SIAM J. MATH. DATA SCI.
Vol. 4, No. 1, pp. 410–430

Spectral Discovery of Jointly Smooth Features for Multimodal Data[*]

Felix Dietrich[†], Or Yair[‡], Rotem Mulayoff[‡], Ronen Talmon[‡], and Ioannis G. Kevrekidis[§]

Abstract. In this paper, we propose a spectral method for deriving functions that are jointly smooth on multiple observed manifolds. This allows us to register measurements of the same phenomenon by heterogeneous sensors and to reject sensor-specific noise. Our method is unsupervised and primarily consists of two steps. First, using kernels, we obtain a subspace spanning smooth functions on each separate manifold. Then, we apply a spectral method to the obtained subspaces and discover functions that are jointly smooth on all manifolds. We show analytically that our method is guaranteed to provide a set of orthogonal functions that are as jointly smooth as possible, ordered by increasing Dirichlet energy from the smoothest to the least smooth. In addition, we show that the extracted functions can be efficiently extended to unseen data using the Nyström method. We demonstrate the proposed method on both simulated and real measured data and compare the results to nonlinear, kernel-based variants of the seminal canonical correlation analysis. Particularly, we show superior results for sleep stage identification. In addition, we show how the proposed method can be leveraged for finding minimal realizations of parameter spaces of nonlinear dynamical systems.

Key words. manifold learning, SVD, CCA, kernel methods, data fusion, multimodal data analysis

AMS subject classifications. 46Nxx, 47Nxx, 35K08, 58C05

DOI. 10.1137/21M141590X

1. Introduction.

Modern data acquisition typically involves the use of multiple modalities. Indeed, nowadays, many high- and low-end devices are equipped with multiple sensors, often of different types, giving rise to multimodal data collections. While such acquisitions have become popular only in the last two decades, the discovery of informative features from multimodal observations has been a central problem in data analysis for many years.

Perhaps the first attempt and the most well-known and widely used algorithm for building efficient representations for multimodal data sets is the celebrated canonical correlation analysis (CCA) [24]. Broadly, CCA builds linear projections of the data sets such that the correlation between them is maximized. Despite the extensive body of evidence proving CCA as extremely useful, it suffers from a few prominent shortcomings, facilitating a large body of work improving CCA. Kernel CCA (KCCA) [27, 3] extends the linear setting and considers instead nonlinear projections by using the kernel trick [31]. Multiview and weighted versions

[*]Received by the editors April 28, 2021; accepted for publication (in revised form) October 27, 2021; published electronically March 24, 2022.
https://doi.org/10.1137/21M141590X

Funding: The second and fourth authors were funded by the European Union's Horizon 2020 research and innovation program under grant agreement 802735-ERC-DIFFOP. The first and fifth authors were funded by the U.S. Army Research Office under contract/grant W911NF1710306 (an ARO MURI) and the DARPA ATLAS program.

[†]Department of Informatics, Technical University of Munich, Munich, 80333 Germany (felix.dietrich@tum.de).

[‡]Viterbi Faculty of Electrical Engineering, Technion, Israel Institute of Technology, Haifa, 3200003 Israel (oryair@campus.technion.ac.il, rotem.mulayof@gmail.com, ronen@ee.technion.ac.il).

[§]Department of Chemical and Biomolecular Engineering and Department of Applied Mathematics and Statistics, Johns Hopkins University and JHMI, Baltimore, MD 21218 USA (yannis@princeton.edu).

of KCCA were proposed in [48, 29]. Deep learning tools allowed the significant extension of the space of possible projections, giving rise to deep CCA [4]. The nonparametric CCA (NCCA) [32] provides a closed-form solution to a nonlinear variant of the CCA optimization problem, assuming that the projections belong to some reproducing kernel Hilbert space (RKHS).

In addition to the statistical viewpoint adopted by CCA and its variants, recently, geometric and manifold learning methods were proposed as well [18, 28, 29] for the purpose of finding common representations to multimodal data. Particularly, in [18], the data are viewed as high-dimensional points residing on low-dimensional manifolds, and the objective is to obtain a basis that simultaneously diagonalizes the Laplacians of all the manifolds. Since this objective cannot be fully achieved, the authors introduce certain off-diagonal penalties to their objective function.

Relying on these recent developments, in this paper, we propose a different approach based on manifold learning. We retain the setting as in [18, 46, 28, 29], assuming that the multimodal data lie on multiple manifolds, building the Laplacian of each manifold and then applying spectral analysis. Instead of products of kernel matrices, however, our approach focuses on the space of real functions defined on these manifolds. We construct a joint function space for all observations, sorted by decreasing smoothness. Such an approach has roots in operator theoretic dynamical system analysis, dating back to Koopman [26], and has recently regained a lot of attention in learning dynamical systems from observations [10, 45, 16] as well as in computer graphics [33]. Concretely, we define smooth functions on a manifold as functions that can be spanned by the top eigenfunctions of the Laplace–Beltrami operator of the manifold. Then, our main goal is to discover functions that are jointly smooth on all observed manifolds. We show that such jointly smooth functions are of great interest since they represent and parametrize the commonality between the observed manifolds, which provides useful information on the co-relationship between the respective data sets obtained through possibly different modalities. To accomplish our goal, we propose a two-step algorithm. First, for each manifold, we extract a subspace spanning the family of smooth functions on that manifold. This is achieved by constructing a kernel on the data approximating the Laplacian. Second, we apply singular value decomposition (SVD) to the union of all subspaces and show analytically that the obtained singular vectors indeed represent jointly smooth functions on the observed manifolds. We test the proposed algorithm on real measured multimodal data. Specifically, we demonstrate the ability to accurately identify the sleep stage from various simultaneous physiological recordings. In addition, we show how joint latent spaces can help in positional alignment from multiple video feeds of a race track and how this method can be exploited for finding effective parameters of nonlinear dynamical systems from observations in a model-free manner.

Our main contributions are as follows. First, we introduce an approach for multimodal data analysis based on the notion of jointly smooth functions on manifolds. Second, we propose an algorithm for finding such jointly smooth functions with theoretical guarantees. We show that the proposed algorithm supports multiple (more than two) manifolds and that it can be efficiently computed and extended to unseen data. We also discuss how the jointly smooth functions can be used to embed the data common to the different sensors in a shared latent space. Third, we show how jointly smooth functions can be constructed efficiently, with an almost linear time and space complexity. Fourth, we showcase the proposed algorithm on simulated and real measured data sets. Specifically, we consider a sleep stage identification task and observe superior results compared to competing methods. We also demonstrate how to obtain a common latent space for two video streams and identify effective parameters in a dynamical systems context.

2. Smooth functions.

2.1. Smooth functions on manifolds.

Let $\mathcal{M}_x \subseteq \mathbb{R}^d$ be a smooth, compact manifold embedded in a d-dimensional Euclidean space. A function $f : \mathcal{M}_x \to \mathbb{R}$ is said to be smooth if $f \in C^\infty(\mathcal{M}_x, \mathbb{R})$; that is, the function f and all of its derivatives are continuous. In this section, we define a different notion of smoothness related to the Laplace–Beltrami operator. The Laplace–Beltrami operator L_x is a linear operator generalizing the Laplacian on Euclidean spaces to Riemannian manifolds [36]. The eigenfunctions ψ_i of the Laplace–Beltrami operator span a dense subset of the function space $H^0 = L^2(\mathcal{M}_x, \mathbb{R})$. The eigenvalues ν of the operator are real (and nonnegative), so we can sort the associated eigenfunctions ψ_i and ψ_j such that $\nu_i \le \nu_j$ for $i < j$. We say that ψ_i is smoother than ψ_j if $\nu_i < \nu_j$. For example, the constant function $\psi \equiv 1$ is an eigenfunction with eigenvalue $\nu = 0$. The higher the value of ν_i, the more oscillatory the corresponding eigenfunction is (that is, less smooth). For more details, we refer to [20]. It was shown that the best representation bases, in terms of truncated representation of functions $f : \mathcal{M}_x \to \mathbb{R}$ such that $\|\nabla f\|_2 \le 1$, are in fact the eigenfunctions of the Laplace–Beltrami operator L_x [2, 1, 9]. Thus, in that sense, we say that $f_i : \mathcal{M}_x \to \mathbb{R}$ is smoother than $f_j : \mathcal{M}_x \to \mathbb{R}$ if $\|L_x f_i\|_2/\|f_i\|_2 < \|L_x f_j\|_2/\|f_j\|_2$. Note that we write $\|\cdot\|_2$ for the L^2 norm on H^0, but we use the same notation if we consider finite-dimensional approximations and the corresponding 2-norm in N dimensions later.

2.2. Smooth functions on data.

Consider a set of points $\{\boldsymbol{x}_i \in \mathcal{M}_x\}_{i=1}^N$ residing on a low-dimensional manifold $\mathcal{M}_x \subseteq \mathbb{R}^d$ embedded in a d-dimensional Euclidean space. Many unsupervised data analysis methods use kernels to effectively represent data [37, 5, 38, 13]. Kernels are typically symmetric and positive functions [31], represented as matrices by evaluating them on pairs of data points. Perhaps the most widely used kernel is the Gaussian kernel, given by

$$\boldsymbol{K}_x[i,j] = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma_x^2}\right), \qquad \boldsymbol{K}_x \in \mathbb{R}^{N\times N}.$$

This Gaussian kernel $\boldsymbol{K}_x$ approximates the operator $\exp(-\sigma_x L_x)$, whose eigenfunctions are the same as the eigenfunctions of L_x [47]. Eigenvalues λ_i of $\boldsymbol{K}$ and eigenvalues ν_i of L_x are thus sorted in opposing order and can be converted through the relation $\lambda_i \approx \exp(-\sigma_x \nu_i)$. The parameter $\sigma_x > 0$ has several interpretations: it defines the half-width of the squared exponential function, it serves as the physical time in the diffusion equation when solved with a Dirac delta as initial condition, and it defines the smoothness of functions generated through a Gaussian process with the given kernel. Hyperparameter tuning of this parameter is often done through log-marginal likelihood optimization in Bayesian statistics [35] or cutoff arguments and metric estimation in manifold learning [40, 34, 7, 8]. If not otherwise stated, we employ a simple heuristic approach and set $\sigma = \frac{1}{2}\text{median}\left\{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|\right\}_{i,j}$, as proposed in [40]. See [34] for an in-depth discussion of the kernel scale estimation problem in manifold learning. Similar to the matrix representation of the kernel, we use a vector representation of functions through their evaluation on the data. Let $\boldsymbol{w}_i$ be a unit norm eigenvector of $\boldsymbol{K}_x$ such that $\boldsymbol{K}_x\boldsymbol{w}_i = \lambda_i\boldsymbol{w}_i$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. These eigenvectors form a basis ordered from the smoothest vector $\boldsymbol{w}_1$ to the most oscillatory vector $\boldsymbol{w}_N$, where smoothness is defined in a manner similar to the smoothness with respect to the Laplace–Beltrami operator L_x. The following definition is inspired by the Dirichlet energy $\|\nabla f\|_2^2$ of a function [20] and adapted to kernel matrices. Note that Dirichlet energy decreases for smoother functions, while our smoothness score increases.

Definition 2.1 (truncated smoothness score). *For a number $d < N$ and a given function $\boldsymbol{f} \in \mathbb{R}^N$, define its d-truncated smoothness score $E_x^d(\boldsymbol{f})$ with respect to a kernel $\boldsymbol{K}_x$ by*

$$E_x^d(\boldsymbol{f}) \coloneqq \sum_{i=1}^{d} \langle \boldsymbol{w}_i, \boldsymbol{f}\rangle^2 = \left\|\boldsymbol{W}_x^T\boldsymbol{f}\right\|_2^2, \tag{2.1}$$

where $\boldsymbol{W}_x \coloneqq [\boldsymbol{w}_1 \quad \boldsymbol{w}_2 \quad \cdots \quad \boldsymbol{w}_d] \in \mathbb{R}^{N\times d}$ is a concatenation of sorted eigenvectors of $\boldsymbol{K}_x$.

Definition 2.2 (d-smooth function). *A function $\boldsymbol{f} \in \mathbb{R}^N$ is called "d-smooth on $\mathcal{M}_x$" if its d-truncated smoothness score with respect to $\boldsymbol{K}_x$ is equal to its squared 2-norm, i.e., $E_x^d(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2$.*

Remark 1. For any $\boldsymbol{f} \in \mathbb{R}^N$, we have

$$E_x^d(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2 \iff \boldsymbol{f} \in \operatorname{span}(\boldsymbol{W}_x).$$

Definition 2.3 (jointly smooth function). *A function $\boldsymbol{f} \in \mathbb{R}^N$ is called "jointly smooth on $\mathcal{M}_x$ and $\mathcal{M}_y$" if it is d-smooth with respect to both $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$, i.e., $E_x^d(\boldsymbol{f}) = E_y^d(\boldsymbol{f}) = \|\boldsymbol{f}\|_2^2$.*

Remark 2. Empirical functions, especially in the presence of observation noise, typically have a nonzero inner product with all eigenvectors of the kernel $\boldsymbol{K}_x$. Even though they would then not be smooth according to Definition 2.2, we can still compare them using our smoothness score (2.1) and say that $\boldsymbol{f}_i \in \mathbb{R}^N$ is smoother than $\boldsymbol{f}_j \in \mathbb{R}^N$ if $\|\boldsymbol{W}_x^T\boldsymbol{f}_i\|_2/\|\boldsymbol{f}_i\|_2 > \|\boldsymbol{W}_x^T\boldsymbol{f}_j\|_2/\|\boldsymbol{f}_j\|_2$.

3. Problem formulation.

Let $\mathcal{M}_x \subseteq \mathbb{R}^{d_x}$ and $\mathcal{M}_y \subseteq \mathbb{R}^{d_y}$ be two manifolds embedded in high-dimensional ambient spaces. We later extend this formulation to more than two manifolds. Consider two sets of observations $\{\boldsymbol{x}_i \in \mathcal{M}_x\}_{i=1}^N$ and $\{\boldsymbol{y}_i \in \mathcal{M}_y\}_{i=1}^N$ such that $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is a corresponding pair. In this work, we search for an orthogonal set of functions $\boldsymbol{f}_m \in \mathbb{R}^N$ such that each $\boldsymbol{f}_m$ is jointly smooth on $\mathcal{M}_x$ and $\mathcal{M}_y$. Note that in this discrete setting, the functions $\boldsymbol{f}_m \in \mathbb{R}^N$ are defined both on $\{\boldsymbol{x}_i \in \mathcal{M}_x\}_{i=1}^N$ and $\{\boldsymbol{y}_i \in \mathcal{M}_y\}_{i=1}^N$, but, in fact, they can be viewed as sampled versions of continuous (and smooth) functions $f_m^x : \mathcal{M}_x \to \mathbb{R}$ and $f_m^y : \mathcal{M}_y \to \mathbb{R}$ such that $f_m^x(\boldsymbol{x}_i) = f_m^y(\boldsymbol{y}_i) = \boldsymbol{f}_m[i]$. We posit that such smooth functions are of great interest since they represent and parametrize the common part between the two manifolds [28]. Therefore, the remainder of this paper revolves around the question, How can we find functions that are jointly smooth, given only the kernels $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$ associated with the observed data?

Figure 1 provides an illustrative example. Consider pairs of observations $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$ such that $\boldsymbol{x}_i \in \mathcal{M}_x \subset \mathbb{R}^2$ resides on a spiral in $\mathbb{R}^2$ and $\boldsymbol{y}_i \in \mathcal{M}_y \subset \mathbb{R}^3$ resides on a torus in $\mathbb{R}^3$ as depicted in Figure 1(a). Figure 1(b) and (c) present smooth functions on $\mathcal{M}_x$ and $\mathcal{M}_y$ that are not jointly smooth. Conversely, Figure 1(d) presents a jointly smooth function.

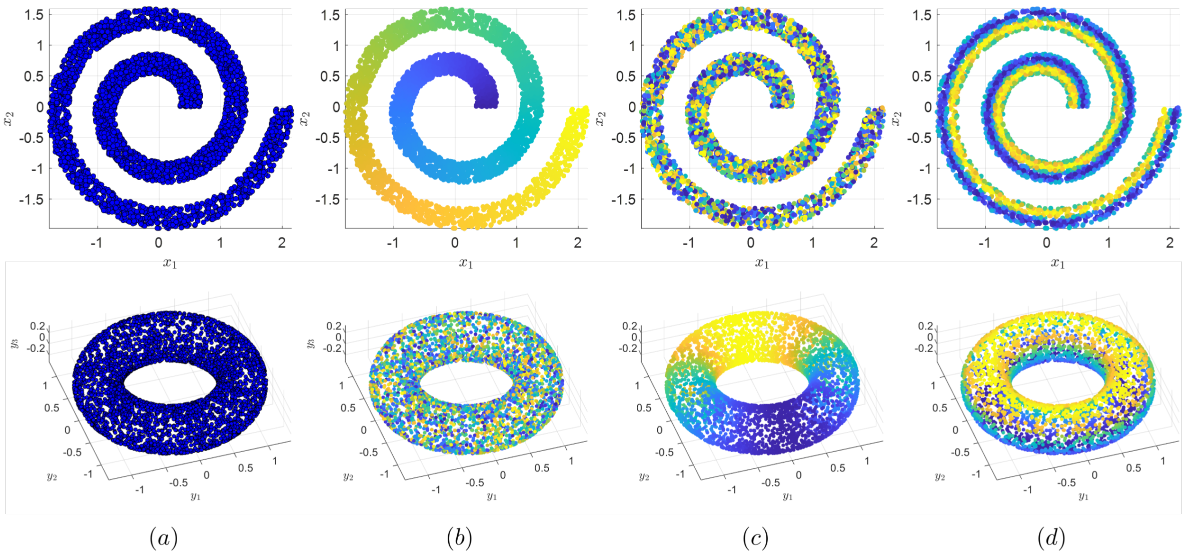


Figure 1. (a) *Points $\boldsymbol{x}_i \in \mathcal{M}_x$ on a spiral at the top row and points $\boldsymbol{y}_i \in \mathcal{M}_y$ on a torus at the bottom row, where each $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is a corresponding pair.* (b) *The two manifolds colored according to the same function $\boldsymbol{f}_x$, which is smooth on $\mathcal{M}_x$.* (c) *The two manifolds colored according to the same function $\boldsymbol{f}_y$, which is smooth on $\mathcal{M}_y$. We observe that $\boldsymbol{f}_x$ is smooth on $\mathcal{M}_x$, but it is not smooth on $\mathcal{M}_y$. Similarly, $\boldsymbol{f}_y$ is smooth on $\mathcal{M}_y$ but it is not smooth on $\mathcal{M}_x$.* (d) *The two manifolds colored according to the same jointly smooth function $\boldsymbol{f}_1$ on $\mathcal{M}_x$ and $\mathcal{M}_y$. For more details, see section 6.1.*

4. Proposed method.

The proposed method is based on the following lemma.

Lemma 4.1. *Consider $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{N\times d}$ such that $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}_d$. Let $\boldsymbol{W} := [\boldsymbol{A} \quad \boldsymbol{B}] \in \mathbb{R}^{N\times 2d}$. Then, the following decomposition of $\boldsymbol{W}$*

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

is an SVD such that

$$\left\|\boldsymbol{A}^T\boldsymbol{u}_i\right\|_2^2 = \left\|\boldsymbol{B}^T\boldsymbol{u}_i\right\|_2^2 = \frac{1}{2}\sigma_i^2 \qquad \forall i,$$

where $\boldsymbol{u}_i$ is the ith column of $\boldsymbol{U}$ and σ_i equals $\boldsymbol{\Sigma}[i,i]$ if $i \le 2d$ and 0 otherwise. In addition, $\boldsymbol{V} = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix}\boldsymbol{Q} & \boldsymbol{Q}\\ \boldsymbol{R} & -\boldsymbol{R}\end{smallmatrix}\right] \in \mathbb{R}^{2d\times 2d}$, $\boldsymbol{\Sigma}^2 = \left[\begin{smallmatrix}\boldsymbol{I}+\boldsymbol{\Gamma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}-\boldsymbol{\Gamma}\end{smallmatrix}\right] \in \mathbb{R}^{2d\times 2d}$, and $\boldsymbol{U} = \boldsymbol{W}\boldsymbol{V}\boldsymbol{\Sigma}^\dagger \in \mathbb{R}^{N\times 2d}$, where $\boldsymbol{\Sigma}^\dagger$ is the pseudoinverse of $\boldsymbol{\Sigma}$ and $\boldsymbol{A}^T\boldsymbol{B} = \boldsymbol{Q}\boldsymbol{\Gamma}\boldsymbol{R}^T \in \mathbb{R}^{d\times d}$ is an SVD.

The proof appears in the supplementary material refer section SM1.

The consequence of Lemma 4.1 in our setting is Corollary 4.2, which is derived by replacing the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ in Lemma 4.1 with the matrices $\boldsymbol{W}_x$ and $\boldsymbol{W}_y$, consisting of the first d dominant (normalized) eigenvectors of kernels $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$, respectively.

Corollary 4.2. *The most dominant left singular vectors $\boldsymbol{U} \in \mathbb{R}^{N\times d}$ of $\boldsymbol{W} := [\boldsymbol{W}_x \quad \boldsymbol{W}_y]$ satisfy*

$$\left\|\boldsymbol{W}_x^T\boldsymbol{u}_i\right\|_2^2 = \left\|\boldsymbol{W}_y^T\boldsymbol{u}_i\right\|_2^2 = \frac{1}{2}\sigma_i^2 = \frac{1}{2}\left(1+\gamma_i\right), \qquad i \le d,$$

where $\gamma_i = \boldsymbol{\Gamma}[i,i]$.

By definition, γ_i are the singular values of $\boldsymbol{W}_x^T\boldsymbol{W}_y$ and, thus, are in fact the cosine of the principal angles θ_i between the two subspaces spanned by $\boldsymbol{W}_x$ and $\boldsymbol{W}_y$, that is, $\gamma_i = \cos(\theta_i)$, where $0 \leq \gamma_i \leq 1$. Therefore, for each singular value $\gamma_i = 1$, there exists a common direction between the two subspaces. In addition, directions that are not common are associated with $\gamma_i < 1$, and the closer γ_i is to the value 1, the smaller the angle is between the two respective directions.

By Corollary 4.2, only when there exists a common direction, the respective singular value $\gamma_i = 1$, $\sigma_i^2 = 2$, and $\boldsymbol{u}_i$ strictly satisfies Definition 2.3 of jointly smooth functions. This typically holds for the constant vector $\boldsymbol{u}_1$, satisfying $\|\boldsymbol{W}_x^T\boldsymbol{u}_1\|_2^2 = \|\boldsymbol{W}_y^T\boldsymbol{u}_1\|_2^2 = 1$ independent of the particular kernels $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$. Other nondegenerate singular vectors representing common directions may exist, yet, due to noise and other possible distortions, it is likely that the obtained singular vectors $\boldsymbol{u}_i$ will not be strictly smooth, that is, $\|\boldsymbol{W}_x^T\boldsymbol{u}_i\|_2^2 < 1$. Still, by Corollary 4.2, the most dominant left singular vectors of $\boldsymbol{U}$ are ordered from the "most" jointly smooth function to the least with respect to their d-truncated smoothness score. This implies that taking the top M columns $\boldsymbol{u}_i \in \mathbb{R}^N$ of $\boldsymbol{U}$ accomplishes the goal of finding jointly smooth functions. The entire procedure is summarized in Algorithm 4.1. Overall, the output of Algorithm 4.1 is smooth functions $\boldsymbol{f}_i$, which are the top M left singular vectors $\boldsymbol{u}_i$.

Remark 3. When the compact SVD of $\boldsymbol{W} := [\boldsymbol{W}_x \quad \boldsymbol{W}_y]$ is unique, any method of computing the SVD can be applied. However, when the compact SVD is not unique, one needs to construct the SVD decomposition as prescribed by the second part of Lemma 4.1 and implemented in steps 3 and 4 of Algorithm 4.1. Importantly, computing this specific SVD rather than applying a generic SVD algorithm is computationally efficient since it requires us to decompose $\boldsymbol{W}_x^T\boldsymbol{W}_y \in \mathbb{R}^{d\times d}$ instead of $\boldsymbol{W} \in \mathbb{R}^{N\times 2d}$, where $d < N$.

4.1. Choosing the value of M. The discussion above implies that the obtained functions $\boldsymbol{u}_i$ might not be strictly smooth, that is, $\|\boldsymbol{W}_x^T\boldsymbol{u}_i\|_2^2 < 1$. Hence, in Algorithm 4.1 we use the top M functions. The number of functions M is therefore a hyperparameter of the algorithm that needs to be set a priori. We propose to set M as the number of functions $\boldsymbol{u}_i$ satisfying $\|\boldsymbol{W}_x^T\boldsymbol{u}_i\|_2^2 > E_0$, where E_0 is a suitable threshold. Similarly to the jackstraw method [12], one can permute one of the observations, e.g., $\{(\boldsymbol{x}_i, \boldsymbol{y}_{\pi(i)})\}$, where π is a random permutation, and then compute the matrix $\tilde{\boldsymbol{\Gamma}}$ from Lemma 4.1. Except for the first trivial singular value $\tilde{\gamma}_1 = 1$, the rest of the singular values $\tilde{\gamma}_i$ and the singular vectors $\tilde{\boldsymbol{u}}_i$ are associated with pure noise observations. Thus, the second singular value $\tilde{\gamma}_2$ provides us a threshold to the original singular values γ_i, that is, $\gamma_i > \tilde{\gamma}_2$, which leads to the following threshold:

$$E_0 = \frac{1}{2}\left(1 + \tilde{\gamma}_2\right).$$

Alternatively, if we assume random (independent) observations we can provide an estimation for $\tilde{\gamma}_2$ since we can write $\tilde{\gamma}_2 = \cos(\theta_2)$, where θ_2 is the smallest principal angle between two random subspaces uniformly distributed (again, ignoring the first trivial constant subspace). Based on [25], we derived our proposed threshold

$$E_0 = \frac{1}{2} + \frac{\sqrt{d - \frac{1}{2}}\sqrt{N - d - \frac{1}{2}}}{N - 1}.$$

See the supplementary material refer section SM2 for the derivation of the analytical formula for $\cos(\theta_2)$ that leads to this expression for E_0.

As an intuitive explanation for the bound E_0, one can think of a boundary between order and randomness. The heuristic computation using the jackstraw method for $\tilde{\gamma}_2$ may help to illustrate this point: when there are no jointly smooth features by construction, because one set of observations is permuted deliberately, the second singular value cannot indicate any commonality (because there is none). That means we can use it as an indicator, or bound, for "noise." The value of E_0 then is the average of 1 (i.e., $\tilde{\gamma}_1$) and this noise indicator.

An interpretation may be less intuitive in the limit of large data ($N \to \infty$, where E_0 equals 0.5, given in the formula above). We must note that we did not prove the following, it is just our current intuition: the singular vectors that approximate jointly smooth features turn into eigenfunctions. If there are common features between the two (or more) sets of inputs, the jointly smooth features $\mathbf{u}_i$ constitute a full set of (countably infinite) eigenfunctions on the common manifold. All associated singular values will be exactly one (they are not eigenvalues of the Laplace–Beltrami operator but the infinite limit of the combined matrix $\mathbf{W}$). However, there is also another countably infinite set of singular values that is exactly zero, corresponding to the uncommon directions, and occurring because $\mathbf{W}$ is not square. The limit of 0.5 for E_0 thus separates this countably infinite set of ones from the countably infinite set of zeros.

4.2. Multiple manifolds (multiple views). The proposed method can be extended to support multiple manifolds in a straightforward manner. While Lemma 4.1 is not valid for more than two manifolds, numerical tests consistently show that the naive extension provides satisfactory results. Consider $K > 2$ sets of observations $\{\boldsymbol{x}_i^{(1)} \in \mathcal{M}_1\}_{i=1}^N, \{\boldsymbol{x}_i^{(2)} \in \mathcal{M}_2\}_{i=1}^N, \dots, \{\boldsymbol{x}_i^{(K)} \in \mathcal{M}_K\}_{i=1}^N$, where each $(\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \dots, \boldsymbol{x}_i^{(K)})$ is a tuple of aligned observations. For $k = 1, \dots, K$, let $\boldsymbol{W}_k \in \mathbb{R}^{N \times d}$ be the d dominant eigenvectors of the kernel $\boldsymbol{K}_k$ constructed from the observations of the kth manifold. Denote $\boldsymbol{W} := [\boldsymbol{W}_1 \quad \boldsymbol{W}_2 \quad \dots \quad \boldsymbol{W}_K] \in \mathbb{R}^{N \times Kd}$. Similar to Corollary 4.2, we propose to compute the top M left singular vectors of $\boldsymbol{W}$ and view them as the jointly smooth functions on all K manifolds. The entire extension is outlined in Algorithm 4.2. See section 6.2 for supporting numerical results.

4.3. Learning the common manifold. Similar to the idea of alternating diffusion maps [28], we can use jointly smooth functions as embedding coordinates for the common manifold between the sensor data. The minimal number of embedding coordinates depends on the intrinsic dimension (bounds given by the theorem of Whitney [43]) and the topology of the common manifold. For example, a circle with intrinsic dimension of one requires at least two coordinates to embed it in a Euclidean space. In our case, these coordinates would correspond to two jointly smooth functions.

Our algorithms to extract jointly smooth features are not suited to construct a minimal, parsimonious set of coordinates, nor are they designed to be invariant to the data density. For example, if we obtain more data for one sensor in a certain region, the embedding into the jointly smooth features will be very different. Thus, in practice, we first compute all jointly smooth functions that have a score larger than E_0 (see section 4.1). Then, we use nonlinear manifold learning algorithms such as diffusion maps [13] to obtain invariance to data density. Lastly, we employ eigenvector selection algorithms such as local linear regression [17, 11] to select good embedding coordinates. We demonstrate this approach on an example in section 6.3.

4.4. Analysis of computational complexity and efficient implementation. Algorithms 4.1 and 4.2 are kernel methods that employ as many eigendecompositions as there are data sets and subsequently involve one SVD to extract jointly smooth features. For data sets with a large number N of data points, numerical methods implementing the algorithms have to be chosen carefully. Otherwise, a brute-force approach scales with a computational complexity of $O(N^3)$ for the eigendecompositions and SVD computations and a memory requirement of $O(N^2)$ due to the storage of the full kernel matrices for each data set. In general, for dense matrices and a complete set of eigenvectors, the best achievable computational complexity is $O(N^\omega)$, $\omega \approx 2.376$ [15]. However, we can exploit more structure and sparse matrices to bring the computational and memory cost down to an almost linear scaling. With efficient distance computations, sparse matrix storage, and numerical eigensolvers that can handle such data structures, we can reduce the computational efficiency. Let d be the number of

---

Algorithm 4.1. Jointly smooth functions from 2 observations

```
Input: 2 observations $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$, where $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$ and $\boldsymbol{y}_i \in \mathbb{R}^{d_y}$.
Output: $M$ joinly smooth functions $\{\boldsymbol{f}_m \in \mathbb{R}^N\}_{m=1}^M$.
    1. Compute the kernels $\boldsymbol{K}_x, \boldsymbol{K}_y \in \mathbb{R}^{N\times N}$:
       $\boldsymbol{K}_x[i,j] = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2\sigma_x^2}\right), \quad \boldsymbol{K}_y[i,j] = \exp\left(-\frac{\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2}{2\sigma_y^2}\right).$
    2. Compute $\boldsymbol{W}_x, \boldsymbol{W}_y \in \mathbb{R}^{N\times d}$, the first $d$ eigenvectors of $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$ associated with the
       largest eigenvalues.
    3. Compute the SVD decomposition: $\boldsymbol{W}_x^T \boldsymbol{W}_y = \boldsymbol{Q\Gamma R}^T \in \mathbb{R}^{d\times d}$.
    4. Compute $\boldsymbol{U} = \frac{1}{\sqrt{2}}[\,\boldsymbol{W}_x \ \boldsymbol{W}_y\,][\begin{smallmatrix}\boldsymbol{Q} & \boldsymbol{Q}\\ \boldsymbol{R} & -\boldsymbol{R}\end{smallmatrix}][\begin{smallmatrix}\boldsymbol{I}+\boldsymbol{\Gamma} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{I}-\boldsymbol{\Gamma}\end{smallmatrix}]^{-\frac{1}{2}} \in \mathbb{R}^{N\times 2d}$.
    5. Set $\boldsymbol{f}_m$ to the $m$th column of $\boldsymbol{U}$.
```

Algorithm 4.2. Jointly smooth functions from K observations

```
Input: $K$ observations $\{\boldsymbol{x}_i^{(1)}, \boldsymbol{x}_i^{(2)}, \dots \boldsymbol{x}_i^{(K)}\}_{i=1}^N$, where $\boldsymbol{x}_i^{(k)} \in \mathbb{R}^{d_k}$.
Output: $M$ jointly smooth functions $\{\boldsymbol{f}_m \in \mathbb{R}^N\}_{m=1}^M$.
    1. For each observation set $\{\boldsymbol{x}_i^{(k)}\}_{i=1}^N$ compute the kernel
       $\boldsymbol{K}_k[i,j] = \exp\left(-\frac{\|\boldsymbol{x}_i^{(k)} - \boldsymbol{x}_j^{(k)}\|_2^2}{2\sigma_k^2}\right).$
    2. Compute $\boldsymbol{W}_k \in \mathbb{R}^{N\times d}$, the first $d$ eigenvectors of $\boldsymbol{K}_k$.
    3. Set $\boldsymbol{W} := [\boldsymbol{W}_1, \boldsymbol{W}_2, \dots, \boldsymbol{W}_K] \in \mathbb{R}^{N\times Kd}$.
    4. Compute the SVD decomposition: $\boldsymbol{W} = \boldsymbol{U\Sigma V}^T$.
    5. Set $\boldsymbol{f}_m$ to be the $m$th column of $\boldsymbol{U}$.
```

eigenvectors we compute per kernel, and let M be the number of jointly smooth functions; then the computational time scales with $O(N\log(N))$, and the memory requirement can be reduced to $O((2d{+}M)N)$ for the final result, with $O(dN{+}d^2)$ during runtime of the iteratively restarted Arnoldi algorithm [41]. The kernel matrices can be kept sparse by using a cutoff for the exponentially decaying Gaussian kernels or by employing (continuous) nearest-neighbor kernels [8]. The factor of $O(N\log(N))$ occurs in the computation of the (k-nearest neighbor) kernel matrix, not in the eigenvector computations. The latter scales with $O(N)$, as long as the matrix-vector products can be computed in linear time (which is possible because of sparsity) and the number of Lanczos vectors is kept fixed; see [41, p. 135]. Note that there are many other approaches to good computational scaling of kernel methods [30, 39].

The following numerical methods were applied:

1. We compute the continuous nearest-neighbor kernel (with parameters $k = 25$ and $\delta = 1.0$ as defined in [8]) instead of the squared-exponential kernel. The limiting operator is the same (i.e., in the large data limit, the function spaces are the same), but the kernel can be computed much faster using SciPy's k-d-tree implementation to obtain the nearest neighbors of each data point.
2. We use the efficient, sparse eigensolver for Hermitian matrices implemented in SciPy to obtain $d = 100$ eigenvectors of each kernel.
3. We then extract $M = 10$ jointly smooth functions $\boldsymbol{f}_m$ using the sparse implementation of the SVD on the merged eigenvectors.

The time and memory efficiency of these algorithms is demonstrated for the toy problem in section 6.1.

5. Extending jointly smooth functions to new data points.

Consider a new (unseen) set of pairs of observations $\{(\boldsymbol{x}_i^\star, \boldsymbol{y}_i^\star)\}_{i=1}^{N^\star}$. We wish to estimate the evaluation of the computed jointly smooth functions in Algorithm 4.1 (or in Algorithm 4.2) on this new set, namely, to estimate new vector coordinates $\boldsymbol{f}_m^\star[i]$ or equivalently the values $f_m^x(\boldsymbol{x}_i^\star)$ and $f_m^y(\boldsymbol{y}_i^\star)$. Clearly, we can append the new points to the existing set of points and reapply Algorithm 4.1 to the extended set. In this section, we propose instead a more efficient estimate of $\boldsymbol{f}_m^\star[i]$ that can be implemented in an online manner, supporting incoming streaming data. The proposed extension is based on the Nyström method, which has been extensively used for out-of-sample extension in the context of kernel methods [14, 6, 19, 44]. Consider a new pair of data $(\boldsymbol{x}_i^\star, \boldsymbol{y}_i^\star)$. Let

$$\boldsymbol{\alpha}_x^m = \boldsymbol{W}_x^T \boldsymbol{f}_m \in \mathbb{R}^{d\times 1}$$

be the expansion coefficients of $\boldsymbol{f}_m$ (obtained by Algorithm 4.1) in the basis $\boldsymbol{W}_x$. Recall that $\boldsymbol{W}_x$ consists of eigenvectors of $\boldsymbol{K}_x$, as defined in section 2, that is, $\boldsymbol{K}_x\boldsymbol{W}_x = \boldsymbol{W}_x\boldsymbol{\Lambda}_x$. The matrix $\boldsymbol{\Lambda}_x$ is zero, except for the sorted list of eigenvalues λ_i on its diagonal. Using the Nyström extension, we extend the eigenvectors by

$$\begin{bmatrix} \ulcorner & & \urcorner \\ & \boldsymbol{K}_x & \\ \llcorner & & \lrcorner \\ — & \boldsymbol{k}_x^\star & — \end{bmatrix} \begin{bmatrix} \\ \boldsymbol{W}_x \\ \\ \end{bmatrix} = \begin{bmatrix} \ulcorner & & \urcorner \\ & \boldsymbol{W}_x & \\ \llcorner & & \lrcorner \\ — & \boldsymbol{w}_x^\star & — \end{bmatrix} \begin{bmatrix} \\ \boldsymbol{\Lambda}_x \\ \\ \end{bmatrix},$$

Algorithm 5.1. Out-of-sample extension

```
Input: A set of new observations $\{(\boldsymbol{x}_i^\star, \boldsymbol{y}_i^\star)\}_{i=1}^{N^\star}$.
Output: The out-of-sample extension $f_m^x(\boldsymbol{x}_i^\star)$ and $f_m^y(\boldsymbol{y}_i^\star)$ $\forall i$.
  1. Compute the coefficients: $\boldsymbol{\alpha}_x^m = \boldsymbol{W}_x^T \boldsymbol{f}_m \in \mathbb{R}^{d\times 1}$.
  2. Extend the eigenvectors: $\boldsymbol{W}_x^\star = \boldsymbol{K}_x^\star \boldsymbol{W}_x \boldsymbol{\Lambda}_x^{-1} \in \mathbb{R}^{N^\star \times d}$.
     where $\boldsymbol{K}_x^\star \in \mathbb{R}^{N^\star \times N}$ such that $\boldsymbol{K}_x^\star[i,j] = \exp(-\frac{\|\boldsymbol{x}_i^\star - \boldsymbol{x}_j\|_2^2}{2\sigma_x^2})$.
  3. Repeat steps 1 and 2 for the $\{\boldsymbol{y}_i^\star\}$ observations.
  4. Set $\boldsymbol{f}_m^\star = \frac{1}{2}(\boldsymbol{W}_x^\star \boldsymbol{\alpha}_x^m + \boldsymbol{W}_y^\star \boldsymbol{\alpha}_y^m) \in \mathbb{R}^{N^\star}$ such that $f_m^x(\boldsymbol{x}_i^\star) = f_m^y(\boldsymbol{y}_i^\star) = \boldsymbol{f}_m^\star[i]$.
```

where $[\text{—}\ \ \boldsymbol{k}_x^\star\ \text{—}] \in \mathbb{R}^{1\times d}$ and $\boldsymbol{k}_x^\star[j] = \exp(-\frac{\|\boldsymbol{x}_i^\star - \boldsymbol{x}_j\|_2^2}{2\sigma_x^2})$ is computed from the existing and new data. In other words, the new coordinates of the basis vectors (eigenvectors) are given by

$$\boldsymbol{w}_x^\star = \boldsymbol{k}_x^\star \boldsymbol{W}_x \boldsymbol{\Lambda}_x^{-1} \in \mathbb{R}^{1\times d},$$

and the induced out-of-sample extension is given by

$$\boldsymbol{f}_m^\star[i] = \boldsymbol{w}_x^\star \boldsymbol{\alpha}_x^m.$$

Note that the procedure above is based on the computation of $\boldsymbol{k}_x^*$ using $\boldsymbol{x}_i^*$ and $\{\boldsymbol{x}_i\}_{i=1}^N$. Similarly, this procedure can be implemented based on $\boldsymbol{y}_i^\star$ and $\{\boldsymbol{y}_i\}_{i=1}^N$. Combining these two alternatives based on the constraint that $\boldsymbol{f}_m^\star[i] = f_m^x(\boldsymbol{x}_i^\star) = f_m^y(\boldsymbol{y}_i^\star)$ gives rise to the following extension:

$$\boldsymbol{f}_m^\star[i] = \frac{1}{2}\left(\boldsymbol{w}_x^\star \boldsymbol{\alpha}_x^m + \boldsymbol{w}_y^\star \boldsymbol{\alpha}_y^m\right).$$

Considering other combinations, for instance, taking a weighted mean value, will be the subject of future work. The entire out-of-sample extension algorithm is given in Algorithm 5.1. As pointed out to us by an anonymous reviewer, one may analyze high-dimensional data by splitting it into multiple, lower-dimensional components and then recombining them through the Nyström procedure outlined here. This may improve kernel regression that would normally be done on the full, high-dimensional data, and we will pursue this direction in the future.

6. Experimental results.

Our source code for all experiments is publicly available.[1]

6.1. Toy problem.

We revisit the toy problem presented in section 3 and start by giving more details. Consider triplets $(z_i, \epsilon_i, \eta_i) \sim U[0,1]^3$ that are independent and identically distributed and uniformly distributed in the unit cube. The 2D spiral in $\mathbb{R}^2$ is given by

$$\boldsymbol{x}_i = g(z_i, \epsilon_i) = \begin{bmatrix} \left(\frac{3}{2}\epsilon_i + \frac{z_i}{3} + \frac{2}{3}\right)\cos(4\pi\epsilon_i) \\ \left(\frac{3}{2}\epsilon_i + \frac{z_i}{3} + \frac{2}{3}\right)\sin(4\pi\epsilon_i) \end{bmatrix} \tag{6.1}$$

such that z_i controls the width of the ribbon. The 2-torus in $\mathbb{R}^3$ is given by

$$\boldsymbol{y}_i = h(z_i, \eta_i) = \begin{bmatrix} \left(1 + \frac{1}{3}\cos(2\pi z_i)\right)\cos(2\pi\eta_i) \\ \left(1 + \frac{1}{3}\cos(2\pi z_i)\right)\sin(2\pi\eta_i) \\ \frac{1}{3}\sin(2\pi z_i) \end{bmatrix} \tag{6.2}$$

[1] See https://gitlab.com/felix.dietrich/JointlySmoothFunctions.

such that z_i controls the smaller angle. Note that z_i is a common variable observed by both observation functions g and h, whereas ϵ_i and η_i are variables captured by only a single observation.

We generate the set $\{(z_i, \epsilon_i, \eta_i)\}_{i=1}^{N}$ consisting of 4,100 realizations of the triplets $(z_i, \epsilon_i, \eta_i) \sim U[0,1]^3$. We keep $N^\star = 100$ points aside for the out-of-sample extension validation and construct the two views $\boldsymbol{x}_i = f(z_i, \epsilon_i)$ and $\boldsymbol{y}_i = g(z_i, \eta_i)$ as in (6.1) and (6.2) based on the remaining $N = 4{,}000$ points. Let $\boldsymbol{K}_x$ and $\boldsymbol{K}_y$ be the kernels associated with the observations $\{\boldsymbol{x}_i \in \mathcal{M}_x\}_i$ and $\{\boldsymbol{y}_i \in \mathcal{M}_y\}_i$, respectively. We apply Algorithm 4.1, where σ_x and σ_y are set to be 30% of the median of the pairwise Euclidean distances, that is, $\sigma_x = 0.3 \cdot \text{median}(\{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2\}_{i,j})$, and similarly for σ_y. We also set $d = \frac{N}{4} = 1{,}000$ and we report that any value between 500–2,000 provides similar empirical results. The reason so many eigenvectors are needed for an accurate approximation is that we deliberately configured the common variable z to be much less dominant than the sensor specific ones (ϵ and η). This means the top eigenvectors of each kernel parameterize mostly the sensor specific data; thus, they will not be able to accurately represent z. In general, if the common variable is more dominant, fewer eigenvectors are needed to extract it. Figure 2(a) presents the two manifolds. At the top row of Figure 2(b), we depict the top three (nontrivial) jointly smooth functions obtained by Algorithm 4.1 and the out-of-sample extension obtained by Algorithm

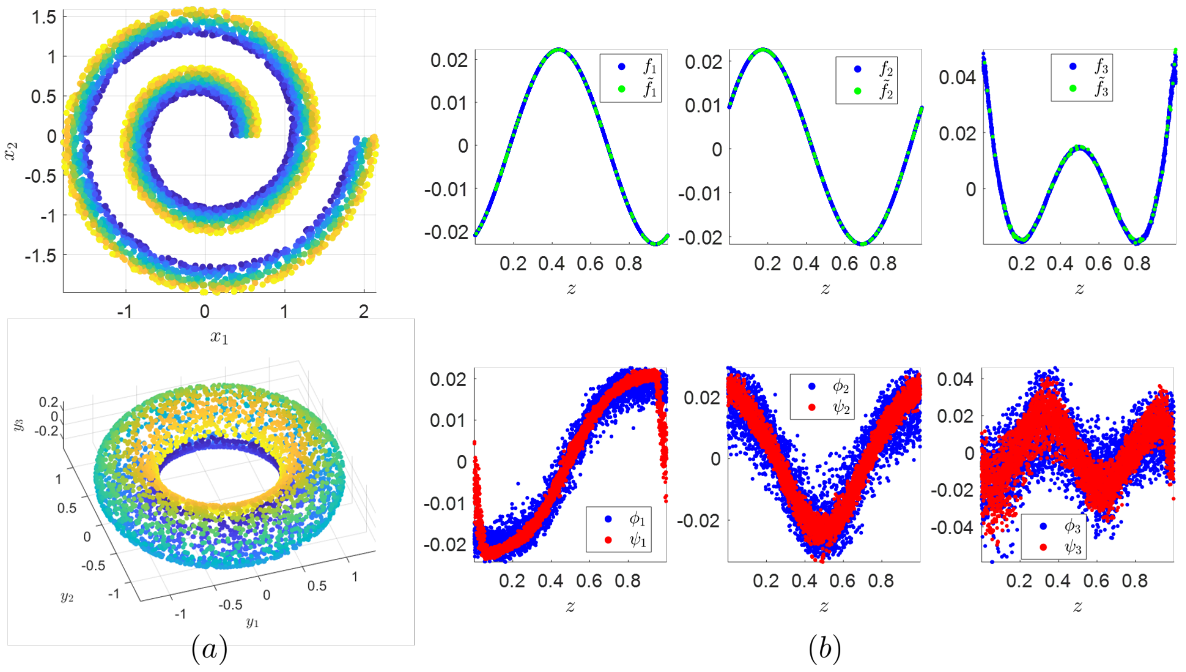


Figure 2. (a) *The two manifolds colored with respect to the common variable z. The blue points at the top row of* (b) *are scatter of the top three (nontrivial) functions f_i obtained by Algorithm 4.1 plotted against the common variable z. The green points are obtained by the out-of-sample extension of f_i to $\tilde{f}_i$, using Algorithm 5.1. The bottom row of* (b) *consists of the top three (nontrivial) left ($\boldsymbol{\phi}_i$) and right ($\boldsymbol{\psi}_i$) singular vectors obtained by NCCA plotted against the common variable z. Recall that in NCCA the left singular vectors correspond to the spiral $\mathcal{M}_x$, and the right singular vectors correspond to the torus $\mathcal{M}_y$.*

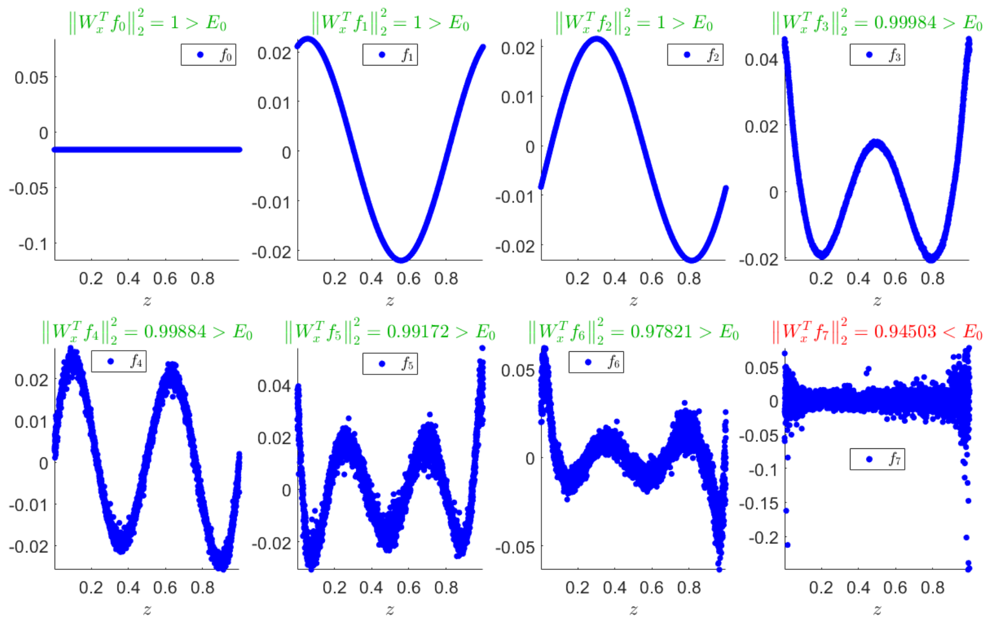


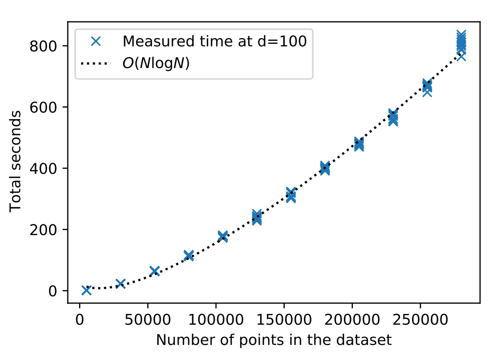
Figure 3. *The top 8 jointly smooth functions obtained by Algorithm 4.1 plotted against the common variable z. Above each plot we depict the smoothness value $\|\boldsymbol{W}_x^T \boldsymbol{f}_i\|_2^2$ and compare it to the threshold $E_0 = 0.9558$ obtained by the jackstraw method described in section 4.1.*

5.1. The bottom row of Figure 2(b) displays the top three (nontrivial) left ($\boldsymbol{\phi}_i$) and right ($\boldsymbol{\psi}_i$) singular vectors obtained by NCCA [32]. We observe that Algorithm 4.1 provides functions which represent the common variable z more accurately in comparison to NCCA.

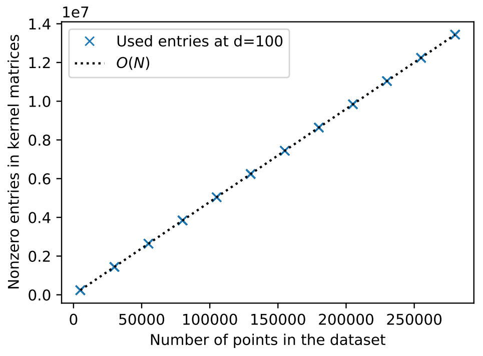
Figure 3 depicts the top 8 jointly smooth functions obtained by Algorithm 4.1. Above each plot we present the smoothness value $\|\boldsymbol{W}_x^T \boldsymbol{f}_i\|_2^2$ and compare it to the threshold $E_0 = 0.9558$ obtained using the jackstraw method as discussed in section 4.1. We observe that indeed the threshold allows us to accurately detect the number of components M correlated with the common variable z. If a common variable such as z is available, the local linear regression algorithm [17, 11] can be employed to quantify the accuracy of an extracted feature.

In Figure 4, we demonstrate that the time and memory complexity of the algorithm actually follows the predicted curves. The data were created for the given toy problem, with a varying number of points. In Figure 5, the number of points in the data set was fixed to $N = 50{,}000$, and we demonstrate how the algorithms scale when the number of eigenvectors d per kernel is changed. As expected, both memory and computational times scale linearly.

6.2. Sleep stage identification. We apply Algorithm 4.2 (multiview) to real physiological signals, addressing the problem of sleep stage identification. The data are available online [21] and described in detail in [42]. The data contain multimodal recordings from several subjects, where each subject is recorded for about 10 hours during a single night of sleep. The data were analyzed by a human expert and divided into six sleep stages: awake, rapid eye movement, stage 1 (shallow sleep), stage 2, stage 3, and stage 4 (deep sleep). The types of sensors used for recording vary between the different subjects. Here, we use a subset of six sensors common

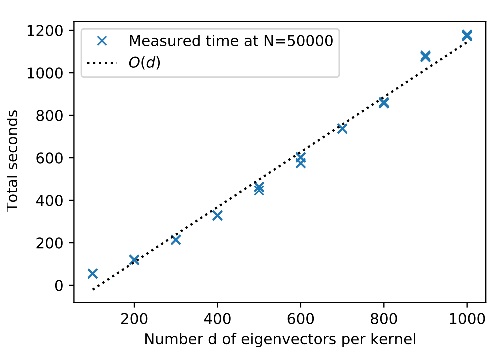


(a)

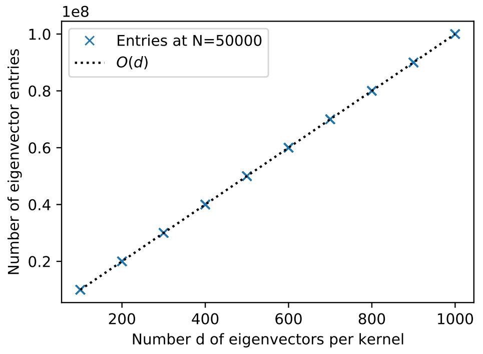


(b)

Figure 4. *Scaling of* (a) *computation time and* (b) *memory, with the number of points N in the data set changing at a constant number of eigenvectors $d = 100$, for Algorithm 4.1 with efficient kernel and SVD algorithms. The actual curve fitted to the data in panel* (a) *is* $0.0013455Nlog(N) - 0.0141995N + 26.9836555$. *Every experiment was repeated* 10 *times.*



(a)



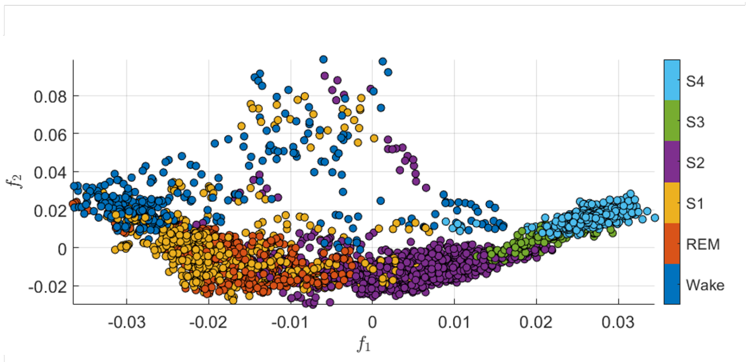
(b)

Figure 5. *Scaling of* (a) *computation time and* (b) *memory, with the number of eigenvectors d per kernel at $N = 50{,}000$, for Algorithm 4.1 with efficient kernel and SVD algorithms. The actual curve fitted to the data in panel* (a) *is* $1.2945669d - 150.0280786$. *Every experiment was repeated* 10 *times.*

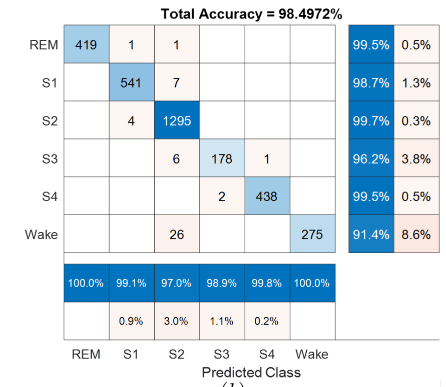
to all reported subjects: four electroencephalogram (EEG) channels, one electromyography (EMG) channel, and one electrocardiography (ECG) channel, sampled at 512 Hz.

Let $\boldsymbol{K}_i^{(k)}$ be the kernel constructed from the ith channel of the kth subject. For details on the construction of the kernels from the measured data, see the supplementary material (Supplement.pdf [local/web 271KB]). We apply Algorithm 4.2 to the six kernels with $d = 1{,}000$ and obtain the top $M = 20$ common functions $\boldsymbol{f}_m$. We note that about 50 functions could be used, but we take only the top 20 to make a fair comparison with KCCA, which attains only 20 relevant canonical directions. We tested KCCA using the top 20, 30, and 50 eigenvectors and report that the best results (in the sense of the average of 10-fold cross-validation) were obtained by using only the top 20.

Figure 6(a) presents the scatter of the top two (nontrivial) functions obtained by Algorithm 4.2 applied to subject 2. We observe that, solely from the top two jointly smooth functions,



(a)

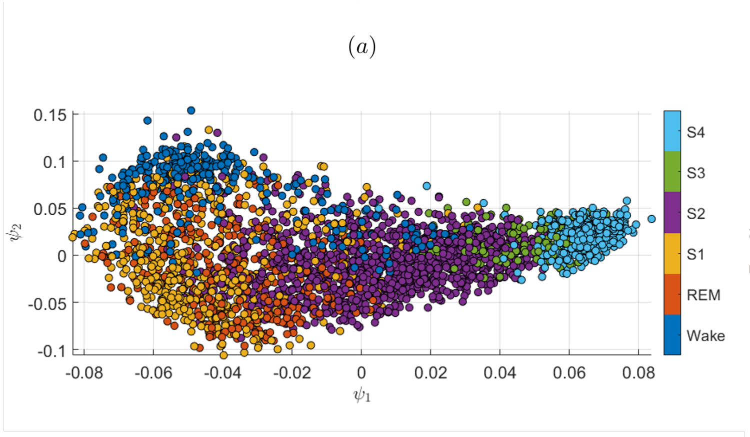


Total Accuracy = 98.4972%

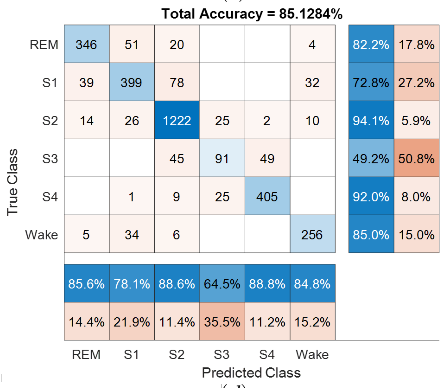
True Class	REM	S1	S2	S3	S4	Wake		
REM	419	1	1				99.5%	0.5%
S1		541	7				98.7%	1.3%
S2		4	1295				99.7%	0.3%
S3			6	178	1		96.2%	3.8%
S4				2	438		99.5%	0.5%
Wake			26			275	91.4%	8.6%
	100.0%	99.1%	97.0%	98.9%	99.8%	100.0%		
		0.9%	3.0%	1.1%	0.2%			

Predicted Class

(b)



(c)



Total Accuracy = 85.1284%

True Class	REM	S1	S2	S3	S4	Wake		
REM	346	51	20			4	82.2%	17.8%
S1	39	399	78			32	72.8%	27.2%
S2	14	26	1222	25	2	10	94.1%	5.9%
S3			45	91	49		49.2%	50.8%
S4		1	9	25	405		92.0%	8.0%
Wake	5	34	6			256	85.0%	15.0%
	85.6%	78.1%	88.6%	64.5%	88.8%	84.8%		
	14.4%	21.9%	11.4%	35.5%	11.2%	15.2%		

Predicted Class

(d)

Figure 6. (a) *A scatter plot of the top two jointly smooth functions obtained by Algorithm* 4.2. *The points are colored according to the different sleep stages as identified by a human expert.* (b) *The confusion matrix obtained by a* 10*-fold kernel SVM classifier applied to the top* $M = 20$ *jointly smooth functions obtained by Algorithm* 4.2. (c) *A scatter plot of the top two principal directions obtained by a multiview KCCA equipped with the same kernels as in* (a) *and colored according to the sleep stage.* (d) *The confusion matrix obtained by a* 10*-fold kernel SVM classifier applied to the top* $M = 20$ *principal directions obtained by multiview KCCA.*

we obtain a meaningful representation of the sleep stages. Namely, shallow sleep stages reside on the left side of the scatter plot, and deep sleep stages reside to the right. We quantify the (unsupervised) separation obtained by the top 20 jointly smooth functions using a kernel support vector machine (SVM) classifier equipped with a Gaussian kernel. The 10-fold cross-validation confusion matrix is displayed in Figure 6(b). Note that we apply a (nonlinear) kernel SVM method on eigenvectors of a kernel. This is unnecessary if enough eigenvectors are extracted, and a linear classifier would suffice. Still, in this example, 20 vectors are not enough to linearly span the classification function accurately. For comparison, we apply a multiview KCCA [48] to the same six kernels. Figure 6(c) presents the scatter of the top two (nontrivial) eigenvectors obtained by KCCA applied to subject 2. Figure 6(d) displays the 10-fold confusion matrix obtained using a kernel SVM applied to the top 20 eigenvectors attained by KCCA. As can be seen both qualitatively and quantitatively, the jointly smooth functions obtained using Algorithm 4.2 better parametrize the sleep stages. We also compare the classification results to (i) the representation obtained by each kernel (sensor) separately

Table 1

Sleep stage identification accuracy [%].

	EEG1	EEG2	EEG3	EEG4	EMG	ECG	All	KCCA	Alg. 4.2
Subject 1	78.73	78.54	76.03	74.77	69.34	81.71	96.27	88.61	**97.52**
Subject 2	80.18	78.02	78.33	76.61	79.05	79.21	93.17	85.12	**98.49**
Subject 3	78.68	77.61	79.74	81.53	79.84	76.30	89.67	81.59	**98.18**
Subject 4	75.39	78.87	76.55	77.52	76.83	70.76	94.99	86.60	**97.68**
Mean	78.25	78.26	77.66	77.61	76.27	77.00	93.53	85.48	**97.97**

and (ii) the representation obtained by concatenating all six feature vectors into one vector. We repeat this experiment for 4 subjects. Table 1 summarizes the results. In each row, we mark in bold the the highest obtained accuracy, which is consistently achieved by the proposed algorithm. We note that that we compare to KCCA rather than NCCA because NCCA is restricted to two views.

6.3. Manifold learning for high-dimensional input data. We now demonstrate how to obtain jointly smooth functions and a common latent space in an experiment involving high-dimensional observations. In this simulation, two cars drive clockwise around a single track, both with their individual but constant speeds. One set of observations is gathered with a camera located on one of the cars; another data set is gathered with a camera observing the entire track. Figure 7 shows the setup for this experiment, together with four example snapshots of the two data sets.

We simultaneously gather 11,100 frames with 400×300 grayscale pixels from both cameras while the two cars drive at constant speed around the track, with 100 frames per loop for the blue car and 111 frames per loop for the yellow car. We preprocess the video data with the following pipeline:

1. Scale down each frame to 15×15 pixels using `PIL.Image.NEAREST` resampling from Python `pillow`.
2. Project the data onto its first 10 principal components.

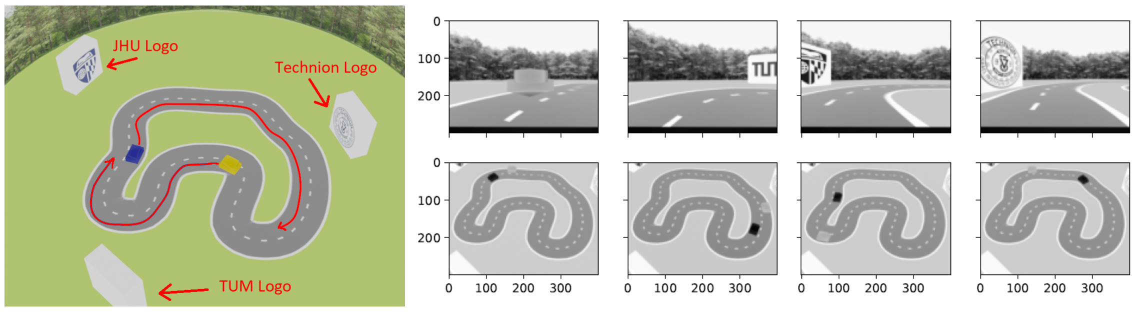


Figure 7. *Left panel: setting for the car race example. Two cars (blue and yellow) drive clockwise around a single track, both with constant speed (yellow: 100 frames per loop, blue: 111 frames per loop). Right panel: one set of observations is gathered with a camera located on the blue car (grayscale, top row with four example snapshots) and another set with a camera observing the entire track (grayscale, bottom row, with corresponding example snapshots).*

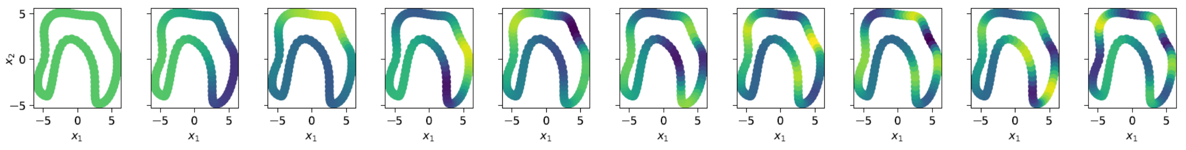


Figure 8. *Jointly smooth functions plotted over the positions of the blue car. Even though there are several smooth functions available, the underlying manifold is low-dimensional, and manifold learning can reveal this in a subsequent analysis (see Figure 9).*

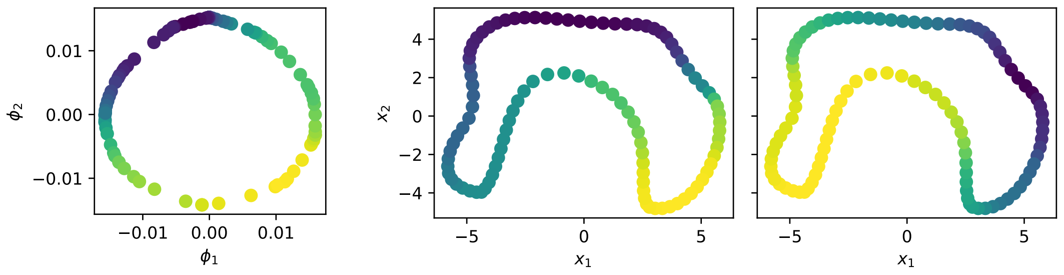


Figure 9. *Left panel: diffusion maps embedding (ϕ_1, ϕ_2) of the first 10 jointly smooth functions, colored by the x_1-coordinate of the blue car on the track. Right panel: the positions of the blue car (not part of the data set), colored by the two diffusion map coordinates ϕ_1 and ϕ_2.*

3. Delay embed each data point by concatenating the subsequent 150 points of the time series to it, forming new data points of 10×151 dimensions.
4. Flatten the data to obtain vectors of dimension $10 \times 151 = 1{,}510$.
5. Again project the data onto its principal components and use the first 5 as a new embedding.

We then apply Algorithm 4.1 to the two data sets to obtain 10 jointly smooth functions (see Figure 8). Applying diffusion maps on these 10 jointly smooth features reveals the periodic and intrinsically one-dimensional structure of the common part between both video streams (see Figure 9). The structure originates from the blue car looping around the track, which is common to both sets of observations. The jointly smooth functions (or, alternatively, the diffusion maps embedding) can be used to segment the blue car in the overhead camera images by simply averaging all collected frames that have similar embedding coordinates (see Figure 10).

6.4. Application to nonlinear dynamical systems analysis.

Constructing minimal realizations of parameter spaces for nonlinear dynamical systems from observations is a long-standing problem [22]. Here, we show that jointly smooth functions can be used to identify effective parameters and corresponding steady states, essentially constructing an effective bifurcation diagram for the dynamical system through level set parametrization. For an alternative approach to this, see [23]. Figure 11 illustrates the dynamical system we study in this example. An aircraft in flight is modeled by a nonlinear dynamical system for its altitude x_1 and speed x_2. The system dynamics is governed by three parameters p_1, p_2, and p_3 (interpreted as fuel flow, thrust, and wind speed):

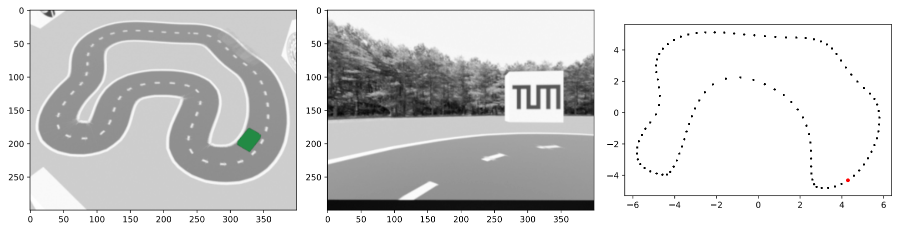


Figure 10. *Image segmentation using k-nearest-neighbor search with jointly smooth functions. The blue car is correctly identified by selecting an arbitrary point in the embedding space and averaging over its nearest neighbors for the two video streams (left and center panel). The right panel indicates where the point is on the track. The averaging procedure leads to the other car being blurred out because several of its positions match the single position of the blue car.*

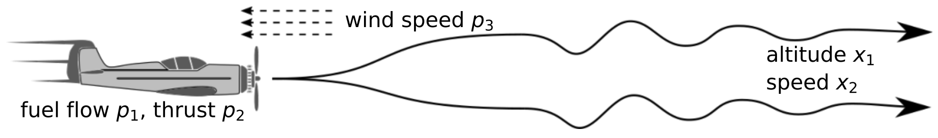


Figure 11. *Illustration for our toy model: an airplane flies at an altitude $x_1(t)$ with speed $x_2(t)$. The pilot can influence fuel flow p_1 and thrust p_2 but not the wind speed p_3.*

$$\mathbb{R}^2 \ni \dot{\boldsymbol{x}} = h_{p_1,p_2,p_3}(\boldsymbol{x}) = J(s(\boldsymbol{x})) \cdot g_{p_1,p_2,p_3}(s(\boldsymbol{x})). \tag{6.3}$$

The function h describes the behavior of the plane as it oscillates around a target altitude $x_1(t \to \infty)$ and target speed $x_2(t \to \infty)$ given by $s^{-1}(p_1 + p_2^3, p_3)$. In this example, we define h through a nonlinear transform $\boldsymbol{y} = s(\boldsymbol{x})$ of a damped, linear pendulum $\dot{\boldsymbol{y}} = g(\boldsymbol{y})$ as

$$J(\boldsymbol{y}) = \begin{bmatrix} 1 & -2y_2 \\ -2y_1 + 2y_2^2 & 1 + 4y_1y_2 - 4y_2^3 \end{bmatrix}, \quad s(\boldsymbol{x}) = \begin{bmatrix} x_1 + x_1^4 + 2x_1^2x_2 + x_2^2 \\ x_1^2 + x_2 \end{bmatrix},$$

and

$$g_{p_1,p_2,p_3}(\boldsymbol{y}) = \begin{bmatrix} -2 & 1 \\ -1 & -1 \end{bmatrix} \begin{bmatrix} y_1 - (p_1 + p_2^3) \\ y_2 - p_3 \end{bmatrix}. \tag{6.4}$$

Figure 12(a) presents the phase portraits of the system given $(p_1, p_2, p_3) = (0.1, -0.2, 0.2)$ and $(p_1, p_2, p_3) = (0.2, 0.3, -0.1)$ at the top and bottom, respectively. By (6.4), despite having three controlling parameters, the system dynamics depend only on p_3 and $p_1 + p_2^3$.

Suppose that the system dynamics, here represented by h, are unknown. Furthermore, suppose we can only control fuel flow p_1 and thrust p_2 but not the wind speed p_3. For each pair (p_1, p_2), we choose a random p_3, simulate the system, and then observe the aircraft's steady-state altitude and speed (in the limit $t \to \infty$).

The identification of an effective parameter is based on the application of Algorithm 4.1 to the (accessible) parameter space (p_1, p_2) and the collection of observed steady states (x_1, x_2). For demonstration purposes, we generate $N = 2{,}000$ points $\{(p_1, p_2, p_3)_i\}_{i=1}^N$ in the parameter

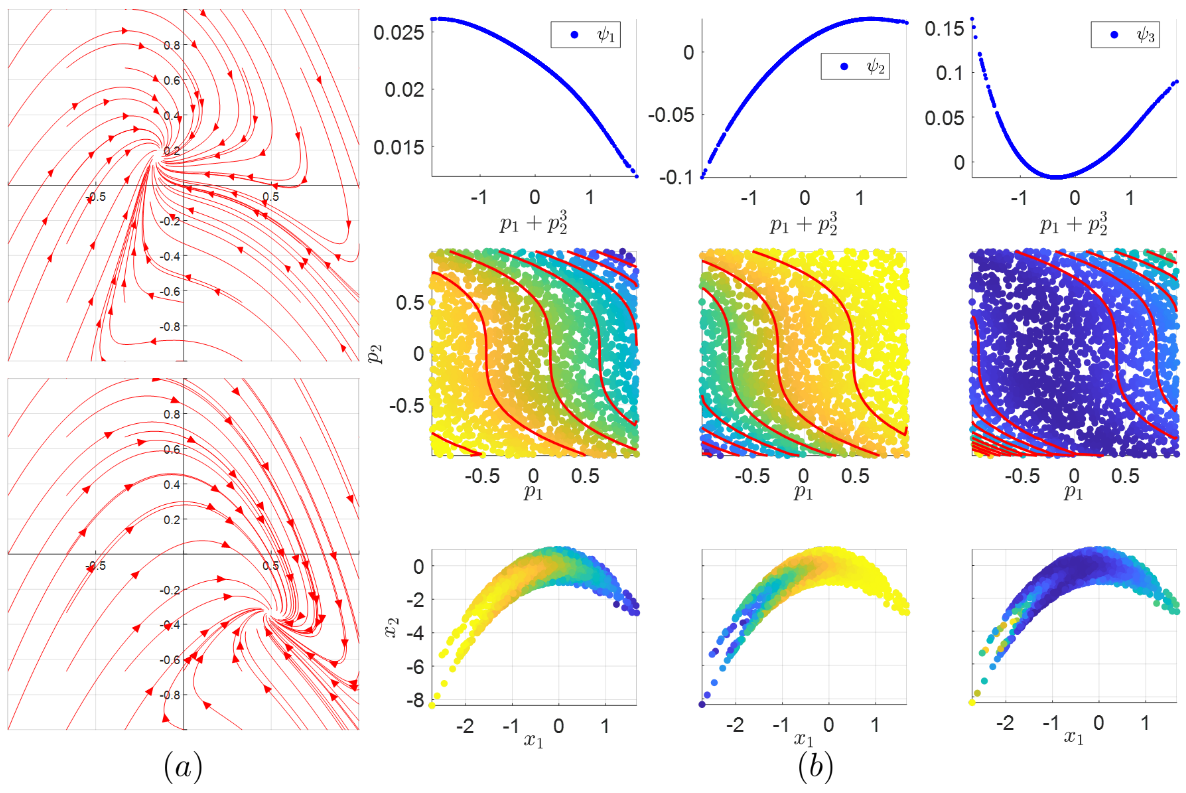


Figure 12. (a) *Phase portraits of the dynamical system with* $(p_1, p_2, p_3) = (0.1, -0.2, 0.2)$ *at the top and with* $(p_1, p_2, p_3) = (0.2, 0.3, -0.1)$ *at the bottom.* (b) *The top row displays the top 3 jointly smooth functions obtained by Algorithm 4.1 as a function of the unknown combination* $p_1 + p_3^3$*. The middle row presents the scatter plots of* p_1 *against* p_2 *colored by the top 3 jointly smooth functions, where the level sets are marked by red curves. The bottom row depicts the scatter plots of the observed steady-state coordinates* x_1 *and* x_2*, which are colored according to the obtained jointly smooth functions* $\boldsymbol{f}_m$*.*

space, uniformly distributed in $[-1, 1]^3$. For each triplet, we simulate the system (6.3) until convergence to a steady-state point $(x_1, x_2)_i$. In our analysis, we first construct two kernels: $\boldsymbol{K}_p \in \mathbb{R}^{N \times N}$ on the accessible parameters $(p_1, p_2)_i$ and $\boldsymbol{K}_x \in \mathbb{R}^{N \times N}$ on the corresponding steady states $(x_1, x_2)_i$. Then, we apply Algorithm 4.1 to the two kernels with $d = 500$, resulting in the most jointly smooth $M = 3$ functions $\boldsymbol{f}_m$. At the top row of Figure 12(b) we display scatter plots of the top three jointly smooth functions $\boldsymbol{f}_m$ as functions of the unknown combination of parameters $p_1 + p_2^3$, where we observe a distinct correspondence. To learn this combination, we plot the scatter of p_1 against p_2 and color the points according to the obtained jointly smooth functions $\boldsymbol{f}_m$ at the middle row of Figure 12(b). Indeed, we observe that the level sets (marked by red curves) coincide with $p_1 + p_2^3 = C$ (up to mild boundary effects). Similarly, at the bottom row of Figure 12(b), we depict the scatter plots of the observed steady-state coordinates x_1 and x_2, which are colored according to the obtained jointly smooth functions $\boldsymbol{f}_m$. We observe that by controlling the combination of parameters $p_1 + p_2^3$, one can shift the steady state along the color gradient, whereas the inaccessible parameter p_3 controls the steady-state location along the observed level sets. In terms of the aircraft illustration, it implies which combinations of altitude and speed can be controlled by the accessible effective parameter.

7. Conclusions. In this paper, we presented an approach for multimodal data analysis by introducing a notion of jointly smooth functions on manifolds. We proposed a new spectral algorithm for discovering such jointly smooth functions solely from observations and provided theoretical justification for it. We demonstrated the efficacy of our approach on simulated and real measured data, achieving superior results compared to competing methods. We believe the generic formulation of the problem and the algorithm facilitates applications in a broad range of fields, reaching well beyond strict data analysis, e.g., in nonlinear dynamical systems analysis, as demonstrated in the paper.

REFERENCES

[1] Y. Aflalo, H. Brezis, A. Bruckstein, R. Kimmel, and N. Sochen, *Best bases for signal spaces*, C. R. Math. Acad. Sci. Paris, 354 (2016), pp. 1155–1167.

[2] Y. Aflalo, H. Brezis, and R. Kimmel, *On the optimality of shape and data representation in the spectral domain*, SIAM J. Imaging Sci., 8 (2015), pp. 1141–1160.

[3] S. Akaho, *A Kernel Method for Canonical Correlation Analysis*, preprint, arXiv:cs/0609071, 2006.

[4] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, *Deep canonical correlation analysis*, in Proceedings of the 30th International Conference on Machine Learning, Proc. Mach. Learn. Res. (PMLR) 28, JMLR, Cambridge, MA, 2013, pp. 1247–1255.

[5] M. Belkin and P. Niyogi, *Laplacian eigenmaps and spectral techniques for embedding and clustering*, in Advances in Neural Information Processing Systems 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, eds., MIT Press, Cambridge, MA, 2001, pp. 585–591.

[6] Y. Bengio, J.-f. Paiement, P. Vincent, O. Delalleau, N. L. Roux, and M. Ouimet, *Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering*, in Advances in Neural Information Processing Systems 16, MIT Press, Cambridge, MA, 2004, pp. 177–184.

[7] T. Berry and T. Sauer, *Local kernels and the geometric structure of data*, Appl. Comput. Harmon. Anal., 40 (2015), pp. 439–469, https://doi.org/10.1016/j.acha.2015.03.002.

[8] T. Berry and T. Sauer, *Consistent manifold representation for topological data analysis*, Foundations of Data Science, 1 (2019), pp. 1–38, https://doi.org/10.3934/fods.2019001.

[9] H. Brezis and D. Gómez-Castro, *Rigidity of optimal bases for signal spaces*, C. R. Math. Acad. Sci. Paris, 355 (2017), pp. 780–785.

[10] M. Budišić, R. Mohr, and I. Mezić, *Applied Koopmanism*, Chaos, 22 (2012), 047510.

[11] Y.-C. Chen and M. Meilă, *Selecting the Independent Coordinates of Manifolds With Large Aspect Ratios*, preprint, arXiv:1907.01651v1, 2019, https://arxiv.org/abs/1907.01651v1.

[12] N. C. Chung and J. D. Storey, *Statistical significance of variables driving systematic variation in high-dimensional data*, Bioinformatics, 31 (2015), pp. 545–554.

[13] R. R. Coifman and S. Lafon, *Diffusion maps*, Appl. Comput. Harmon. Anal., 21 (2006), pp. 5–30.

[14] R. R. Coifman and S. Lafon, *Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions*, Appl. Comput. Harmon. Anal., 21 (2006), pp. 31–52.

[15] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput., 9 (1990), pp. 251–280, https://doi.org/10.1016/s0747-7171(08)80013-2.

[16] F. Dietrich, G. Köster, and H.-J. Bungartz, *Numerical model construction with closed observables*, SIAM J. Appl. Dyn. Syst., 15 (2016), pp. 2078–2108.

[17] C. J. Dsilva, R. Talmon, R. R. Coifman, and I. G. Kevrekidis, *Parsimonious representation of nonlinear dynamical systems through manifold learning: A chemotaxis case study*, Appl. Comput. Harmon. Anal., 44 (2018), pp. 759–773, https://doi.org/10.1016/j.acha.2015.06.008.

[18] D. Eynard, A. Kovnatsky, M. M. Bronstein, K. Glashoff, and A. M. Bronstein, *Multimodal manifold analysis by simultaneous diagonalization of Laplacians*, IEEE Trans. Pattern Anal. Mach. Intell., 37 (2015), pp. 2505–2517.

[19] C. Fowlkes, S. Belongie, and J. Malik, *Efficient spatiotemporal grouping using the nystrom method*, in Proceedings of the IEEE Computer Society Conference on Computer vision and Pattern Recognition, vol. 1, IEEE, Piscataway, NJ, 2001.

[20] D. Giannakis, *Data-driven spectral decomposition and forecasting of ergodic dynamical systems*, Appl. Comput. Harmon. Anal., 47 (2017) pp. 338–396.

[21] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, *Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals*, Circulation, 101 (2000), pp. e215–e220.

[22] R. N. Gutenkunst, J. J. Waterfall, F. P. Casey, K. S. Brown, C. R. Myers, and J. P. Sethna, *Universally sloppy parameter sensitivities in systems biology models*, PLoS Comput. Biol., 3 (2007), p. e189.

[23] A. Holiday, M. Kooshkbaghi, J. M. Bello-Rivas, C. W. Gear, A. Zagaris, and I. G. Kevrekidis, *Manifold learning for parameter reduction*, J. Comput. Phys., 392 (2019), pp. 419–431.

[24] H. Hotelling, *Relations between two sets of variates*, Biometrika, 28 (1936), pp. 321–377.

[25] I. M. Johnstone, *Multivariate analysis and Jacobi ensembles: Largest eigenvalue, Tracy–Widom limits and rates of convergence*, Ann. Stat., 36 (2008), p. 2638.

[26] B. O. Koopman, *Hamiltonian systems and transformation in Hilbert space*, Proc. Natl. Acad. Sci. USA, 17 (1931), pp. 315–318.

[27] P. L. Lai and C. Fyfe, *Kernel and nonlinear canonical correlation analysis*, Int. J. Neural Syst., 10 (2000), pp. 365–377.

[28] R. R. Lederman and R. Talmon, *Learning the geometry of common latent variables using alternating-diffusion*, Appl. Comput. Harmon. Anal., 44 (2018), pp. 509–536.

[29] O. Lindenbaum, A. Yeredor, M. Salhov, and A. Averbuch, *Multi-view diffusion maps*, Informat. Fusion, 55 (2020), pp. 127–149, https://doi.org/10.1016/j.inffus.2019.08.005.

[30] J. McQueen, M. Meilă, J. VanderPlas, and Z. Zhang, *Megaman: Scalable manifold learning in python*, J. Mach. Learn. Res., 17 (2016), pp. 1–5, http://jmlr.org/papers/v17/16-109.html.

[31] J. Mercer, *Functions of positive and negative type, and their connection with the theory of integral equations*, Philos. Trans. R. Soc. A, 209 (1909), pp. 415–446.

[32] T. Michaeli, W. Wang, and K. Livescu, *Nonparametric canonical correlation analysis*, in Proceedings of the 33rd International Conference on Machine Learning, Proc. Mach. Learn. Res. (PMLR) 48, JMLR, Cambridge, MA, 2016, pp. 1967–1976.

[33] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, *Functional maps: A flexible representation of maps between shapes*, A CM Trans. Graphics, 31 (2012), pp. 1–11.

[34] D. Perrault-Joncas and M. Meilă, *Improved Graph Laplacian via Geometric Self-Consistency*, preprint, arXiv:1406.0118, 2014.

[35] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation And Machine Learning)*, MIT Press, Cambridge, MA, 2005.

[36] S. Rosenberg, *The Laplacian on a Riemannian Manifold*, Cambridge University Press, Cambridge, UK, 1997.

[37] B. Schölkopf, A. Smola, and K.-R. Müller, *Kernel principal component analysis*, Artificial Neural Networks – ICANN' 97, Springer, Berlin, 1997, pp. 583–588.

[38] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK, 2004.

[39] C. Shen and H.-T. Wu, *Scalability and Robustness of Spectral Embedding: Landmark Diffusion Is All You Need*, preprint, arXiv:2001.00801, 2020, https://arxiv.org/abs/2001.00801.

[40] A. Singer and H. tieng Wu, *Orientability and diffusion maps*, Appl. Comput. Harmon. Anal., 31 (2011), pp. 44–58, https://doi.org/10.1016/j.acha.2010.10.001.

[41] D. C. Sorensen, *Implicitly Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations*, Springer, Dordrecht, the Netherlands, 1997, pp. 119–165, https://doi.org/10.1007/978-94-011-5412-3_5.

[42] M. G. Terzano, L. Parrino, A. Smerieri, R. Chervin, S. Chokroverty, C. Guilleminault, M. Hirshkowitz, M. Mahowald, H. Moldofsky, A. Rosa, R. Thomas, and A. Walters, *Atlas, rules, and recording techniques for the scoring of cyclic alternating pattern (CAP) in human sleep*, Sleep Med., 3 (2002), pp. 187–199.

[43] H. Whitney, *Differentiable manifolds*, Ann. Math. (2), 37 (1936), pp. 645–680, https://doi.org/10.2307/1968482.

[44] C. K. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, in Advances in Neural Information Processing Systems B, MIT Press, Cambridge, MA, 2001, pp. 682–688.

[45] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, *A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition*, J. Nonlinear Sci. Appl., 25 (2015), pp. 1307–1346.

[46] O. Yair and R. Talmon, *Local canonical correlation analysis for nonlinear common variables discovery*, IEEE Trans. Signal Process., 65 (2017), pp. 1101–1115, https://doi.org/10.1109/tsp.2016.2628348.

[47] K. Yosida, *Functional Analysis*, Springer, Berlin, 1995.

[48] S. Yu, L.-C. Tranchevent, B. De Moor, and Y. Moreau, *Kernel-based Data Fusion for Machine Learning*, Springer, Berlin, 2013.