4})$ [23] and $O(n \log n)$ [19]. Multilevel approaches with GPU implementation have also been used to speed-up convergence and handle large data sets [17, 14]. Most of the algorithms described above require the computation of at least $O(n)$ distances in each time step, which is a considerable computational cost.

The simplicity and ease of implementation of force-based methods have motivated their use as a pre- or post-processing mechanism. Tejada et al. [35] employ an heuristic to embed instances in $\mathbb{R}^2$ using a force-scheme as a relaxation mechanism. Paulovich et al. [25] proposed a technique called LSP that uses a force-based scheme to first position a subset of representatives and then map the remaining instances through a Laplace operator. Although LSP presents a good computational performance, it requires at least $O(n)$ distances in order to define neighborhoods from which the Laplace operator is derived. The technique presented in this work also makes use of a force-based scheme to place representatives in the visual space, but, in contrast to methods describe above, only distances among representatives need to be known, rendering PLMP attractive for massive data projection.

## 3 Method Overview

The reasoning behind our technique is that if one knows how to map a subset of instances from $\mathbb{R}^m$ to $\mathbb{R}^p$ (the visual space), $p < m$, then such a priori information can be used to compute a linear transformation $\Phi : \mathbb{R}^m \to \mathbb{R}^p$ to project the whole data set efficiently. In fact, given the linear mapping, the projection of each instance onto $\mathbb{R}^p$ reduces to the product between the matrix representing $\Phi$ and the vector corresponding to the Cartesian coordinates in $\mathbb{R}^m$ of the instance to be projected, an operation that can be performed very efficiently. The quality of the linear mapping essentially depends on the a priori information, that is, the selection of the subset of representative samples and the precision with which these representatives have been placed in $\mathbb{R}^p$. As illustrated in Figure 2, these two aspects are handled independently in our approach (first and second dark boxes from left to right). In fact, the positioning of representative samples corresponds to the non-linear phase of our approach, which is responsible for the good quality of the linear phase. The final step of our approach, highlighted as the third dark box in Figure 2, is the linear mapping computation. The linear mapping is computed by a least-squares approximation from the Cartesian coordinates of representative samples in $\mathbb{R}^m$ as well as $\mathbb{R}^p$. For low-dimensional projection spaces ($p = 2$ or 3 for visualization ends) this computation consists of solving just a few (2 or 3) $m \times m$ symmetric linear systems, which can be done quite efficiently with modern solvers.

A basic premise of our approach is that the computed linear transformation is a good approximation for the non-linear mapping used to project representative samples. Due to the continuity of linear transformations, one should expect that instances close to a representative sample would be mapped close to the projection of that representative in $\mathbb{R}^p$. As we show in Section 5, if representatives are chosen coherently, that rationale is true, resulting in satisfactory projections. Details of each step, beginning with the mathematical foundation behind the linear mapping construction, are presented below.

## 4 The PLMP Technique

Let $H = \{h_1, h_2, \ldots, h_n\}$ be a data set with instances $h_i \in \mathbb{R}^m$. The Part-Linear Multidimensional Projection (PLMP) technique proposed in this work aims at finding a linear transformation $\Phi : \mathbb{R}^m \to \mathbb{R}^p$, $p < m$, which preserves the distance between data instances as much as possible. In more mathematical terms, $\Phi$ should satisfy:

$$\Phi = \underset{\hat{\Phi} \in \mathscr{L}_{m,p}}{\operatorname{argmin}} \left\{ \frac{1}{D} \sum_{ij} \left( d(h_i, h_j) - d(\hat{\Phi}(h_i), \hat{\Phi}(h_j)) \right)^2 \right\} \qquad (2)$$

where $\mathscr{L}_{m,p}$ is the space of linear transformations from $\mathbb{R}^m$ to $\mathbb{R}^p$ and $D = \sum_{ij} d(h_i, h_j)^2$.

Solving the minimization problem (2) directly becomes prohibitive for large values of $n$. We overcome such a hurdle by approximating the linear mapping $\Phi$ rather than solving (2) explicitly. The approximation is carried out using a priori information obtained from representative samples embedded in $\mathbb{R}^p$. More precisely, suppose that a subset $H' = \{h'_1, h'_2, \ldots, h'_k\}$ with $k \ll n$ instances from $H$ has already been mapped to $\mathbb{R}^p$ in a way that distances are preserved as much as possible (the positioning of representative instances in $\mathbb{R}^p$ is discussed in subsection 4.2). Denoting the projection of $h'_i$ in $\mathbb{R}^p$ by $\overline{h}'_i$, the *ideal* linear mapping $\Phi$ minimizing (2) should satisfy:

$$\Phi(h'_i) = \overline{h}'_i \qquad (3)$$

for every $i = 1, \ldots, k$. Equation (3) allows us to compute an approximation for the transformation $\Phi$ by considering the product between each row of the matrix representing $\Phi$ and instances in $H'$. More specifically, consider the product between the first row of $\Phi$ and $h'_i$, $i = 1, \ldots, k$, which result in:

$$\begin{aligned} \phi_{11} x'_{1,1} + \cdots + \phi_{1m} x'_{1,m} &= \overline{x}'_{1,1} \\ \phi_{11} x'_{2,1} + \cdots + \phi_{1m} x'_{2,m} &= \overline{x}'_{2,1} \\ &\vdots \\ \phi_{11} x'_{k,1} + \cdots + \phi_{1m} x'_{k,m} &= \overline{x}'_{k,1} \end{aligned} \qquad (4)$$

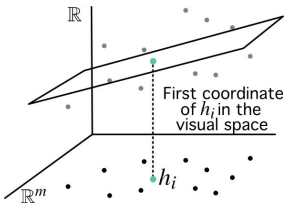
where $\phi_{1i}$, $i = 1, \ldots, m$ are the entries in the first row of the matrix representing $\Phi$, $(x'_{j,1}, \ldots, x'_{j,m})$ are the coordinates of $h'_j$ in $\mathbb{R}^m$ and $\overline{x}'_{j,1}$ is the first coordinate of $\overline{h}'_j$ in $\mathbb{R}^p$. The system of equations (4) gives rise to a linear system $\mathbf{L}\phi = b$, where $\mathbf{L}$ is a $k \times m$ matrix with entries $x'_{j,1}, \ldots, x'_{j,m}$ in the row $j$, $\phi$ is the transpose of the first row of $\Phi$, and $b$ is the vector containing the first coordinate of $\overline{h}'_j$, $j = 1, \ldots, k$.

Assuming that $k$ is larger than $m$ (which is generally the case for large data sets), the first row $\phi$ of $\Phi$ can be approximated by solving the following normal equation:

$$\mathbf{L}^T \mathbf{L} \phi = \mathbf{L}^T b \qquad (5)$$

A complete approximation for $\Phi$ can be obtained by repeating the process above for each line of $\Phi$, that is, $p$ linear systems of size $m \times m$ have to be solved to compute an approximation for $\Phi$. In the context of visualization, $p$ is equal to 2 or 3 and $\Phi$ can efficiently be approximated even for moderate values of $m$ using either fast Cholesky factorization methods devoted to solve the normal equations (5) or the conjugate gradient method. Note that we did not make use of distance information in the mathematical deduction described above, that is, the linear mapping $\Phi$ is approximated from the Cartesian coordinates of representatives. In fact, distances will only be needed to position representatives in $\mathbb{R}^p$, as discussed in subsection 4.2.

In order to gain intuition and better understand the proposed projection method lets analyze the geometrical structure behind the minimization scheme described above. As can be seen from Equation (4), entries in the first row of $\Phi$ can be interpreted as coefficients of a linear mapping that assigns to each point in $\mathbb{R}^m$ a scalar representing its first coordinate in the visual space. The coefficients $\phi_{1j}$ of the linear mapping from $\mathbb{R}^m$ to $\mathbb{R}$ are computed by fitting a hyperplane to the known points $(h'_j, \bar{x}_{j,1}) \in \mathbb{R}^m \times \mathbb{R}$, as illustrated on the left. Therefore, the first coordinate of any instance $h_i$ in the



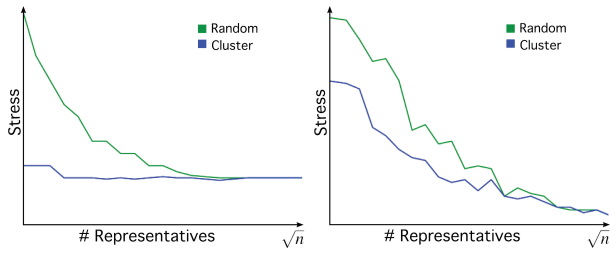



Fig. 3. Comparison between clustering and random sampling schemes for two different data sets. Clustering scheme tends to produce better results only for small number of samples.

visual space is given by the distance between $h_i$ and its image on the hyperplane in $\mathbb{R}^{m+1}$. The same reasoning extends to the other rows of $\Phi$. It becomes clear from the geometrical interpretation that representative instances carry most of the data information to the visual space. Consequently, the quality of the obtained approximation depends basically on two factors: the choice of $H'$ and how precisely instances $\overline{h}'_i$ are placed in $\mathbb{R}^p$ regarding their original distances in $\mathbb{R}^m$. The mechanisms we use to address these aspects are discussed in the following section.

### 4.1 Sample selection

As previously mentioned, the quality of the mapping produced by the PLMP technique depends on the choice of representative instances comprising the subset $H'$. Our implementation uses a random scheme to select representatives and we provide below a justification for such a choice.

A tenable assumption would be to pick out a representative $h'_i$ for each group of instances that characterize a well defined cluster in the original space. Lets focus initially on how to choose a representative $h'$ for a single group of instances $H = \{h_1, \ldots, h_n\}$. A reasonable assumption is to choose $h'$ equidistant from all instances it represents, in mathematical words, $h'$ should minimize the function

$$f(\vec{x}') = \sum_i |\vec{x}' - \vec{x}_i|^2 \tag{6}$$

where $\vec{x}'$ and $\vec{x}_i$ are coordinate vectors of $h'$ and $h_i$ respectively. Setting the gradient of $f$ equal to zero we find that $h'$ is given by the following point:

$$\vec{x}' = \frac{1}{n}\sum_i \vec{x}_i \tag{7}$$

Equation (7) shows us that the best representative for a group of instances embedded into an Euclidean space is not an instance itself, but the average of the instances belonging to that group. Based on this result, we would expect that applying a clustering scheme to the data set and choosing the centroid of each cluster as representatives would produce good results in terms of distance preservation. Surprisingly, this rationale is only partially true in practice, as can be seen in the graphics shown in Figure 3, which depict the resulting stress when one increases the number of representatives using the clustering scheme mentioned above (we use *bisecting k-means* algorithm [33] to build the clusters) and using a random choice of representatives. As one can observe in both examples, the clustering scheme tends to work better for a small number of samples, however, when the sampling rate increases and gets close to $\sqrt{n}$, random and clustering schemes produce very similar results in terms of stress. In fact, we notice this behavior not only in the two examples shown in Figure 3 but in most of the tests we have carried out. Random and cluster-based schemes behave similarly when the number of samples increases as a consequence of the heuristic employed by clustering algorithms (we have tested clustering schemes other than bisecting k-means which gave similar results), which end up evenly distributing clusters (and centroids) in the whole domain, being exactly what is expected of a random sample.

The results discussed above encourage us to use $\sqrt{n}$ random sampling as the mechanism to choose representative samples. However, it is worth noting that uniformly sampling representatives may not always be the best option. For example, if the density of data changes considerably throughout the domain, more precise results should be reached by choosing a larger number of representatives in denser regions of the domain. Estimating the density of data, though, is a difficult and costly task. Since fast projection is one of our main goals, we have opted to use the possibly less accurate but computationally more efficient uniform sampling in all examples presented in this paper.

### 4.2 Positioning samples

Two main aspects should be considered when positioning representative samples in $\mathbb{R}^p$: the number $k$ of samples in $H'$ and the precision in terms of distance preservation. These two aspects can not be considered independently, as the computational cost and accuracy for positioning representative instances depend directly on the number of samples, that is, the larger the number of samples the more costly and inaccurate (due to local minima) their positioning tends to be. A reduced number of representatives, however, can also negatively impact in the final linear projection.

In order to accurately preserve distances during sample positioning we employ the precise non-linear *Force Scheme* [35] technique. As previously mentioned, the Force Scheme has $O(n^2)$ complexity, where $n$ is the total number of instances to be placed in $\mathbb{R}^p$. As discussed in the previous subsection, $k = \sqrt{n}$ randomly chosen representatives yield a good balance between computational cost and quality of the final linear mapping. Moreover, using $\sqrt{n}$ representatives renders the force scheme linear in the number of instances. It is worth pointing out that $k$ is a parameter and, as such, it can be tuned by the user to fit specific applications.

The resulting linear mapping is expected to be in agreement with the organization of representative samples, and any rearrangement of sample instances should directly impact the organization of the final projection. This characteristic can be exploited towards providing user control during projection, that is, by handling control points the user can incorporate some knowledge about the data organization on the final projection. This flexibility is exploited in one of the applications described in Section 6.

### 4.3 PLMP as a generalization of PCA

The matrix $\mathbf{L}^T\mathbf{L}$ in Equation (5) is indeed a covariance matrix, so why not to use Principal Component Analysis (PCA) [18] to approximate $\Phi$ rather than the proposed least-squares scheme? The answer for this question comes from the following analysis.

We can write $\mathbf{L} = \mathbf{UDV}^T$ using the singular value decomposition of $\mathbf{L}$. A simple algebraic manipulation shows that the eigenvectors resulting from PCA should minimize $\|\mathbf{L}\phi - \lambda u\|^2$, where $\lambda$ is a singular value in the diagonal matrix $\mathbf{D}$ and $u$ is a column of the matrix $\mathbf{U}$. On the other hand, the solution of Equation (5) is also a minimizer of $\|\mathbf{L}\phi - b\|^2$, where $b$ is the vector containing the Cartesian coordinates of representatives in the visual space, which can be adjusted, if desired, to be equal $\lambda u$. Therefore, the methodology proposed in this work can be seen as a generalization of PCA, where more flexibility is provided by handling the vector $b$.

Since the energy minimized by PCA is not directly related to the stress function (distances to the fitting subspace are minimized rather than the stress function), PCA is prone to produce projections with high stress and poor visual quality. The nonlinear *Force Scheme* used by PLMP to position representative samples in the visual space introduces the stress minimization component not present in PCA, rendering PLMP a more flexible projection method. In fact, the combination of a nonlinear followed by a linear mapping makes PLMP behave very differently from a PCA-based mapping, resulting in projections with lower stress and better visual quality.

### 4.4 Computational complexity

The computational complexity of PLMP can be computed as $O(R + P + S + I)$, where $R, P, S$, and $I$ are the complexities for choosing repre-

Table 2. Methods employed in comparisons. The first column contains name abbreviations, the second column shows the source reference, the third column contains the computational complexity, and the symbol ✓ in the last column indicates whether the original code has been used rather than our implementation. $n$ and $k$ represent the total number of instances and the number of samples, $c$ is the number of iterations, and $m$ is the dimensionality of the high-dimensional space.

| Name | Source | Complexity | Original |
|---|---|---|---|
| FastMap | Faloutsos and Lin [13] | $O(n)$ | |
| L-MDS | de Silva and Tenenbaum [9] | $O(k^3+kn)$ | ✓ |
| Pekalska | Pekalska et al. [26] | $O(2k^3+kn)$ | |
| Pivot | Brandes and Pich [4] | $O(k^3+k^2+kn)$ | ✓ |
| $\text{Glimmer}_\text{G}$ | Ingram et al. [17] | $O(cn\log(m))$ | ✓ |
| $\text{Glimmer}_\text{C}$ | Ingram et al. [17] | $O(cn\log(m))$ | ✓ |
| LSP | Paulovich et al. [25] | $O(k^2+n^2)$ | ✓ |
| Classical | Torgeson [37] | $O(n^3)$ | |
| Hybrid | Jourdan and Melançon [19] | $O(n\log n)$ | ✓ |
| Force | Tejada et al. [35] | $O(cn^2)$ | ✓ |
| Chalmers | Chalmers [7] | $O(cn)$ | ✓ |
| Smacof | de Leeuw [8] | $O(cn^2)$ | ✓ |
| L-Isomap | Tenenbaum et al. [31] | $O(k^2n)$ | |
| LLE | Roweis and Saul [28] | $O(n^2)$ | |
| Sammon | Sammon [29] | $O(cn^2)$ | |

sentatives, projecting the representatives, solving the system of Equations (5), and projecting the whole data set, respectively. Considering a random choice of representatives, $R = O(1)$. Getting $\sqrt{n}$ representatives and using the *Force Scheme* to position them in $\mathbb{R}^p$, $P = \mathrm{O}(n)$. Since the linear system (5) can be solved using an iterative solver such as Conjugate Gradient, and considering $k = \sqrt{n}$ representatives, we end up with $S = O(k^2) = O(n)$ [30]. Finally, the complexity for projecting the entire data set is $I = O(n)$ ($O(pmn)$ to be more precise), since it only requires the product between the matrix $\Phi$ and the coordinate vector representing each data instance. Therefore, the overall complexity of PLMP is $O(n)$.

## 5 Results and Comparisons

In order to verify the quality of the obtained results we compare the proposed PLMP technique against fifteen existing methods. All the results were generated in an Intel® Core™ i7 CPU 920 2.66GHz, with an NVIDIA® Quadro FX 3800 video card and 8GB of RAM memory. PLMP is implemented in Java, as is the numerical solver – we use the Cholesky factorization available on Java Colt Project (`http://acs.lbl.gov/~hoschek/colt/`). We are using Cholesky rather than Conjugate Gradient (CG) because the linear system (5) has to be solved twice and the already computed Cholesky factorization can be used in the solution of the second system, thus resulting in a performance gain.

Table 2 contains the complete list of compared methods. The technique denoted by "Pekalska" refers to the linear mapping scheme described in [26] and "$\text{Glimmer}_\text{G}$, $\text{Glimmer}_\text{C}$" refers to the GPU and CPU implementation described in [17], respectively. Original codes, kindly provided by the authors, were used for techniques ticked as ✓. We use $\sqrt{n}$ samples when running L-MDS, Pekalska, Pivot, and Hybrid. By using $n$ iterations for Chalmers and Sammon, and 50 iterations for Smacof and Force, we achieve precise projections. With the exception of "$\text{Glimmer}_\text{C/G}$", which are implemented in C++/GPU, all other techniques are purely implemented in Java. All tests were conducted considering the Euclidean distance on the original $m$-dimensional space.

We have employed seven distinct data sets in our experiments, some of them synthetic, permitting us to analyze the different techniques in data sets that vary in size and data dimensionality. The *WDBC* is a breast cancer data set obtained from digitized images of breast masses. Its instances are classified into two distinct groups, the malignant and benign cancer. The *Wine-red* and *Wine-white* sets are related to red and white variants of the Portuguese "Vinho Verde" wine. These include different measures of the wine's feature, which indicate its quality. The *Segmentation* data set is composed of instances randomly drawn from a database of 7 outdoor images. The images were hand-segmented to

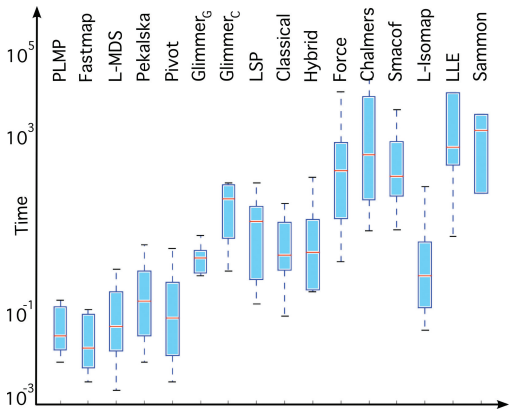



Fig. 4. Boxplot of times (log scale) shown in Table 1

create a classification for every pixel, defining 7 different classes on the data set. *Shuttle* is composed by log information instances split into 7 different classes. The *Mammals* is an artificially generated data set representing different features of mammals belonging to four distinct classes (dogs, cats, horses, and giraffes). Finally, *Explosion* corresponds to a sample of time step 99 of a data containing information from a simulation of an ionization front instability propagation during the formation of a galaxy. The Explosion data set was obtained from the *IEEE Visualization 2008 Contest data set* [38] and the remaining ones were recovered from the *UCI Machine Learning Repository* [2].

Table 1 shows the result of projecting the seven distinct data sets with PLMP and the methods described in Table 2. Numbers in the first column (below the name of each data set) represent the number of instances (left) and the data dimension. For example, the first data set WDBC has 569 instances each one with 30 attributes, that is, each instance is embedded in $\mathbb{R}^3$0. Each entry in Table 1 contains two measures: normalized stress (top), given by Equation (1), and computational time (bottom, highlighted with the letter "s" after the last digit). Empty entries in the table mean that either the corresponding method took longer than three hours to accomplish a projection or memory resources were not enough to load all the data (dissimilarity matrix in most cases) needed to run that method. The sole exception is the empty entry in the $\text{Glimmer}_\text{C}$ column, which is due to a flaw in the code when processing the *wine-red* data set.

Boxplots from data in Table 1 are presented in Figures 4 and 5. Figure 4 shows that PLMP outperforms other methods regarding computational times while still being competitive in terms of stress minimization (see Figure 5). Only Fastmap presents computational times comparable to that of PLMP, however, its stress tends to be worse. Classic and Pivot are designed to minimize a strain function rather than normalized stress while L-Isomap minimizes geodesic distances. In fact, MDS techniques differ in the energy they try to minimize and finding a fair mechanism to compare all of them is not an easy task. Therefore, we follow the trend of other authors [17], using normalized stress as measure of precision for all different techniques.

In addition to enabling a wide range of comparisons with PLMP, Table 1 also provides other interesting information, easily seen from boxplots in Figures 4 and 5. For example, one can notice that among the five fastest techniques (PLMP, Fastmap, L-MDS, Pekalska, and Pivot) Pekalska's method produces the best result in terms of stress, followed by our PLMP approach. The $\text{Glimmer}_\text{C/G}$ techniques gives good stress results, making them a good alternative in applications involving data sets of moderate size and not having fast projections as main requirement. The high computational times of LLE is a consequence of the $O(n^2)$ distance computations used to define neighborhoods and the solver we are using to perform the linear fittings required by LLE. Sub-quadratic methods could be employed to find neighborhoods and a sparse eigenvector solver could be used to accomplish the fittings, the attained poor stress results did not encourage us to incorporate those speed-ups in our implementation.

Table 1. Result of running sixteen projection methods in seven different data sets. The two numbers appearing in each entry correspond to stress (top) and computational time (bottom).

| | PLMP | FastMap | L-MDS | Pekalska | Pivot | $\text{Glimmer}_\text{G}$ | $\text{Glimmer}_\text{C}$ | LSP | Classical | Hybrid | Force | Chalmers | Smacof | L-Isomap | LLE | Sammon |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDBC | 0.0672 | 0.1243 | 0.0603 | 0.0631 | 0.2951 | 0.0603 | 0.0731 | 0.1008 | 0.0558 | 0.1218 | 0.0555 | 0.1980 | 0.1132 | 0.0422 | 0.2674 | 0.0655 |
| $569 \times 30$ | 0.030s | 0.003s | 0.002s | 0.008s | 0.003s | 0.670s | 0.751s | 0.146s | 0.080s | 0.284s | 1.194s | 5.531s | 5.646s | 0.039s | 4.155s | 35.13s |
| Wine-red | 0.1243 | 0.2608 | 0.1302 | 0.1127 | 0.3048 | 0.0853 | | 0.1322 | 0.1000 | 0.1636 | 0.0691 | 0.1092 | 0.2325 | 0.1039 | 0.3199 | 0.0855 |
| $1,599 \times 11$ | 0.008s | 0.006s | 0.014s | 0.030s | 0.011s | 0.779s | | 0.489s | 0.775s | 0.262s | 9.800s | 25.79s | 31.17s | 0.118s | 145.0s | 795.4s |
| Segmentation | 0.0681 | 0.0914 | 0.0491 | 0.0399 | 0.3648 | 0.0366 | 0.0657 | 0.0399 | 0.0461 | 0.0972 | 0.0439 | 0.5181 | 0.1107 | 0.1705 | 0.4006 | 0.0506 |
| $2,100 \times 19$ | 0.015s | 0.010s | 0.015s | 0.049s | 0.021s | 0.576s | 3.777s | 0.498s | 1.596s | 0.445s | 17.78s | 55.31s | 80.03s | 0.168s | 341.2s | 1764s |
| Wine-white | 0.1443 | 0.3940 | 0.1527 | 0.1026 | 0.5251 | 0.0949 | 0.2081 | 0.1527 | 0.1116 | 0.1868 | 0.0747 | 0.0770 | 0.1827 | 0.1351 | 0.6707 | |
| $4,898 \times 11$ | 0.024s | 0.016s | 0.047s | 0.162s | 0.073s | 1.403s | 6.474s | 8.869s | 8.261s | 1.849s | 108.4s | 241.1s | 449.2s | 0.579s | 5097s | |
| Mammals | 0.0380 | 0.0414 | 0.0429 | 0.0305 | 0.3279 | 0.0234 | 0.0445 | 0.0313 | 0.0365 | 0.1087 | 0.0315 | 0.0500 | 0.1055 | 0.0894 | | |
| $10,000 \times 72$ | 0.174s | 0.085s | 0.265s | 0.752s | 0.424s | 4.316s | 26.61s | 18.12s | 20.63s | 9.370s | 445.9s | 4225s | 2183s | 3.122s | | |
| Explosion | 0.0123 | 0.0070 | 0.0044 | 0.0016 | 0.7435 | 0.0014 | 0.0172 | 0.0132 | | 0.4247 | 0.0186 | 0.7194 | | 0.0941 | | |
| $30,000 \times 10$ | 0.126s | 0.110s | 0.812s | 2.723s | 2.250s | 2.060s | 53.25s | 59.28s | | 76.24s | 5460s | 10092s | | 48.74s | | |
| Shuttle | 0.0372 | 0.1156 | 0.0817 | 0.2295 | 2.8926 | 0.0272 | 0.0713 | 0.1187 | | 0.4599 | | | | 0.5209 | | |
| $43,500 \times 9$ | 0.172s | 0.276s | 1.376s | 4.252s | 4.267s | 4.521s | 58.99s | 113.9s | | 177.4s | | | | 21.47s | | |

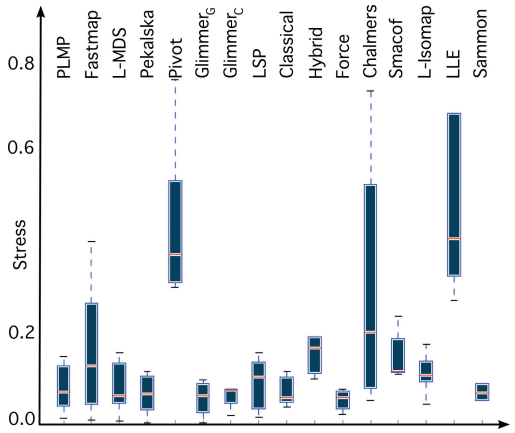



Fig. 5. Boxplot of stress shown in Table 1

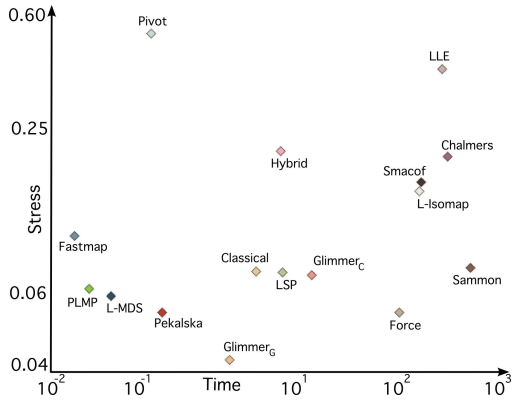



Fig. 6. $\log - \log$ plot of average Time vs. average Stress: the five diamonds closest to the origin represent techniques with the best combined performance of stress minimization and computational times.

Figure 6 shows a $\log - \log$ plot of average time vs. average stress of the data in Table 1. Points on the bottom left correspond to the five techniques with the best performance in terms of computational times as well as stress minimization, namely, PLMP, Fastmap, L-MDS, Pekalska, and $\text{Glimmer}_\text{G}$. Therefore, those five techniques are the only ones able to compete with PLMP in terms of stress minimization and computational times simultaneously. Due to GPU memory constraint, $\text{Glimmer}_\text{G}$ does not scale properly, encouraging us to submit only the first four techniques (PLMP, Fastmap, L-MDS, Pekalska) to a second round of comparisons using larger data sets.

Three distinct data sets have been employed in the second set of comparisons, two are synthetic, called Mammals and Explosion, and one is real, named Fibers. Mammals and Explosion are the same data sets previously mentioned, but with a larger number of instances. Fibers was obtained from the *2009 Pittsburgh Brain Competition (PBC) – Brain Connectivity Challenge* (`http://pbc.lrdc.pitt.edu/`), comprising a set of 250,000 three-dimensional fiber tract pathways, from which a portion is classified into 8 different classes. To calculate the similarity amongst the fibers we transform each one into a multi-dimensional vector composed of 30 coordinates derived from Fourier spectral coefficients of each pathway [6]. Only the magnitude (real part) of the high-frequency coefficients are considered to compose the vectors, since they contain the most information about finer details of the fiber shape. The three data sets were sampled in two different resolutions: 100K and 200K instances, given a total of six data sets. The performance of the four algorithms on these six data sets can be seen in Table 3 and in the boxplots depicted in Figure 7.

As one may notice, PLMP and Fastmap are at least one order of magnitude faster than the other two techniques. Regarding stress minimization, PLMP and Pekalska present the best results. It is worth pointing out that the stress boxplot of Fastmap reflects the high variance w.r.t stress quality presented by that technique. It is clear from Figure 7 that PLMP is the most attractive technique when stress quality and computational efficiency are considered simultaneously.

Figure 8 provides a qualitative visual comparison of the resulting projections from the four techniques. Notice that visual results of PLMP are quite similar to the other three methods, supporting our assertion that PLMP has the best trade-off between speed-up and stress. Moreover, one can see that PLMP was able to separate the classified instances in a satisfactory way (each color corresponds to similar instances classified as ground truth) in most of the cases.

In order to analyze the scalability of our technique, mainly when facing massive data which can not be completely loaded in memory, we implement an out-of-core (OOC) version of our method. More specifically, instead of loading the full data set into memory, only a

Table 3. Comparing the four fastest projection methods. The two numbers appearing in each entry correspond to stress (top) and computational time (bottom).

| | PLMP | FastMap | L-MDS | Pekalska |
|---|---|---|---|---|
| Mammals | 0.0379 | 0.0410 | 0.0357 | 0.0326 |
| $100\text{K} \times 72$ | 0.409s | 0.363s | 4.291s | 10.48s |
| Fibers | 0.0668 | 0.4485 | 0.1517 | 0.1655 |
| $100\text{K} \times 30$ | 1.066s | 0.914s | 11.89s | 20.54s |
| Explosion | 0.0065 | 0.0192 | 0.0037 | 0.0017 |
| $100\text{K} \times 10$ | 0.578s | 0.647s | 6.234s | 13.14s |
| Mammals | 0.0339 | 0.0417 | 0.0474 | 0.0335 |
| $200\text{K} \times 72$ | 2.064s | 3.697s | 64.85s | 72.40s |
| Fibers | 0.0525 | 0.4259 | 0.1459 | 0.0473 |
| $200\text{K} \times 30$ | 1.661s | 1.816s | 36.17s | 58.90s |
| Explosion | 0.0054 | 0.0081 | 0.0068 | 0.0030 |
| $200\text{K} \times 10$ | 1.545s | 1.094s | 24.35s | 58.93s |

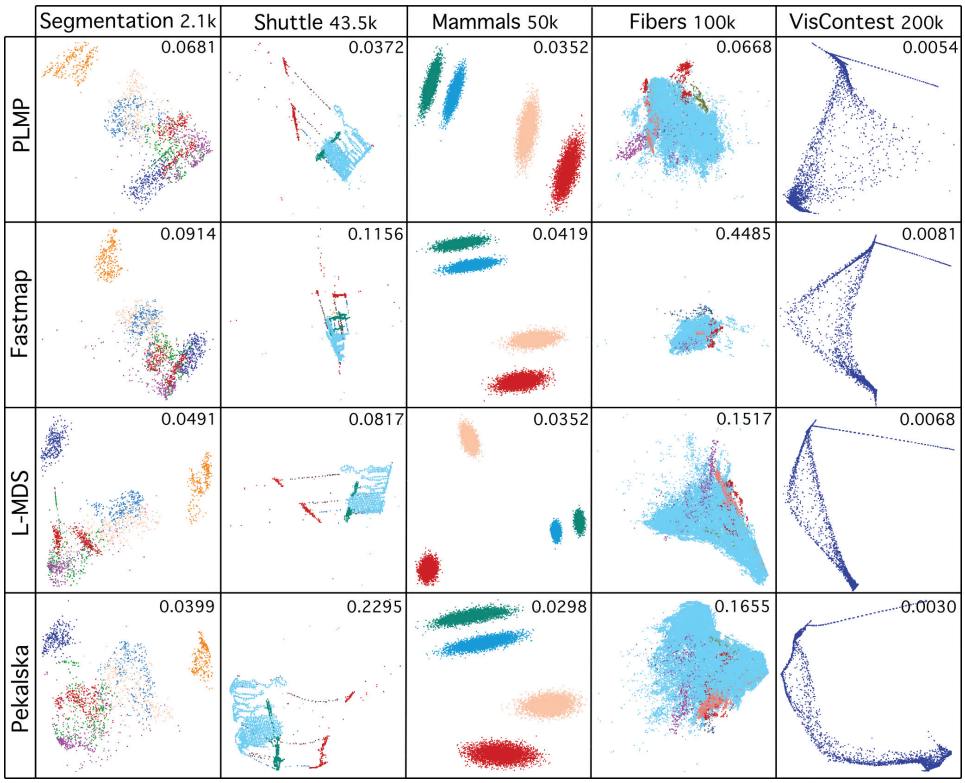



Fig. 8. Projection layouts for five different data sets. The first four data sets are endowed with labels which allow identification of similar instances. Labels are mapped as colors in the plots, providing a qualitative analysis of the projections. Stress is shown in the top right.

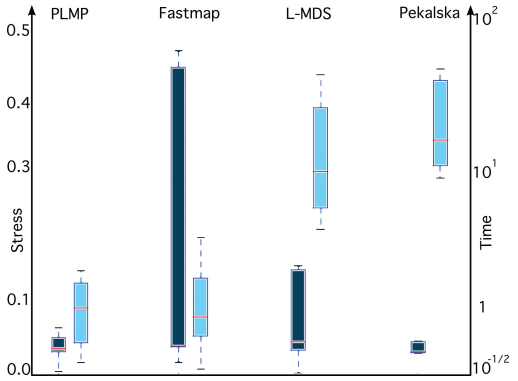



Fig. 7. Boxplot of stress (dark blue) and times in log scale (light blue) for the results shown in Table 3. The vertical axis on left and right shows the scale for stress and times, respectively.

small part is read into a buffer and data is retrieved from the buffer. For the sake of comparison, we also implement an out-of-core version of Fastmap (following the same approach), which is the only technique with similar times to PLMP (see Figure 3). A drawback when implementing the out-of-core version of Fastmap is the need of traversing the whole data set several times to compute pivot elements, which negatively impacts the computational cost. PLMP, though, requires only traversing twice the data set to project the whole data.

Figure 9 (left) shows the resulting computational performance of PLMP and Fastmap as well as their out-of-core versions on Explosion data set sampled in distinct resolutions from 200K to 500K. While computational times of PLMP and Fastmap are practically identical when processing in-core data, Fastmap's performance drops off considerably for OOC data. In fact, OOC-PLMP is at least one order of magnitude faster than OOC-Fastmap. More importantly, the overhead of OOC-PLMP is small, enabling its use in massive data. Projections generated with OOC-PLMP and OOC-Fastmap for a complete time step of the Explosion data set, which contains $36,902,400$ instances, are shown on the right in Figure 9. Transparency is used to give more insight about the concentration of instances in the visual space. OOC-PLMP project the 36.9M instances in $234.16s$ on the same machine previously mentioned with an ordinary disc of 7200rpm, which is a very reasonable processing time for such a huge data set. OOC-Fastmap took $1,757.95s$ to project the same data, an order of magnitude slower than OOC-PLMP.

## 6 Applications

As previously mentioned, the reduced requirement for distance information renders PLMP appropriate for new applications. In the following, two applications are outlined, showing how the nice properties of PLMP can be used to interact and drive projections and how PLMP can be used to project streaming data.

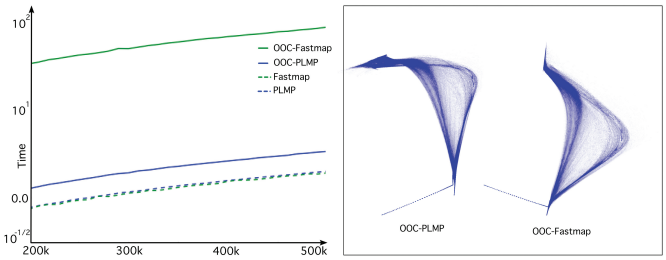



Fig. 9. Comparison between OOC-PLMP and OOC-Fastmap. Left: OOC-PLMP is one order of magnitude faster than OOC-Fastmap while presenting a small overhead when compared with in-core PLMP. Right: OOC-PLMP and OOC-Fastmap projections for a complete time step of Explosion data set with $36,902,400$ instances.

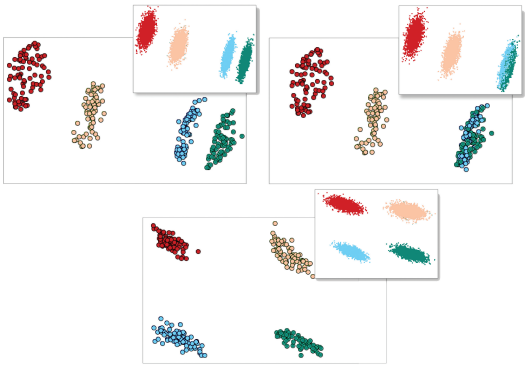


Fig. 10. Different projections of the Mammals data set produced by changing the position of representatives. Each picture shows the position of representatives (main frame) and the final projection of the whole data set (top-right window).

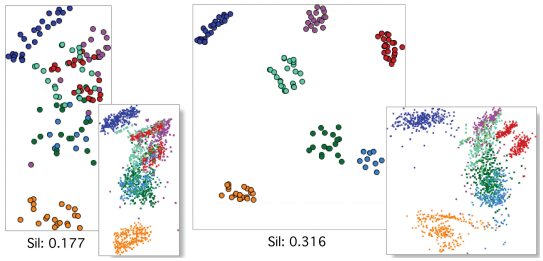



Fig. 11. Different projections of the Segmentation data set. On the left is the projection generated by taking the *Force Scheme* as a mechanism to position representatives. The picture on the right shows the result of manually placing representatives in the visual space. Silhouette coefficients (bottom) quantitatively confirms the group separation improvement resulting from manually positioning representatives.

### 6.1 Handling the Position of Representatives

As discussed in Section 4, PLMP builds the linear mapping from Cartesian coordinates of some representative instances. Therefore, the layout of a projection can be changed, up to an extent, by manipulating representatives in the visual space. Although attempts have already been made toward steering projections [39], the freedom to move representatives renders our methodology more versatile. Figure 10 presents three different projections of the Mammals (10K) data set obtained by changing the position of representatives. Notice that for this data set the final projection, shown on the top-right window, follows the position of representatives exactly. That behavior is always observed when groups are well separated in the high-dimensional space and representatives are chosen for each group.

As one can observe in Figure 11, the manipulation of representative samples can improve the quality of the final layout even when instances are not clearly separated in the original space. In Figure 11, on the left, we show the projection produced by automatically placing representatives using the *Force Scheme*. On the right, representatives were interactively re-positioned and used as the basis for the construction of the linear mapping, improving the separation of groups considerably. The *silhouette coefficient* [34] of each projection, shown at the bottom of the images, confirms the better separation when representatives are "manually" placed in the visual space. The silhouette coefficient, which was originally proposed to evaluate clustering algorithms, measures both the cohesion and separation between grouped instances. Given a data instance $h_i$, its cohesion $a_i$ is calculated as the average of the distances between $h_i$ and all other instances belonging to the same group as $h_i$. The separation $b_i$ is the minimum distance between $h_i$ and all other instances belonging to other groups. The silhouette of a projection is given as the average of the silhouette of all instances and is calculated as $Sil = \frac{1}{n}\sum_{i=1}^{n}\frac{(b_i-a_i)}{max(a_i,b_i)}$, thus ranging in the interval $[-1,1]$. Large values indicate better cohesion and separation between groups of instances.

An important aspect of handling the position of representatives is that knowledge about the data can be used to improve the resulting layout. For example, the user can interact with representatives trying to reach a projection that best matches his/her expectations in terms of group formations.

### 6.2 Streaming Projection

As far as we know, existing projection techniques require for each instance of data (at least) its distance to representative samples, thus enforcing to traverse the entire dataset at least once before starting the projection. PLMP, in contrast, does not require distances to representative instances, therefore, if representatives and their position in the visual space are known a priori, it is possible to build the mapping $\Phi$ and project instances immediately after accessing then, characterizing a streaming projection mechanism. However, to implement a truly streaming version of PLMP we need the representative samples, which until now have been extracted from the original data set. We show in the following that with some knowledge about the data, representatives can be "manufactured" without traversing the original data set, thus allowing PLMP to be used as a streaming projection technique.

Suppose that every instance of a data set $H$ is contained in a hypercube $C = [\alpha_1,\beta_1]\times[\alpha_2,\beta_2]\times\cdots\times[\alpha_m,\beta_m] \subset \mathbb{R}^m$, where the intervals $[\alpha_i,\beta_i]$ are known in advance. The rationale behind our "manufacturing" approach is that if points are randomly drawn inside $C$ then a linear mapping $\Phi$ can be built by considering the random points as representatives. If the number of random points is large enough to ensure that every instance of $H$ has a manufactured representative in its neighborhood, continuity should ensure a good result when mapping $H$ using $\Phi$.Recalling that the *Force scheme* places manufactured samples in the visual space respecting their original distances, the neighborhood of those representatives should also be coherently positioned by $\Phi$, thus producing in a good projection for instances in $H$.

Figure 12 confirms our assumption regarding the manufactured representatives, showing that a drastic loss of quality (in terms of stress) is not introduced when representatives are manufactured rather than picked out from the original data set. Notice that the stress produced by the streaming version of PLMP (representatives chosen inside the hypercube containing all instances) is still better than the ones generated by Fastmap and L-MDS (boxplot on the right). The graphic on the left in Figure 12 shows a comparison of the stress produced by PLMP when representatives are manufactured or taken as samples from the original data. Notice that the stress degradation is quite uniform, confirming that randomly spreading representatives in the hypercube containing the data seems to be an acceptable approach.

For the sake of comparison, we have implemented a random projection scheme [1] which can also project instances without any knowledge about the geometrical structure of the data. We have found that for large data sets PLMP is at least one order of magnitude more accurate than the random projection. The reason is that random projection methods usually rely on the Johnson-Lindenstrauss lemma, which ensures tight error bounds only when the projection space has dimension $O(\varepsilon^{-2}\log(n))$, where $\varepsilon$ is the maximum allowed distortion of the pairwise distances on the projection space and $n$ is the number of data instances. Therefore, no guarantee can be provided when $n$ is large (massive data sets) and the projection space is low-dimensional, as is the case in our applications. Moreover, the non-linear stage of PLMP ensures an accurate positioning for the manufactured representatives, characteristic not present in any random projection scheme.

A better mechanism to measure how the linear mapping $\Phi$ varies when representatives are chosen in the hypercube is the *distortion factor* (DF) of linear transformations [22]. DF is defined as the ratio of the largest singular value to the smallest singular value of $\Phi$ and the closer to 1.0 the better the mapping is. Figure 13 depicts the DF of $\Phi$ for an increasing number of representatives. Notice that in both examples DF converges to a constant value (this behavior was noticed in all tests we have carried out), showing that $\Phi$ tends to have a fixed

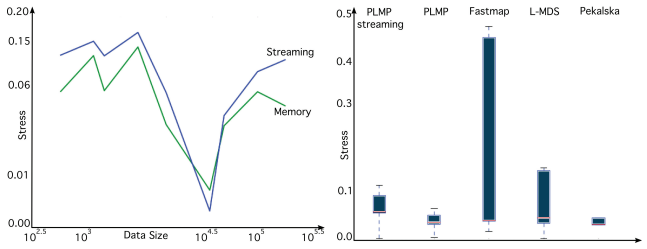



Fig. 12. Left: PLMP resulting stress when representatives are manufactured (blue) and sampled from the original data (green, $\log - \log$ scale). Right: Boxplot of the stress resulting from the streaming implementation compared with techniques shown in Table 3.

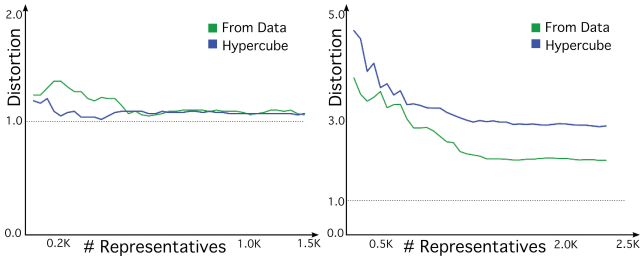



Fig. 13. Distortion of the linear mapping $\Phi$ when representatives are randomly sampled from the data set and randomly drawn in the hypercube containing the data (left: Fibers, right: Explosion ). Distortion converges to a constant when the number of representatives increases.

distortion after considering a sufficient number of representatives.

DF is also an interesting metric to measure how much $\Phi$ distorts neighborhoods. In fact, DF to 1.0 means that $\Phi$ is not substantially "stretching" (or "shrinking") neighborhoods during projection. The example shown on the left of Figure 13 corresponds to the Fibers data set and shows that DF converges to a value close to 1.0. Not coincidentally, the stress produced by PLMP in the Fibers data set (see Table 3) is substantially better than the ones produced by Fastmap and L-MDS.

The distortion factor can also be used as a tool to choose the number of random points to be sampled in the hypercube. In fact, we can progressively increase the number of representatives, computing $\Phi$ and its DF for each set of representatives, until DF converges. Notice that this procedure can be done as a preprocessing step, not introducing any additional cost for the streaming projection itself.

Figure 1 illustrates the potential of the streaming version of PLMP to assist visual data analysis applications involving massive data sets. In that example, instances are continuously projected into the visual space, allowing the user to interactively select regions of interest (the green rectangles in the bottom-right windows) to be further inspected. Simulation cells corresponding to instances falling inside the user delimited regions are rendered in 3D, enabling the visualization of similar data features over time.

## 7 Discussion and Limitations

Results and comparisons shown in Section 5 clearly highlight the effectiveness of PLMP for projecting data embedded in high-dimensional Cartesian spaces. Such a good performance is a consequence of combining the reduced number of distance information required (if any) with the continuity of the linear mapping, which preserves neighborhood relations satisfactorily. Another important characteristic of PLMP is its simplicity in terms of computational implementation, essentially requiring only a numerical solver for the normal equation (5).

The extensive set of comparisons presented in Section 5 provides a panoramic view of existing techniques in terms of computational cost as well as quality of stress minimization. Besides confirming that PLMP presents the best trade-off in terms of computational times and stress quality (see Figure 6), the given comparisons enable an assessment of other methods in practical examples, which is important when deciding on the appropriate technique for a specific application.

Our tests have shown that PLMP is very robust with respect to the choice of representatives, that is, stress measure is barely affected by changing the set of $\sqrt{n}$ randomly chosen representatives (this can also be justified from the distortion coefficient). As pointed out in the streaming application described in Section 6, good results can also be obtained by randomly manufacturing representatives in the data domain. The idea of using manufactured representatives rather than data samples could also be adapted to other methods, however, the need for distances impairs their use in out-of-core data and streaming applications, rendering PLMP quite unique in this context.

The possibility of interactively changing the position of representatives in the visual space is another interesting feature of PLMP. In fact, only the right side of the normal System (5) is affected when representatives are repositioned in the visual space. Therefore, using Cholesky factorization, the linear mapping $\Phi$ can be updated in real time when interacting with representatives, thus enabling a truly interactive projection mechanism for visual data analysis. It is worth noticing that due to the need for distance computation, interactivity can hardly be achieved with other projection methods.

A limitation of PLMP, which is a direct consequence of its strength, is the requirement of having embedded data, that is, in contrast to MDS methods, PLMP can not handle dissimilarity information directly. This characteristic hampers PLMP to be useful in problems where dissimilarity is the only information provided. The need for the number of representatives to be larger than the dimensionality of the data ($k > m$) is another weakness of PLMP. This limitation can impair the use of PLMP in problems involving data embedded in very high-dimensional spaces, such as visual exploration of text collections [24], where data easily reaches thousands of dimensions. The effectiveness of manufacturing representatives inside hypercube containing the data also deserve further investigation. Although good results were obtained in the tests we have carried out, it is important to analyze, for instance, the quality of projections in data sets containing many outliers that increase the size of the hypercube, thus allowing representatives in "void" regions of the space.

## 8 Conclusions

In this work we proposed a novel projection technique called Part-Linear Multidimensional Projection (PLMP), which is shown to be very effective when processing large and complex data sets. PLMP can be seen as a generalization of PCA where a nonlinear stage is introduced so as to provide more flexibility on how data is projected in the visual space. The comprehensive set of comparisons we provided shows that PLMP outperforms existing projection methods with respect to computational times while still being competitive in terms of stress minimization. Moreover, the potential of using PLMP to project streaming data opens new possibilities for applications which could not be addressed until now, such as the visual analysis of data originating from remote sensing or surveillance system. Therefore, flexibility, effectiveness, and ease of implementation render PLMP one of the most attractive projection methods for high-dimensional data embedded in Cartesian spaces. We are investigating the applicability of PLMP as an interactive tool to segment volumes, which is a difficult task to be done with existing tools. The possibility of interactively repositioning representative samples so as to better separate "clusters" is another feature that deserves to be further explored.

## Acknowledgments

We thank Matt Berger for the discussion regarding the generalization of PCA and for his many suggestions that help us to improve the text. We also thank Claurissa Tuttle for her comments and suggestions. This work was supported in part by grants from Fapesp-Brazil and CNPq-NSF. Our research has been funded by the National Science Foundation, the Department of Energy, and IBM Faculty Awards.

## REFERENCES


[1] D. Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

[2] A. Asuncion and D. Newman. UCI machine learning repository, 2007.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, 2003.

[4] U. Brandes and C. Pich. Eigensolver methods for progressive multidimensional scaling of large data. In M. Kaufmann and D. Wagner, editors, *Lecture notes in Computer Science*, volume 4372, pages 42–53. 2007.

[5] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh. Multigrid multidimensional scaling. *Numerical Linear Algebra with Applications*, 13:149–171, 2006.

[6] O. Bruno, L. G. Nonato, M. Pazoti, and J. Batista. Topological multicontour decomposition for image analysis and image retrieval. *Pattern Recognition Letters*, 29:1675–1683, 2008.

[7] M. Chalmers. A linear iteration time layout algorithm for visualising high-dimensional data. In *IEEE Visualization*, pages 127–ff., 1996.

[8] J. de Leeuw. Applications of convex analysis to multidimensional scaling. *Recent Developments in Statistics*, pages 133–146, 1977.

[9] V. de Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. Technical report, Stanford, 2004.

[10] D. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proc. Natl. Acad. Sci.*, 100:5591–5596, 2003.

[11] P. A. Eades. A heuristic for graph drawing. In *Congressus Numerantium*, volume 42, pages 149–160, 1984.

[12] N. Elmqvist, P. Dragicevic, and J.-D. Fekete. Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Trans. Vis. Comp. Graph.*, 14(6):1141–1148, 2008.

[13] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, datamining and visualization of traditional and multimedia databases. In *ACM SIGMOD*, pages 163–174, 1995.

[14] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *IEEE Trans Vis Comput Graph.*, 13:1310–1319., 2007.

[15] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Lecture Notes in Computer Science*, volume 3383, pages 239–250. Springer, 2005.

[16] J. Heinrich and D. Weiskopf. Continuous parallel coordinates. *IEEE Trans. Vis. Comp. Graph.*, 15(6):1531–1538, 2009.

[17] S. Ingram, T. Munzner, and M. Olano. Glimmer: Multilevel mds on the gpu. *IEEE Trans. Vis. Comp. Graph.*, 15(2):249–261, 2009.

[18] I. Jolliffe. *Principal Component Analysis*. Springer, second edition, 2002.

[19] F. Jourdan and G. Melançon. Multiscale hybrid mds. In *Information Visualisation*, pages 388–393, 2004.

[20] Y. Koren, L. Carmel, and D. Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In *IEEE Information Visualization*, page 137, 2002.

[21] J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:115–129, 1964.

[22] C. D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2000.

[23] A. Morrison, G. Ross, and M. Chalmers. A hybrid layout algorithm for sub-quadratic multidimensional scaling. In *IEEE Information Visualization*, page 152, 2002.

[24] F. V. Paulovich and R. Minghim. HiPP: A novel hierarchical point placement strategy and its application to the exploration of document collections. *IEEE Trans. Visual. Comp. Graph.*, 14(6):1229–1236, 2008.

[25] F. V. Paulovich, L. G. Nonato, R. Minghim, and H. Levkowitz. Least square projection: A fast high-precision multidimensional projection technique and its application to document mapping. *IEEE Transactions on Visualization and Computer Graphics*, 14(3):564–575, 2008.

[26] E. Pekalska, D. de Ridder R. P. W. Duin, and M. A. Kraaijveld. A new method of generalizing Sammon mapping with application to algorithm speed-up. In M. Boasson, J. A. Kaandorp, J. F. M. Tonino, and M. G. Vosselman, editors, *Annual Conference of the Advanced School for Computing and Imaging*, pages 221–228, 1999.

[27] J. Platt. Fastmap, metricmap, and landmark mds are all nyström algorithms. In *Intl. Workshop Artificial Intelligence and Statistics*, pages 261–268, 2005.

[28] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000.

[29] J. W. Sammon. A nonlinear mapping for data structure analysis. In *IEEE Transactions on Computers*, volume C-18, pages 401–409, May 1969.

[30] J. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.pdf, 1994.

[31] V. D. Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 15*, pages 705–712. MIT Press, 2003.

[32] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum*, 28(3):831–838, 2009.

[33] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Workshop on Text Mining, ACM SIGKDD International Conference on Data Mining*, pages 109–110, 2000.

[34] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[35] E. Tejada, R. Minghim, and L. G. Nonato. On improved projection techniques to support visual exploration of multidimensional data sets. *Information Visualization*, 2(4):218–231, 2003.

[36] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[37] W. S. Torgeson. Multidimensional scaling of similarity. *Psychometrika*, 30:379–393, 1965.

[38] D. Whalen and M. L. Norman. Competition data set and description. In *2008 IEEE Visualization Design Contest*. http://vis.computer.org/VisWeek2008/vis/contests.html, 2008.

[39] M. Williams and T. Munzner. Steerable, progressive multidimensional scaling. In *INFOVIS '04*, pages 57–64, 2004.