Visualizing and Interacting with Kernelized Data

A. Barbosa, F. V. Paulovich, A. Paiva, S. Goldenstein, F. Petronetto, L. G. Nonato

Abstract—Kernel-based methods have experienced a substantial progress in the last years, tuning out an essential mechanism for data classification, clustering and pattern recognition. The effectiveness of kernel-based techniques, though, depends largely on the capability of the underlying kernel to properly embed data in the feature space associated to the kernel. However, visualizing how a kernel embeds the data in a feature space is not so straightforward, as the embedding map and the feature space are implicitly defined by the kernel. In this work, we present a novel technique to visualize the action of a kernel, that is, how the kernel embeds data into a high-dimensional feature space. The proposed methodology relies on a solid mathematical formulation to map kernelized data onto a visual space. Our approach is faster and more accurate than most existing methods while still allowing interactive manipulation of the projection layout, a game-changing trait that other kernel-based projection techniques do not have.

Index Terms—Multidimensional Projection, Visualization, Kernel Methods

1 Introduction

Kernel methods have emerged as a versatile mechanism to handle generic data. The growing interest in kernels is mainly motivated by the positive impact they have in important applications such as data clustering and classification. Intuitively, a kernel function corresponds to a dot product in a feature space, that is, given a positive definite kernel $k(\cdot,\cdot)$, there exits a map that embeds the data into a feature space where the dot product between instances is given by the kernel k [1]. Typically the embedding map and the feature space associated to a kernel are defined only implicitly, making it difficult to understand how a kernel embeds the data into the underlying feature space. A clear understanding of the mapping performed by a kernel simplifies the choice and the design of kernels as well as the fine-tuning of kernel parameters, thus improving the effectiveness of kernels in specific applications.

Despite the relevance, little has been done towards developing computational tools to assist users in understanding the behavior of kernels. Even more scarce are methods that rely on visualization resources to perform such a task. Techniques such as multidimensional scaling [2] can be used to map kernelized data (data whose similarity is given by a kernel function) to a visual space, but their high computational cost and lack of flexibility as to user interaction have hampered their use as visualization tool to investigate the action of kernels. Multidimensional projection (MP) methods [3] developed in the context of visualization provide more user-friendly interactive mechanisms, but most those methods demand data embedded in a Cartesian space, which prevents their use with kernel functions. The few interactive MP methods able to handle kernelized data are computationally intensive, impairing user experience [4], [5], [6], [7].

This work presents the *Kernel-based Linear Projection* (Kelp), a novel technique able to map data from a kernel defined feature space to a visual space. Kelp relies on a solid mathematical formulation, it has low computational cost and enables interactive resources for users to dynamically interact with the resulting layout. These desirable properties render Kelp an attractive visualization tool in different scenarios. In fact, besides providing a comprehensive set of experiments that confirm the effectiveness of Kelp as a projection technique, we show how Kelp can support visualization tools devoted to handle kernelized data. More specifically, we show how applications such as data classification and image segmentation can benefit from Kelp. Moreover, we derive a kernel-based version of differential coordinates which allows for analysis of change in the neighborhood structure of the original data, due to the action of a kernel. As far as we know, the proposed mechanism is one of the first to enable the visual analysis of how a kernel embeds data onto a feature space.

In summary, the main contributions of this work are:

- A novel kernel-based multidimensional projection technique called *Kelp*, which relies on a solid mathematical formulation to provide a computational efficient visualization for analyzing kernelized data.
- The use of Kelp as a visualization tool to assist kernel-based applications such as data classification and image segmentation.
- The combination of *kernel differential coordinates* (also proposed in this work) with Kelp towards understanding how kernel functions affect neighborhood structures during the embedding process. This novel

- *A. Barbosa, F. V. Paulovich, A. Paiva, L. G. Nonato are with Instituto de Computação e Matemática Computacional, Universidade de São Paulo, São Carlos-SP*
  E-mail: {barbosa, paulovic, apneto, gnonato}@icmc.usp.br
- *S. Goldenstein is with Instituto de Computação, Universidade Estadual de Campinas, Campinas-SP*
  E-mail: siome@ic.unicamp.br
- *F. Petronetto is with Departamento de Matemática, Universidade Federal do Espírito Santo, Vitória-ES*
  E-mail: fabiano.carmo@ufes.br

mechanism is a step forward in enabling visualization resources for users to comprehend the behavior of kernels.

2 Related Work

In order to better contextualize our contribution, we provide an overview of multidimensional scaling and multidimensional projection methods in the context of visualization.

Multidimensional scaling methods (MDS) have long been investigated by the machine learning community as a tool to perform dimensionality reduction. Typically, these methods consider only distance information (dissimilarity measure) between instances to embed data into a Cartesian space. Distinct classes of methods have been proposed to perform the embedding, spectral decomposition being a classical approach. It computes embedding coordinates from the eigenvectors of a transformed version of the dissimilarity matrix (symmetric matrix containing the dissimilarity between each pair of instances) [8], [9], [10]. Spectral methods, in general, are computationally intensive, so they do not scale well to large data sets. In order to alleviate the computational burden, techniques such as Landmark MDS [11], Pivot MDS [12], Fastmap [13] and their variants [14], [15], [16] perform the spectral decomposition only for a subset of samples, projecting the remaining instances based on those samples. Another common characteristic of spectral decomposition methods is the lack of flexibility as to user interaction, which hampers the effective use of those methods in visualization-oriented applications.

First proposed by Kruskal [2], techniques based on nonlinear-optimization comprise other important class of MDS methods, which accomplishes the embedding into a feature space by minimizing an energy function, usually called *stress function*. Those methods are also computationally expensive, even when using efficient numerical solvers [17]. The approach proposed by Chalmers [18] and its variants [19], [20] also rely on subset of samples to reduce the computational burden. GPU implementation has been exploited as an alternative to alleviate computational effort [21], [22], however, this class of methods is still prohibitive for interactive applications that deal with large data sets. Milder computational times can be obtained with the technique proposed by Pekalska et al. [23], which first maps a subset of samples to the visual space by minimizing a stress function and then places the remaining instances using a linear mapping built from the first mapping step.

Multidimensional projection (MP) is a particular class of multidimensional scaling methods where the embedding space is two or three dimensional, targeting mainly data visualization. One of the main characteristics of MP methods is to enable resources that allow users to interact with the projection layout. Least Squares Projection [24] (LSP) and its variant Piecewise Laplacian-based Projection [25] (PLP) are typical examples of MP methods. LSP employs a two-step procedure that first place a subset of sample points onto the visual space and then projects the remaining instances through a Laplacian mapping. The user can steer the projection by manipulating the sample points. Although flexible and capable of projecting data based only on similarity information, LSP does not scale well, as the cost to solve the Laplace systems may become prohibitive for large data sets. The PLP method uses a force-based scheme to place the subset of samples in the visual space. The remaining data instances are projected using local Laplacian maps, which are built from disjoint local neighborhood graphs. MP methods such as Part-Linear Multidimensional Projection [26] (PLMP) and Local Affine Multidimensional Projection [3] (LAMP) among other interactive techniques [27], [28] can handle massive data sets while still ensuring interactive manipulation of the layout. However, PLMP and LAMP rely on Cartesian coordinates of the data to perform the projection onto the visual space, rendering them inadequate for applications where only the similarity between instances is available, as the case of kernelized data.

In summary, most methods able to map data from similarity information either are not flexible enough as to user interaction or do not scale properly to large data sets. Existing interactive and computationally efficient methods can only handle data embedded in a Cartesian space, which considerably restricts their applicability. The Kelp method proposed in this paper fills this gap, since it is computationally efficient, enables interactive manipulation of the projection layouts and it is able to handle kernelized data.

3 Kernel Mapping

The motivation to use kernel functions as measure of similarity between data instances is that kernels provide a way to manipulate data as if it was embedded into a higher dimensional space. The advantage of embedding data in a higher dimensional space is that structures hidden in the data can be untangled in a higher dimension, making operations such as data classification and clustering more reliable. However, the embedding mechanism underlying a kernel function is typically not explicitly defined, making it difficult to understand how data and its neighborhood structure are organized in the embedded space. In order to overcome this issue, we propose a multidimensional projection technique able to handle kernelized data, thus enabling the visualization of how the implicit embedding process underlie kernels affects the arrangement and structure of data instances.

Similarly to most MP methods, the proposed kernel-based projection technique comprises two main steps. In the first step, a subset of samples is mapped to the visual space using an energy minimization scheme. Sample points can also be interactively laid down onto the visual space. The samples may be picked out randomly as proposed in [26] or chosen by the user to better reflect her/his knowledge about the data. Since the subset of samples is typically small (our method is stable even when dealing with a fairly small number of samples), it can be quickly mapped to the visual space even when using a costly energy minimization approach such as the *Force Scheme* [29], which is the method employed in our implementation. The whole data set is mapped to the visual space taken as basis the position of sample points and similarity information between instances given by a kernel function.

The rationale behind our development is to assume initially that we know the embedding map associated to

a given kernel, knowing therefore the image of each instance in the embedding space (feature space). Embedded instances can be mapped to the visual space using a linear transformation which can be computed from the sample points previously mapped. The kernel trick converts the "products" between pairs of embedded instances that show up during the computation of the linear transformation into dot products in the feature space, thus making manageable our assumption about the knowledge of the embedding map.

Before describing the mathematical construction that supports our approach, we provide some basic concepts important in the present context.

3.1 Mathematical Preliminaries

Let $X = \{x_1, x_2, \ldots, x_m\}$ be a set of data instances and $k : X \times X \rightarrow \mathbb{R}$ be a real function that assigns a similarity measure $k(x_i, x_j)$ to each pair of instances x_i and x_j in X. The function k is called a (positive definite) *kernel* if the matrix K with entries $k_{ij} = k(x_i, x_j)$ is positive definite.

Given a kernel as defined above it is possible to construct a map ϕ from X to a (high-dimensional) feature space $\mathcal{H}$ such that

$$k(x_i, x_j) = \phi(x_i)^\top \phi(x_j)\,, \tag{1}$$

where $\phi(x_i)^\top \phi(x_j)$ is a dot product between $\phi(x_i)$ and $\phi(x_j)$. Equation (1) shows that a kernel corresponds to a dot product in a feature space $\mathcal{H}$ (see [1] for a proof), thus the matrix with entries k_{ij} is a Gram matrix.

Assuming for the moment that the image of X by the mapping ϕ is centered in the feature space, that is, $\frac{1}{m}\sum_{i=1}^{m} \phi(x_i) = 0$, the covariance matrix of the mapped data is given by:

$$C = \frac{1}{m}\sum_{i=1}^{m} \phi(x_i)\phi(x_i)^\top\,, \tag{2}$$

where $\phi(x_i)^\top$ is the transpose of $\phi(x_i)$.

Remember that, ϕ is defined implicitly, so the values of $\phi(x_i)$ are unknown. Our formulation makes use of the well known kernel trick to avoid this issue. The technical details can be found in the supplementary material.

A useful property for our formulation is that the eigenvectors $\mathbf{u}_i$ of C can be written as a linear combination of the embedded instances with coefficients given by the eigenvectors of K, more precisely,

$$\mathbf{u}_i = \sum_{j=1}^{m} a_{ij}\phi(x_j)\,, \tag{3}$$

where the vectors $\mathbf{a}_i = (a_{i1}, a_{i2}, \ldots, a_{im})^\top$ are eigenvectors of K (see supplementary material for a proof).

Note that when dealing with kernels, the dimension of the feature space is typically much larger than the number of instances embedded in such space, that is, $m \ll d$. In this scenario, the matrix C has rank at most equal to m, thus each eigenvector associated to a nonzero eigenvalue of C corresponds to an eigenvector of K.

3.2 The Kelp Method

Our kernel-based multidimensional projection method relies on a subset of samples to perform the mapping. Let $X_s \subset X$, $X_s = \{x_{s_1}, x_{s_2}, \ldots, x_{s_n}\}$ be a subset of samples from X (n accounts for the number of samples while m is the total number of instances in X) and $Y_s = \{\mathbf{y}_{s_1}, \mathbf{y}_{s_2}, \ldots, \mathbf{y}_{s_n}\}$ be the image of X_s in the visual space (Y_s results from the Force Scheme applied to X_s). Lets also denote by K_s the Gram matrix built from X_s, that is, the entries in K_s are given by $k(x_{s_i}, x_{s_j})$.

Suppose that the embedding map ϕ associated to the kernel k is known, our goal is to find a linear mapping $M : \mathcal{H} \rightarrow \mathbb{R}^2$ ($\mathbb{R}^2$ being the visual space) such that

$$M\phi(x_{s_i}) = \mathbf{y}_{s_i}\,. \tag{4}$$

The linear transformation M should map each sample $\phi(x_{s_i})$ to $\mathbf{y}_{s_i}$ in the visual space. The rationale behind the construction above is that, due to linearity, the neighborhood structure of each $\phi(x_{s_i})$ should be preserved by M.

Equation (4) can be written in matrix form as

$$M\Phi = Y, \tag{5}$$

where

$$\Phi = \begin{bmatrix} \vdots & & \vdots \\ \phi(x_{s_1}) & \cdots & \phi(x_{s_n}) \\ \vdots & & \vdots \end{bmatrix}, \quad Y = \begin{bmatrix} \vdots & & \vdots \\ \mathbf{y}_{s_1} & \cdots & \mathbf{y}_{s_n} \\ \vdots & & \vdots \end{bmatrix}$$

have dimensions $h \times n$ and $2 \times n$ respectively, and h is the dimension of the $span\{\phi(x_{s_1}), \ldots, \phi(x_{s_n})\}$.

Multiplying both sides of Equation (5) by $\Phi^\top$ we obtain

$$M\Phi\,\Phi^\top = Y\Phi^\top \;\rightarrow\; nMC_s = Y\Phi^\top\,, \tag{6}$$

where C_s is the covariance matrix as defined in Equation (2) but computed from the subset of samples X_s. Since C_s is symmetric it can be decomposed as $C_s = UDU^\top$, where the columns of U are the orthonormal eigenvectors $\mathbf{u}_i$ of C_s and D is a diagonal matrix containing the eigenvalues λ_i as diagonal elements. The pseudo inverse of C_s is given by $C_s^+ = U\tilde{D}^{-1}U^\top$, being $\tilde{D}^{-1}$ the inverse of nonzero diagonal elements in D. Applying the pseudo inverse in Equation (6) results in:

$$M = \frac{1}{n}Y\Phi^\top C_s^+ = \frac{1}{n}Y\Phi^\top\left(U\tilde{D}^{-1}U^\top\right).$$

The projection of any instance $\phi(x)$ is so given by

$$M\phi(x) = \frac{1}{n}Y\Phi^\top U\tilde{D}^{-1}U^\top\phi(x)\,. \tag{7}$$

Let A be the matrix with columns formed by eigenvectors $\mathbf{a}_i$ of K_s (see Equation 3) and, making an abuse of notation, let U be now the matrix containing only the eigenvectors of C_s associated to nonzero eigenvalues. From Equation (3) we can derive

$$U = \Phi A \;\;\Rightarrow\;\; \Phi^\top U = \Phi^\top\Phi A \;\;\Rightarrow\;\; \Phi^\top U = K_s A \tag{8}$$

and

$$U^\top\phi(x) = (\Phi A)^\top\phi(x) = A^\top\Phi^\top\phi(x) = A^\top\mathbf{k}_x\,, \tag{9}$$

where $\mathbf{k}_x = (k(x, x_{s_1}),\; k(x, x_{s_2}), \ldots,\; k(x, x_{s_n}))^\top$.

Using the fact that the eigenvalues of C_s and K_s relate to each other according to $\gamma_i = n\lambda_i$ (see appendix), where γ_i are the eigenvalues of K_s, and using Equations (8) and (9) in Equation (7) we have

$$M\phi(x) = Y K_s A \Gamma^{-1} A^\top \mathbf{k}_x \,, \tag{10}$$

where Γ^{-1} is the diagonal matrix with elements $1/\gamma_i$.

Notice that the term on the right in Equation (10) involves only known quantities. In fact, Y is the matrix containing the coordinates of the samples in the visual space, K_s is the Gram matrix built from X_s, matrix A has columns given by eigenvectors of K_s, diagonal elements in Γ are the inverse of the eigenvalues of K_s, and the vector $\mathbf{k}_x$ is made up of kernel values between x and x_{s_i}, where x is an instance to be projected. Therefore, given the samples, their image in the visual space, and the kernel $k(x, x_{s_i})$, we project any data instance x_i from X to the visual space by simply evaluating Equation (10) in $x = x_i$. In fact, besides Y_s, only $k(\cdot,\cdot)$ need to be known to accomplish the projection of X.

3.3 Centralizing Data in Feature Space

In the previous subsection we assumed the samples X_s centered around the origin (zero mean) in the feature space $\mathcal{H}$. Therefore, we have to center the matrix K_s and the vector $\mathbf{k}_x$ before starting the projection process. The procedure to center a Gram matrix is well known from machine learning literature [1] and consists in applying the following transformation to K_s:

$$\tilde{K}_s = K_s - \mathbb{1}_n K_s - K_s \mathbb{1}_n + \mathbb{1}_n K_s \mathbb{1}_n \,, \tag{11}$$

where $\mathbb{1}_n$ is the square matrix with all entries equal to $1/n$ ($\tilde{K}_s$ will be the matrix used in the projection process).

An important aspect in our approach is that the kernel have to be evaluated only between instances and samples, that is, only $k(x_i, x_{s_j})$ have to be known, that reduces considerably the amount of information to be computed. Nevertheless, the full Gram matrix have to be known to faithfully center each vector $\mathbf{k}_x$, hampering the previously stated advantage of only evaluate $k(x_i, x_{s_j})$. We get around that issue by centering $\mathbf{k}_x$ as to K_s rather than consider the full Gram matrix K. Although such centering mechanism is only an approximation, it worked well in all tests we have carried out. In mathematical terms, the centralization of $\mathbf{k}_x$ is given by:

$$\tilde{\mathbf{k}}_x = \mathbf{k}_x - K_s 1_n - \mathbb{1}_n \mathbf{k}_x + \mathbb{1}_n K_s 1_n \,, \tag{12}$$

where 1_n is a vector with all entries equal to $1/n$. The mathematical justification for Equation (12) can be found in the appendix.

3.4 Projecting the Samples

As already mentioned at the beginning of this section, the subset of samples is mapped to the visual space using the Force Scheme [29]. Since the Force Scheme has been designed to operate on distances rather than similarity measures, we have to convert the kernel information into a distance function. In mathematical terms, a metric can be defined from a dot product in $\mathcal{H}$ by $d(x_i, x_j) = \sqrt{(x_i - x_j)^\top (x_i - x_j)}$, where $x_i,\ x_j \in \mathcal{H}$ and the dot product is the one from the feature space. Expanding the dot product on the right and using the embedding ϕ and the associate kernel k, we get:

$$d(\phi(x_i), \phi(x_j)) = \sqrt{k(x_i, x_i) - 2\,k(x_i, x_j) + k(x_j, x_j)}. \tag{13}$$

The distance function derived from the kernel information is the metric to be preserved by the Force Scheme when arranging the instances Y_s in the visual space. More specifically, the force scheme computes, for each instance y_i in the visual space, the vector $v = y_j - y_i$ and moves y_j in the direction of v by fraction $\Delta = \frac{D(x_i,x_j) - D_{\min}}{D_{\max} - D_{\min}} - d(y_i, y_j)$, D and d are the distances in the original and visual spaces, respectively. Since M is a linear transformation it maps the origin of the feature space to the origin of the visual space. Therefore, we also centralize Y_s after applying the Force Scheme such that the centroid of Y_s coincides with the origin of the visual space.

3.5 Computational Aspects

The mathematical construction in Subsection 3.2 assumes that the eigenvectors $\mathbf{u}_i$ of C are orthogonal and unitary. Using Equation (3), we have

$$1 = \mathbf{u}_i \cdot \mathbf{u}_i = \sum_{l,p=1}^{n} a_{il}\, a_{ip}\, \phi(x_{s_l})^\top \phi(x_{s_p}) = \mathbf{a}_i^\top K_s \mathbf{a}_i = \gamma_i \mathbf{a}_i^\top \mathbf{a}_i$$

that is, the eigenvectors $\mathbf{a}_i$ of K_s as defined in Equation (3) are orthogonal but they are not normalized. However, numerical libraries typically output eigenvectors with norm equal to one. In order to ensure that $\|\mathbf{u}_i\|_2 = 1$, we must multiply the normalized eigenvectors of K_s (given by numerical libraries) by $1/\sqrt{\gamma_i}$. Algorithm 1 summarizes the steps to project each instance $\phi(x_i)$.

Algorithm 1 The Kelp's algorithm.

Require: Data set X and samples X_s

1: Project X_s using the Force Scheme with distances defined in Eq. (13). Compute the mean $\overline{\mathbf{y}} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{y}_{s_j}$ and set the matrix Y as $Y = [\mathbf{y}_{s_1} - \overline{\mathbf{y}}, \mathbf{y}_{s_2} - \overline{\mathbf{y}}, \ldots, \mathbf{y}_{s_n} - \overline{\mathbf{y}}]$

2: Compute the Gram matrix from X_s and centralize it using Eq. (11), obtaining K_s

3: Compute the eigenvectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ and corresponding eigenvalues $\gamma_1, \gamma_2, \ldots, \gamma_n$ of K_s

4: Create the matrix $A_{n\times n} = \left[\frac{\mathbf{a}_1}{\sqrt{\gamma_1}}, \frac{\mathbf{a}_2}{\sqrt{\gamma_2}}, \ldots, \frac{\mathbf{a}_p}{\sqrt{\gamma_n}}\right]$

5: Create the diagonal matrix $\Gamma^{-1}_{p\times p}$ with entries $\Gamma^{-1}_{ii} = \frac{1}{\gamma_i}$

6: Compute the matrix $P = Y\, K_s\, A\, \Gamma^{-1}\, A^\top$

7: **for** each $x \in X$ **do**

8: Compute $\mathbf{k}_x = (k(x, x_{s_1}),\ \ldots,\ k(x, x_{s_k}))^\top$ and centralize it using Eq. (12)

9: Compute the mapping $\mathbf{y} = P\,\mathbf{k}_x$

10: **end for**

The most costly part of Algorithm 1 is the spectral decomposition of matrix K_s, whose complexity is $O(n^3)$, where n is the number of samples. Although costly, the

spectral decomposition is computed only once and requires few samples.

4 Evaluation and Comparisons

Results presented in this section were produced in an Intel® Core™ i7 CPU 920 2.66GHz, with 8GB of RAM. The proposed projection method, Kelp, is implemented in Java using the JBlas numerical library [30] to perform the eigendecomposition of K_s. We use a Gaussian kernel to generate most of the results, we make clear when other kernels are used. The subset of samples used to steer the projection has been chosen randomly and the number of samples is set as $n = \sqrt{m}$, where m is the number of instances in the data set. Some experiments make use of a different choice of samples, which will be clear in the context.

The quality of Kelp is attested through two different sets of comparisons. The first set assesses Kelp's performance as to accuracy and computational time. Kelp is compared against 5 existing techniques employing 8 data sets which vary considerably in terms of size and dimensionality (see Table 1). Techniques employed in the comparisons were chosen because they share similarities with Kelp, namely, they also rely on a subset of samples to perform the projection and can deal with kernelized data. Moreover, those techniques are well known by their good performance in terms of accuracy and/or computational time, ensuring that the provided comparisons are fair and encompass state-of-art projection methods. More specifically, Fastmap [13], Hybrid [19], Landmark MDS [11], Pekalska [23], and PLP [25] are methods that present a good performance in terms of stress/time. Regarding computational implementation, Pekalska and PLP demand linear system solving libraries to be properly implemented. L-MDS demands an efficient implementation of SVD, such as the LAS2 algorithm [34]. Fastmap and Hybrid are straightforward to be coded. Similarly to Kelp, the methods PLP and Pekalska allows for interactive manipulation of samples in the visual space.

Accuracy has been measured based on the *stress function* given by $\frac{1}{\sum_{ij} d_{ij}} \sum_{ij} \frac{(d_{ij} - \overline{d}_{ij})^2}{d_{ij}^2}$, where $d_{ij} = d(\phi(x_i), \phi(x_j))$ is the distance from Equation (13) and $\overline{d}_{ij}$ is the Euclidean distance between instances x_i and x_j (original data) in the visual space.

TABLE 1
Data sets used in the comparisons, from left to right the columns correspond to the data set name, size, dimension (number of attributes), and source.

Name	Size	Dim	Source
wdbc	569	30	[31]
diabetes	768	8	[31]
segmentation	2,100	19	[31]
us-countries	3,028	14	[32]
wine	4,898	11	[31]
letter rcn	20,000	16	[31]
mammals	50,000	72	[31]
viscontest	200,000	10	[33]

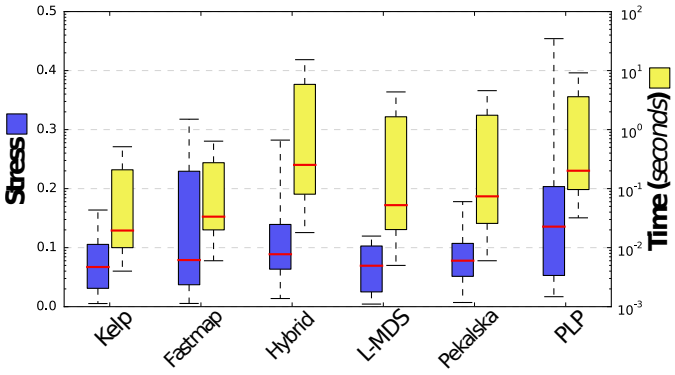


Fig. 1. Box plot of stress and time for data sets in Table 1.

The blue box plots in Figure 1 show the range of stress obtained by Kelp and the other techniques when mapping the data sets in Table 1. One can easily see that Kelp is one of the most accurate technique, being comparable to highly precise methods such as Landmark MDS and Pekalska. Box plots in yellow show that Kelp also performs well in terms of computational times, being comparable to Fastmap, which is well known for its computational efficiency. Notice that Kelp is almost one order of magnitude faster than Landmark MDS and Pekalska, the two methods comparable to Kelp in terms of accuracy.

The *original-distance* $\times$ *projected-distance* scatter plots presented in Figure 2 allow for assessing Kelp's accuracy visually. Notice that Kelp gives rise to nearly $45°$ diagonal layout in almost all test cases, attesting that neighborhoods are well preserved in the visual space. The same is not true for other projection methods such as Hybrid and PLP, which result in a spread distribution around the diagonal direction.

Data sets used in the comparisons above are endowed with instances embedded in a vector space, which allows for employing highly accurate techniques such as LAMP [3] to project the data. Therefore, one could surmise that the proposed kernel-based method is useless for this kind of data. Figure 3 contradicts such reasoning, showing that the projection resulting from a kernel has better defined clusters than the projection generated by mapping the data directly from its intrinsic feature space. Figures 3(a) and 3(b) show the projections resulting from applying Kelp and LAMP to map the Segmentation data (see Table 1) and Figures 3(c) and 3(d) the results of applying Kelp and LAMP to a data set with 574 scientific articles collected on three different subjects [24]. Projections in Figures 3(a) and 3(c) have been produced by kernelizing the original data as

$$k(x_i, x_j) = \exp(-d_{ij}^2 / 2\sigma^2)\,,$$

where d_{ij} is the Euclidean distance $d(x_i, x_j)$ computed from the corresponding bag-of-words and σ is the average variance of the data. Figures 3(b) and 3(d) show the projection of the same data sets but using LAMP (the bag-of-words of each instance is used to perform the projection). Notice that the layout generated by Kelp is less tangled, showing up clusters and similar instances. The better quality of the layout resulting from Kelp is quantitatively attested by the

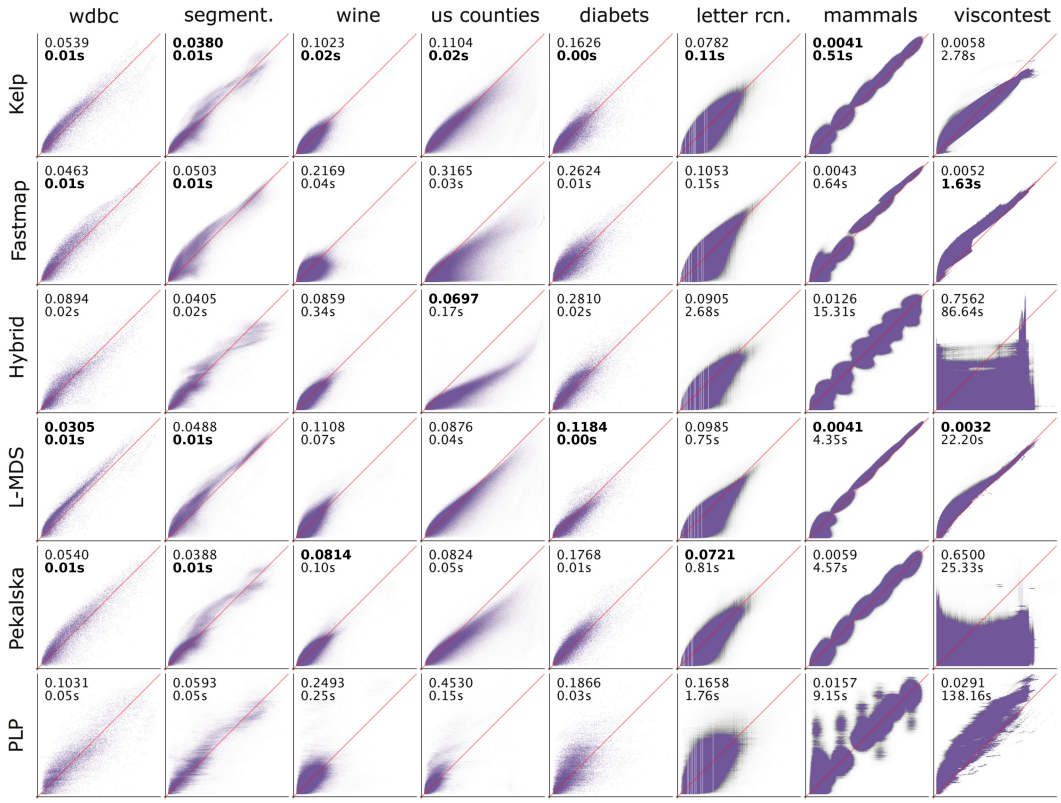
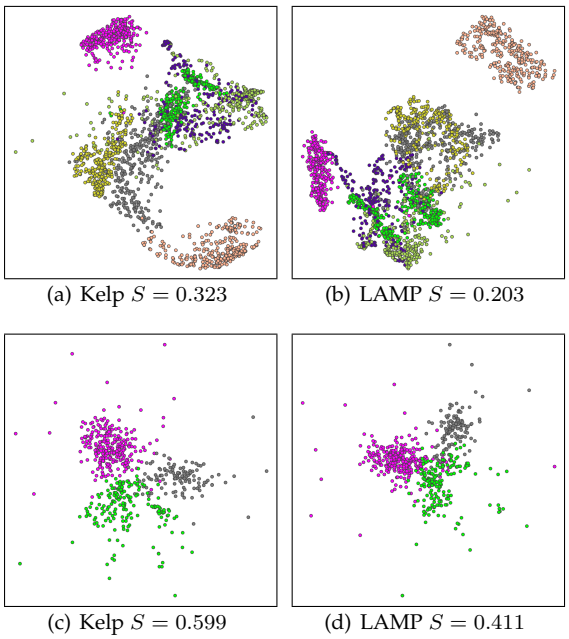


Fig. 2. Original-distance × projected-distance scatter plots: top-left numbers account for stress and computational time (in seconds). Bold values are the best results for each data set.

silhouette coefficient S, which assumes larger values in the layouts produced by Kelp. The *silhouette coefficient* accounts for both the cohesion and separation between grouped instances and it is computed as $S = \frac{1}{m}\sum_i \frac{(b_i - a_i)}{\max\{a_i, b_i\}}$, where a_i is the average distance between $\mathbf{y}_i$ (the image of x_i in the visual space) and all other instances in the same class as $\mathbf{y}_i$ and b_i is the minimum distance between $\mathbf{y}_i$ and all other instances in the other groups. The silhouette ranges in the interval $[-1, 1]$ and the larger the value of S the better is the cohesion and separation of the data.

Kelp's sensitivity with respect to user intervention is analyzed in Figure 4. Figure 4(a) shows the projection produced by Kelp when samples are arranged in the visual space by the Force Scheme. The top right inset depicts the position of the samples after applying the Force Scheme to a subset of randomly selected samples. Figures 4(b), 4(c), and 4(d) show the layouts produced by Kelp, PLMP, and LAMP respectively, after user intervention, that is, user has manually grouped samples accordingly to their classes so as to better define clusters in the visual space (see the top right insets). Notice that the layout resulting from Kelp has the highest silhouette value, even superior to LAMP, which is known to be quite sensitive to user intervention. Moreover, PLMP and LAMP require data embedded in a Cartesian space, thus they can not be directly employed in kernelized data.

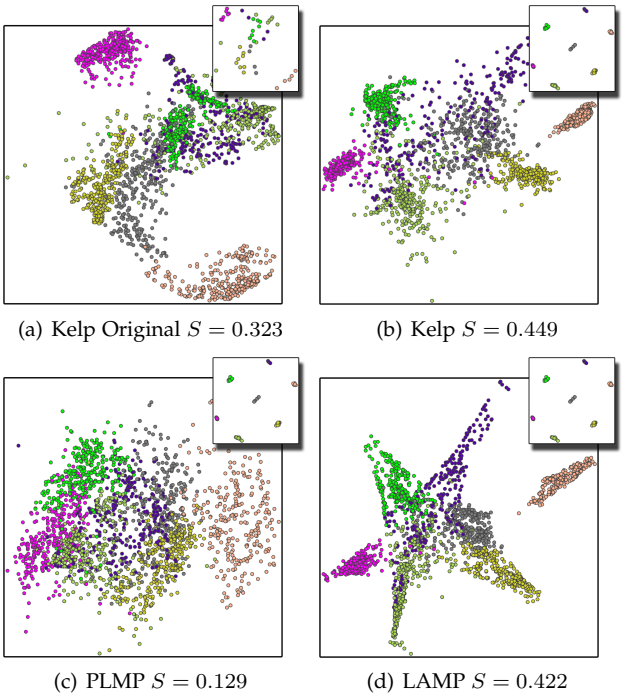
Figure 5 shows Kelp's behavior as to the number of samples used to steer the projection. It is easy to see that in most case stress stabilizes when nearly 5% of the data is used as samples. The jagged behavior of the curves is due to the random selection of sample instances, a phenomenon already reported in the literature [26]. Notice though that the amplitude of the oscillation is quite small, close to



(a) Kelp $S = 0.323$ (b) LAMP $S = 0.203$

(c) Kelp $S = 0.599$ (d) LAMP $S = 0.411$

Fig. 3. Improving the cohesion and separation of groups on the final projection using a kernel. The silhouette (S) is larger for projections created by kernel-based technique than using an Euclidean distance-based technique such as LAMP.



(a) Kelp Original $S = 0.323$ (b) Kelp $S = 0.449$

(c) PLMP $S = 0.129$ (d) LAMP $S = 0.422$

Fig. 4. Comparing Kelp's sensitivity as to user interaction. The upper right insets show the position of the samples.

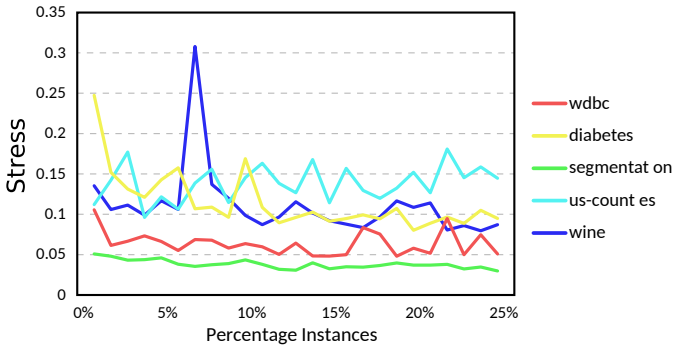
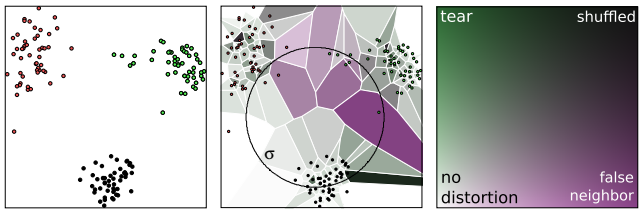


Fig. 5. Varying the sample size: stress stabilize nearly 5% of the instances used as samples.

0.05. In practice we notice that Kelp can provide good results even with a quite reduced number of samples, which encouraged us to use $\sqrt{n}$ as the number of samples in all our experiments. Users should start with such a reduced number of samples and if necessary (measuring the stress error) increase the number of samples.

We further analyze Kelp in terms of some artifacts that may appear in multidimensional projections, namely, tears and false neighbors. Tears happen when neighbor points in the feature space are mapped far apart from each other. False neighbors take place when distant point in the feature space a mapped close to each other in the visual space. As proposed in [35], we use a color code to represent distortions: purple indicating false neighbors, green indicating tears, black indicating that a point is a tear and a false neighbor simultaneously, and white corresponding to no distortion. The regions (Voronoi cells) in Figure 6 are colored by interpolating the color code of the corresponding site. Notice in Figure 6(b) that the green regions surrounding the clusters indicates that tears are happening, but false neighbors are only observed for points placed among the clusters. As pointed out by in [35], this is the best on can expected, since the projection is not creating misleading neighbors within clusters.



(a) Kelp projection (b) CheckViz Analysis (c) CheckViz color map

Fig. 6. Verifying the projection quality with CheckViz methodology: (a) projection of an artificial data set with 150 instances and 4 dimensions [36]; (b) purple regions indicate false neighbors and green regions indicate tears; (c) CheckViz color map. The circle σ indicates the size of the neighborhoods considered in each point.

5 Applications

In this section we illustrate the usefulness of Kelp in three distinct applications. The first application, motivated by the issue discussed in Section 1, employs Kelp as a tool to assist users in understanding the behavior of a kernel. More specifically, we rely on Kelp to build a tool that allows for visually analyzing how a kernel affects neighborhood relations. The second application exploits the interactive mechanism enabled by Kelp to assist Support Vector Machine data classification tasks. The third application shows that a kernel based visualization process can greatly benefit from a visualization assisted mechanism such as Kelp.

5.1 Kernel Induced Neighborhood Changes

As motivated in Section 1, figuring out how a kernel affects neighborhood structures is of paramount importance for the proper choice, design, and tuning of kernels in specific applications. The visualization tool described in this section is a first attempt in visualizing the behavior of kernels and assisting users in kernel-based applications.

Our approach relies on a metric to compare neighborhood structures defined in the original Cartesian space against their counterpart in the feature space induced by a kernel. The metric is defined as follows: let $X = \{x_1, x_2, \ldots, x_m\}$ be a set of instances in a Cartesian space (we will use the same symbol x_i to represent the data and its vector representation) and $\delta_i = x_i - \frac{1}{\#N_i}\sum_{j \in N_i} x_j$ be the differential coordinate of x_i, where N_i accounts for the indexes of the k-nearest neighbors of x_i and $\#N_i$ is the cardinality of N_i. The norm $\|\delta_i\|$ is a measure of how far x_i is from the centroid of its neighbors. Let now $\phi(x_i)$ be the image of x_i in a feature space induced by a kernel k.

The norm of the differential coordinate δ_{ϕ_i} of $\phi_i = \phi(x_i)$ in the feature space is given by $\|\delta_{\phi_i}\| = \sqrt{\delta_{\phi_i}^\top \delta_{\phi_i}}$ with

$$\begin{aligned} \|\delta_{\phi_i}\|^2 &= \left(\phi_i - \frac{1}{\#N_i}\sum_{j\in N_i}\phi_j\right)^\top \left(\phi_i - \frac{1}{\#N_i}\sum_{j\in N_i}\phi_j\right) \\ &= k(x_i,x_i) - \frac{2}{\#N_i}\sum_{j\in N_i} k(x_i,x_j) \\ &+ \frac{1}{(\#N_i)^2}\sum_{j,s\in N_i} k(x_j,x_s) \end{aligned} \tag{14}$$

Equation (14) shows that the norm of differential coordinates in feature space can be obtained from kernel values, making it possible to measure how far each instance $\phi(x_i)$ is from the centroid of its neighbors in the feature space. Notice that we are always defining neighborhoods in the Cartesian space, because our goal is to measure how those neighborhoods are affected by the kernel.

Figures 7(a) and 7(b) show color maps corresponding to values of $\|\delta_i\|$ and $\|\delta_{\phi_i}\|$ computed in each data instance (artificial data set generated from [36]) in layouts generated by PLMP and Kelp, respectively. Red regions correspond to large values of $\|\delta_i\|$ and $\|\delta_{\phi_i}\|$ while blue colors represent low values, (green color accounts for intermediate values). We choose PLMP to project the original data to the visual space because, as Kelp, PLMP makes use of linear transformation to map the data, thus enabling a fair visual comparison of the resulting layouts. Notice that that after applying the kernel groups of instances becomes even better defined.

The ratio $\|\delta_i\|/\|\delta_{\phi_i}\|$ measures changes in neighborhood structures when data is embedded in a feature space by the kernel k. Values close to 1 indicate no changes, values close to 0 indicate that instances get farther from their neighbors in a non-symmetric way, and values greater than 1 means that, after applying the kernel, instances become more centralized with respect to their neighbors. Using a transfer function as illustrated in Figure 8, we can visualize the regions where neighborhoods are more affected by the kernel. The background in Figure 7 is colored by interpolating the differential coordinate ratio values from each instance to a background regular grid. Figure 7(b) tells us that the Gaussian kernel better positions instances in terms of their neighbors within the data groups, that is, within the well defined groups, the Gaussian kernel tend to place instances closer to the centroid of their neighbors. However, Figure 7(c) clearly shows that, when analyzing the ratio between the norm of differential coordinates, red regions (corresponding to values close to zero or greater than one) show up within well defined groups. Since $\|\delta_{\phi_i}\|$ is small within well defined groups (Figure 7(b)) and the groups have not spread out due to the kernel action, we conclude that the large values of $\|\delta_i\|/\|\delta_{\phi_i}\|$ are due to a tighter grouping produced by the Gaussian kernel. Therefore, as expected, a Gaussian kernel tends to better define the groups.

The experiment above involving a Gaussian kernel validates and supports the correctness of our methodology. The same analysis can be performed with kernels other than Gaussian, as illustrated in Figure 7(d) and 7(e). Figure 7(d) depicts $\|\delta_{\phi_i}\|$ when a polynomial kernel given by

$$k(x_i,x_j) = \left(x_i^\top x_j\right)^2 \tag{15}$$

is used to map data to a feature space. Polynomial kernels are less intuitive than Gaussian kernels, thus hampering their use in practical applications. Using visualization tool, though, one can see that the polynomial kernel Equation (15) behaves quite similarly to the Gaussian kernel, avoiding to push "outliers" closer to clusters while tightening instances that lie within clusters.

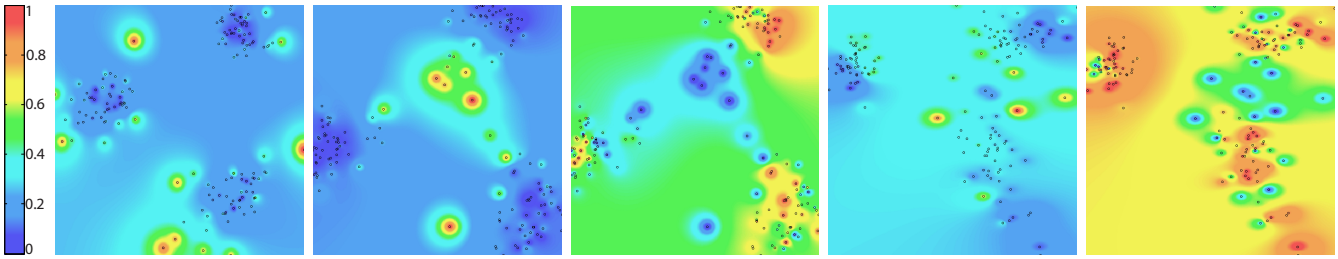
As one can clearly see, differential coordinates turn out to be quite effective to visualize neighborhood changes induced by kernels. It is worth mentioning that, as far as we know, this is the first time that differential coordinates is used to measure neighborhood structures in the context of kernelized data, thus being another contribution of this work.

5.2 SVM Visualization

We take advantage of the flexibility provided by Kelp in terms of interactive resources to support data classification tasks, more precisely, Support Vector Machine classification. SVM is a linear classifier that operates in feature space (nonlinear on input space) where the separating hyperplane maximizes the training margin. Intuitively, instances away from the margins in feature space are classified with good degree of confidence while instances laying inside the margins are more likely to be wrongly classified. Therefore, the typical mental model of a SVM classifier (assuming two classes) comprises two planar regions where data can be classified with certain confidence and a strip bounded by two straight lines (the margins) defining the region of uncertainty.

We can exploit the flexibility enabled by Kelp to interactively change the position of sample points in the visual space to realize the SVM mental model. Figure 9 presents the projection of the wdbc data set (see Table 1) using a Gaussian kernel. The same Gaussian was used as kernel for the SVM. Using the LIBSVM [37] to perform the SVM classification, we get, for each instance, the probability to belong to a class. Darker colors in Figure 9 correspond to instances where SVM has high confidence in terms of classification while lighter instances correspond to the ones with low confidence and the separation line (black) corresponds the region where the interpolated probability (given by the SVM classifier) is 50%. Notice in Figure 9(a) that confidence regions can not be clearly defined when the Force Scheme is applied to position sample points in the visual space. Since the data set used in this experiment has a reduced number of instances, we use 10% of the instances as sample points in order to enable a better interaction.

However, when the sample points are interactively arranged so as to distinguish the classes, as depicted in Figure 9(b), the resulting layout clearly uncoveres the usual SVM mental model, making it easier to interpret the behavior of the classifier. Notice that even the margin where the classification is dubious clearly shows up when sample points are properly arranged. The background in Figure 9 is colored by interpolating in a grid the probability (given by



(a) $\|\delta_i\|$ (PLMP) (b) $\|\delta_{\phi_i}\|$ (Kelp-Gaussian) (c) $\frac{\|\delta_i\|}{\|\delta_{\phi_i}\|}$ (Kelp-Gaussian) (d) $\|\delta_{\phi_i}\|$ (Kelp-$\langle\cdot,\cdot\rangle^2$) (e) $\frac{\|\delta_i\|}{\|\delta_{\phi_i}\|}$ (Kelp-$\langle\cdot,\cdot\rangle^2$)

Fig. 7. Visualizing how a kernel affects neighborhood structures (artificial data set with 150 instances and 4 dimensions): differential coordinate magnitudes in a layout generated by PLMP with Euclidean distance (a) and Kelp using Gaussian (b) and polynomial kernels (d); Magnitude ratio in the Kelp-Gaussian (c) and Kelp-polynomial layouts (e).

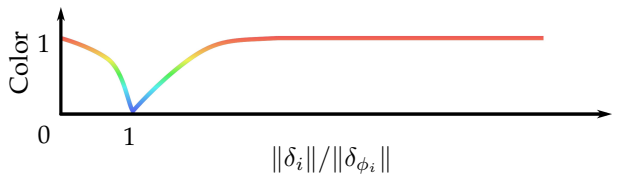
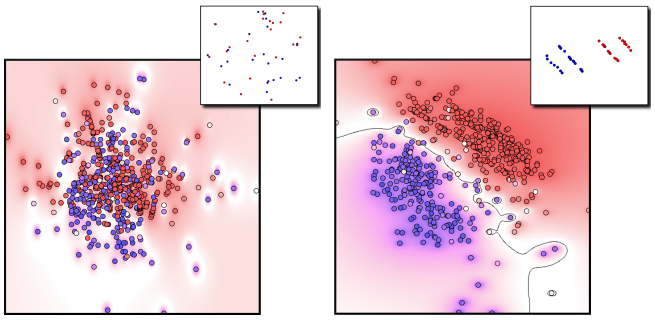


Fig. 8. Color transfer function.



(a) (b)

Fig. 9. Realizing the mental model of a SVM classifier by interactively arranging the sampling points in the visual space with the separation line (black): (a) initial projection of classified data where the separation line is unclear and (b) projection after layout manipulation. The upper right corner image shows the sampling points position.

the SVM classifier) of each instance to belong to one of the classes.

It is important to say that other method devoted to visualize the output of SVM classifiers [38], [39] assume as input Cartesian data (kernel is only used in the classifier), what is not the case with Kelp.

5.3 Kernel-based Image segmentation

User assisted techniques comprise an important class of image segmentation methods. In this context, most methods enable interactive resources for users to brush seeds or regions on the image space, driving the segmentation process from the brushes [40]. However, interactive image segmentation methods that operate in the feature space are not so common, although recent works have shown the advantage of interacting directly on feature to improve the performance of tasks such as image coloring and retrieval [41], [42].

In the following we show how Kelp can be used to support an interactive image segmentation application that operates directly in feature spaces. More specifically, we build upon bilateral filtering [43] to define a kernel and use Kelp to enable interactive resources for users to brush regions in the feature space induced by the kernel.

Let I be an input image and $\mathcal{I}$ its filtered counterpart generated by the bilateral filter:

$$\mathcal{I}_p = \frac{1}{W} \sum_{q \in N_p} I_p \, \mathbf{G}_{\sigma_1} \left(\|I_p - I_q\| \right) \mathbf{G}_{\sigma_2} \left(\|p - q\| \right)$$

$$\text{with} \quad W = \sum_{q \in N_p} \mathbf{G}_{\sigma_1} \left(\|I_p - I_q\| \right) \mathbf{G}_{\sigma_2} \left(\|p - q\| \right) ,$$

where $\mathbf{G}_\sigma(x) = \exp(-x^2/2\sigma^2)$. The values I_p and $\mathcal{I}_p$ are the color intensities in CIE-Lab color space for a pixel p in I and $\mathcal{I}$, respectively, σ^2 is the variance typically used in Gaussian filters and N_p is a square pixel neighborhood centered in p.

Let k be a kernel defined as follows:

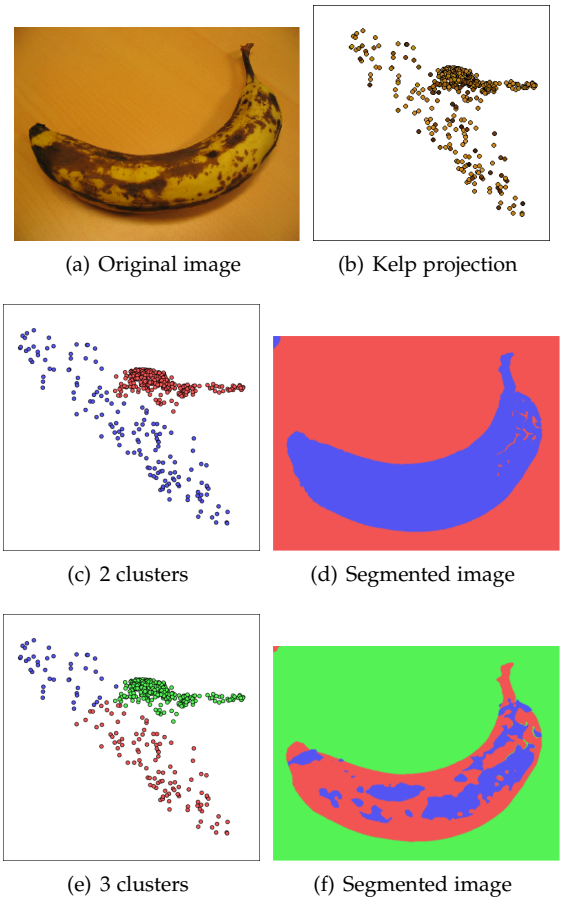
$$k\left(p, q\right) = \mathbf{G}_\sigma \left(\|\mathcal{I}_p - \mathcal{I}_q\| \right) . \tag{16}$$

Figure 10(b) shows the mapping of each pixel in Figure 10(a) using the kernel defined in Equation (16) with the Kelp technique. The color assigned to each mapped point is the color of the corresponding pixel in the original image. Notice that (Figures 10(c) and 10(d)) in the mapping produced by Kelp the yellow background is clearly separated from the yellow part of the banana. Moreover, the user can interactively define clusters in the projection layout, which is equivalent to picking out regions in the feature space and back in the original image, as illustrated in Figures 10(e) and 10(f).

The image segmentation application described above shows that Kelp can, in fact, assist in the construction of kernels for specific purposes. The kernel defined in Equation (16) is only one example.

6 Discussion and Limitations

Comparisons and results presented in Section 4 clearly show the effectiveness of Kelp, which presented a good trade-off between accuracy and computational time. The solid mathematical foundation supporting Kelp's formulation ensures distance preserving and versatility towards incorporating user knowledge into the projection process. In fact, among techniques able to dealing with kernels and that enable user intervention, Kelp turned out to be one of the best alternatives. Simplicity as to computational implementation is another strength of Kelp, which essentially requires a numerical eigendecomposition library. The need for kernel



(a) Original image (b) Kelp projection

(c) 2 clusters (d) Segmented image

(e) 3 clusters (f) Segmented image

Fig. 10. Image segmentation pipeline.

values only between instances and samples is another positive aspect of Kelp, as storage is pushed down from $O(m^2)$ to $O(mn)$, where n, the number of samples, is much smaller than m, the number of instances in the whole data set.

Another interesting aspect of our technique is that it is inherently incremental and parallelizable. Only the left most term $\mathbf{k}_x$ in Equation (10) changes when projecting distinct data instances. Therefore, once the matrix K_s is built and its eigendecomposition performed, one can project instances x independently from each other, that is, the product of the matrix $Y K_s A \Gamma^{-1} A^\top$ by the vector $\mathbf{k}_x$ can be done in parallel for each instance.

Although simple, the application devoted to visualize neighborhood changes induced by kernels has tuned out to be quite interesting and it opens a multitude of possible visualization alternatives. In the provided application we have just exploited the norm of differential coordinates as a measure of neighborhood changes, but many other metrics could also be employed. We are currently investigating more sophisticated visualization mechanisms towards further understanding how kernels act on data sets. Moreover, the other two applications, namely, interactive image segmentation and SVM space visualization, prove that Kelp can support a variety of applications.

An issue that is not properly related to our formulation but impacts directly on its results is how to set the parameters that control a kernel. For instance, it is well known that the parameter σ used in Gaussian kernels affects the result and effectiveness of SVM classifiers and Kernel PCA dimensionality reduction techniques [1]. Finding the appropriate value of σ to reach the best result is difficult. In our tests we used the average variance of the data to set σ, but such an automatic mechanism did not work properly for certain data sets, which demanded a manual fine tune of σ. We believe that Kelp can also be a very useful tool to assist the task of setting the parameters controlling the behavior of kernels, being this one interesting aspect to be explored in a future work.

7 Conclusion

In this work we proposed a novel projection technique designed specifically to map kernelized data to a visual space. Called Kelp, the proposed method has a solid mathematical foundation and it outperforms state-of-art techniques with regard to accuracy and computational times. The potential use of Kelp to support kernel-based applications with visualization resources opens new possibilities which could not be efficiently addressed until now. Therefore, flexibility, effectiveness, and ease of implementation render Kelp one of the most attractive multidimensional projection methods for handling kernelized data.

Acknowledgments

The authors acknowledge the financial support from FAPESP (#2011/22749–8), (#2013/19760–5), (#2014/09546–9), CNPq (#302643/2013–3), CAPES and Samsung.

References

[1] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

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

[3] P. Joia, D. Coimbra, J. A. Cuminato, F. V. Paulovich, and L. G. Nonato, "Local affine multidimensional projection," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 2563–2571, 2011.

[4] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[5] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, *A kernel view of the dimensionality reduction of manifolds*, ser. ICML '04. New York, NY, USA: ACM, 2004.

[6] F. Inaba, E. O. T. Salles, and T. Rauber, "Kernel sammon map," in *Sibgrapi 2011 (24th Conference on Graphics, Patterns and Images)*, 2011, pp. 329–336.

[7] H. Choi and S. Choi, "Kernel isomap," *Electronics Letters*, vol. 40, no. 25, pp. 1612–1613, 2004.

[8] W. S. Torgeson, "Multidimensional scaling of similarity," *Psychometrika*, vol. 30, pp. 379–393, 1965.

[9] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[10] H. Hosobe, "A high-dimensional approach to interactive graph visualization," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 1253–1257.

[11] V. de Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," Stanford, Tech. Rep., 2004.

[12] U. Brandes and C. Pich, "Eigensolver methods for progressive multidimensional scaling of large data," in *Graph Drawing*, ser. Lecture Notes in Computer Science. Springer, 2007, vol. 4372, pp. 42–53.

[13] C. Faloutsos and K. Lin, "FastMap: A fast algorithm for indexing, datamining and visualization of traditional and multimedia databases," in *ACM SIGMOD*, 1995, pp. 163–174.
[14] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Computation*, vol. 15, no. 6, pp. 1373–1396, 2003.
[15] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[16] Y. Koren, L. Carmel, and D. Harel, "ACE: A fast multiscale eigenvectors computation for drawing huge graphs," in *IEEE Information Visualization*, 2002, pp. 137–144.
[17] M. M. Bronstein, A. M. Bronstein, R. Kimmel, and I. Yavneh, "Multigrid multidimensional scaling," *Numerical Linear Algebra with Applications*, vol. 13, pp. 149–171, 2006.
[18] M. Chalmers, "A linear iteration time layout algorithm for visualising high-dimensional data," in *IEEE Visualization*, 1996, pp. 127–ff.
[19] F. Jourdan and G. Melançon, "Multiscale hybrid MDS," in *Information Visualisation*, 2004, pp. 388–393.
[20] A. Morrison, G. Ross, and M. Chalmers, "A hybrid layout algorithm for sub-quadratic multidimensional scaling," in *IEEE Information Visualization*, 2002, pp. 152–158.
[21] Y. Frishman and A. Tal, "Multi-level graph layout on the GPU," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 1310–1319, 2007.
[22] S. Ingram, T. Munzner, and M. Olano, "Glimmer: Multilevel MDS on the GPU," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 2, pp. 249–261, 2009.
[23] 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 *Annual Conf. Advanced School for Comput. Imag.*, 1999, pp. 221–228.
[24] 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*, vol. 14, no. 3, pp. 564–575, 2008.
[25] F. V. Paulovich, D. M. Eler, J. Poco, C. P. Botha, R. Minghim, and L. G. Nonato, "Piecewise Laplacian-based projection for interactive data exploration and organization," *Computer Graphics Forum*, vol. 30, no. 3, pp. 1091–1100, 2011.
[26] F. V. Paulovich, C. T. Silva, and L. G. Nonato, "Two-phase mapping for projecting massive data sets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1281–1290, 2010.
[27] J. Alsakran, Y. Chen, D. Luo, Y. Zhao, J. Yang, W. Dou, and S. Liu, "Real-time visualization of streaming text with a force-based dynamic system," *IEEE Computer Graphics and Applications*, no. 1, pp. 34–45, 2011.
[28] X. Hu, L. Bradel, D. Maiti, L. House, C. North, and S. Leman, "Semantics of directly manipulating spatializations," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 19, no. 12, pp. 2052–2059, 2013.
[29] E. Tejada, R. Minghim, and L. G. Nonato, "On improved projection techniques to support visual exploration of multidimensional data sets," *Information Visualization*, vol. 2, no. 4, pp. 218–231, 2003.
[30] M. L. Braun, J. Schaback, M. L. Jugel, and N. Oury, "jBlas: Linear algebra for java," 2011. [Online]. Available: http://www.jblas.org/
[31] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml
[32] B. Shneiderman and J. Seo, "Hierarchical clustering explorer for interactive exploration of multidimensional data," 2008. [Online]. Available: http://www.cs.umd.edu/hcil/hce/examples/application_examples.html
[33] D. Whalen and M. L. Norman, "Competition data set and description," in *2008 IEEE Visualization Design Contest*, 2008. [Online]. Available: http://vis.computer.org/VisWeek2008/vis/contests.html
[34] M. W. Berry, "Large-scale sparse singular value computations," *International Journal of Supercomputer Applications*, vol. 6, no. 1, pp. 13–49, 1992.
[35] S. Lespinats and M. Aupetit, "Checkviz: Sanity check and topological clues for linear and non-linear mappings," *Computer Graphics Forum*, vol. 30, no. 1, pp. 113–125, 2011.
[36] I. Guyon, "Design of experiments of the NIPS 2003 variable selection benchmark," in *NIPS 2003 workshop on feature extraction and feature selection*, 2003. [Online]. Available: http://www.nipsfsc.ecs.soton.ac.uk/datasets/
[37] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[38] A. Jakulin, M. Možina, J. Demšar, I. Bratko, and B. Zupan, "Nomograms for visualizing support vector machines," in *ACM SIGKDD*, 2005, pp. 108–117.
[39] L. Hamel, "Visualization of support vector machines with unsupervised learning," in *IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, 2006, pp. 1–8.
[40] W. Casaca, A. Paiva, E. Gomez-Nieto, P. Joia, and L. G. Nonato, "Spectral image segmentation using image decomposition and inner product-based metric," *Journal of mathematical imaging and vision*, vol. 45, no. 3, pp. 227–238, 2013.
[41] W. Casaca, E. Gomez-Nieto, C. O. Ferreira, G. Tavares, P. Pagliosa, F. Paulovich, L. G. Nonato, and A. Paiva, "Colorization by multidimensional projection," in *Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2012, pp. 32–38.
[42] G. M. Mamani, F. M. Fatore, L. G. Nonato, and F. V. Paulovich, "User-driven feature space transformation," in *Computer Graphics Forum*, vol. 32, no. 3pt3. Wiley Online Library, 2013, pp. 291–299.
[43] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in *Proceedings of the Sixth International Conference on Computer Vision*, ser. ICCV '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 839+.

Adriano Barbosa received the BSc in Mathematics in 2008 and MSc degree in Applied Mathematics in 2011, both from the Universidade Federal de Alagoas, Brazil. He is currently a PhD candidate at the Instituto de Ciências Matemáticas e de Computação (ICMC) — Universidade de São Paulo (USP) — Brazil and member of Visual and Geometry Processing Group at ICMC-USP. His research interests comprise interactive visualization for kernelized data.

Fernando V. Paulovich received the BSc in computer science from the Universidade Federal de São Carlos, Brazil, in 2000, and the PhD degree in computer sciences from the Universidade São Paulo, Brazil, in 2008, with a period as invited researcher at the Delft University of Technology, the Netherlands. He is currently an associate professor at the Computer Science Department of the Instituto de Ciências Matemáticas e de Computação, at the Universidade de São Paulo, Brazil. His research interests involve computational visualization, more specifically information visualization, visual analytics and visual data mining. He is a member of the Brazilian Computer Society.

Afonso Paiva is an assistant professor at the Instituto de Ciências Matemáticas e de Computação — Universidade de São Paulo (ICMC-USP), Brazil. He holds a PhD degree (2007) in Applied Mathematics from Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) and a MSc degree (2003) in Mathematics from the Instituto Nacional de Matemática Pura e Aplicada (IMPA), Rio de Janeiro, Brazil. From 2003 to 2007, he worked in research projects supported by Petrobras (Brazilian Oil Company) at PUC-Rio. He is currently member of Visual and Geometry Processing Group at ICMC-USP and his research interests comprise computer animation, geometry processing, geometric modeling and numerical methods in computer graphics.

Siome Goldenstein is an Associate Professor at the Institute of Computing, Universidade Estadual de Campinas, Unicamp, Brazil, and a senior IEEE member. He received an Electronic Engineering degree from the Universidade Federal do Rio de Janeiro in 1995, an MSc in Computer Science from the Pontifícia Universidade Católica do Rio de Janeiro in 1997, and a PhD in Computer and Information Science from University of Pennsylvania in 2002. In 2003, he was a postdoctoral fellow at the CBIM Center, at Rutgers University, and during 2010–2012 he was a Visiting Associate professor at the Division of Engineering, Brown University. He is an Area Editor of IEEE Transactions on Information Forensics and Security (T.IFS), Elsevier's Computer Vision and Image Understanding (CVIU), and Elsevier's Graphical Models (GMOD). His interests lie in computational forensics, computer vision, computer graphics, and machine learning.

Fabiano Petronetto received his PhD degree in Applied Mathematics from Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio) in 2008. After a short time working in research projects supported by Petrobras (Brazilian Oil Company), he started as professor of the Mathematics Department at Universidade Federal do Espírito Santo in 2009. His research interests are in development of meshless methods using the particle method Smoothed Particle Hydrodynamics (SPH). He has collaborated with the Visual and Geometry Processing Group at Instituto de Ciências Matemáticas e de Computação (ICMC-USP), where he currently holds a post-doc (2013–2015).

Luis Gustavo Nonato received the PhD degree in applied mathematics from the Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro — Brazil, in 1998. He is a full professor at the Instituto de Ciências Matemáticas e de Computação (ICMC) — Universidade de São Paulo (USP) — Brazil. He spent a sabbatical leave in the Scientific Computing and Imaging Institute at the University of Utah from 2008 to 2010. Besides having served in several program committees, including IEEE Visualization and Eurovis, he was a member of the editorial board of Computer Graphics Forum and the president of the Special Committee on Computer Graphics and Image Processing of Brazilian Computer Society. Currently Dr. Nonato leads the Visual and Geometry Processing Group at ICMC-USP.