Mach Learn (2016) 103:81–102
DOI 10.1007/s10994-015-5538-4

Learning from patches by efficient spectral decomposition of a structured kernel

Moshe Salhov[1,2] · **Amit Bermanis**[1] · **Guy Wolf**[1,2] · **Amir Averbuch**[1,2]

Received: 28 August 2014 / Accepted: 5 October 2015 / Published online: 17 December 2015

Abstract We present a kernel based method that learns from a small neighborhoods (patches) of multidimensional data points. This method is based on spectral decomposition of a large structured kernel accompanied by an out-of-sample extension method. In many cases, the performance of a spectral based learning mechanism is limited due to the use of a distance metric among the multidimensional data points in the kernel construction. Recently, different distance metrics have been proposed that are based on a spectral decomposition of an appropriate kernel prior to the application of learning mechanisms. The diffusion distance metric is a typical example where a distance metric is computed by incorporating the relation of a single measurement to the entire input dataset. A method, which is called patch-to-tensor embedding (PTE), generalizes the diffusion distance metric that incorporates matrix similarity relations into the kernel construction that replaces its scalar entries with matrices. The use of multidimensional similarities in PTE based spectral decomposition results in a bigger kernel that significantly increases its computational complexity. In this paper, we propose an efficient dictionary construction that approximates the oversized PTE kernel and its associated spectral decomposition. It is supplemented by providing an out-of-sample extension for vector fields. Furthermore, the approximation error is analyzed and the advantages of the proposed dictionary construction are demonstrated on several image processing tasks.

Editor: Paolo Frasconi.

✉ Moshe Salhov
moshebar-s@013.net; moshesa@post.tau.ac.il

Amit Bermanis
abermanis@gmail.com

Guy Wolf
guy.wolf@cs.tau.ac.il

Amir Averbuch
amir1@post.tau.ac.il

1 School of Computer Science, Tel Aviv University, 69978 Tel Aviv, Israel

2 Department of Mathematical Information Technology, University of Jyväskylä, Jyväskylä, Finland

Keywords Kernel method · Out of sample extension · Non-scalar similarity · Diffusion maps · Vector field interpolation · Manifold learning · High-dimensional data analysis

1 Introduction

Recent machine learning methods for massive high dimensional data analysis model observable parameters in such datasets by the application of nonlinear mappings of a small number of underlying factors (Belkin and Niyogi 2003; Singer and Coifman 2008). Mathematically, these mappings are characterized by the geometric structure of a low dimensional manifold that is immersed in a high dimensional ambient space. Under this model, the dataset is assumed to be sampled from an unknown manifold in the ambient observable space. Then, manifold learning methods are applied to reveal the underlying geometry. One approach for revealing elements of this geometry, such as dimensionally estimated and tangent space inference, is by encoding tensor similarities directly in the the ambient space, as proposed in Mordohai and Medioni (2010). Another approach, which also involves dimensionality reduction, is the to use kernel methods such as k-PCA, Laplacian Eigenmaps (Belkin and Niyogi 2003) and Diffusion Maps (DM) (Coifman and Lafon 2006), which are designed for this task and utilize the intrinsic manifold geometry. These methods have provided good results in representing and analyzing such data. Kernel methods are based on a scalar similarity measure between individual multidimensional data points.[1]

In this paper, we focus on patch-to-tensor embedding (PTE) kernel construction from Salhov et al. (2012), which considers patches that are taken from the underlying manifold. The data analysis considers these patches instead of analyzing single data points from the manifold. Each patch is defined as a local neighborhood of a data point in a dataset sampled from an underlying manifold. The relation between any two patches in a PTE based kernel construction is described by a matrix that represents both the affinities between data points at the centers of these patches and the similarities between their local coordinate systems. Then, the constructed matrices between all patches are combined into a block matrix that is called super-kernel. This super-kernel is a non-scalar affinity kernel between patches (local neighborhood) of the underlaying manifold (Salhov et al. 2012). The super-kernel is positive definite. It captures the intrinsic geometry of the underlying manifold/s. However, the proposed enhanced kernel increases its size. Hence, it increases the computational complexity of a direct spectral decomposition of this kernel.

In this paper, we generalize the dictionary construction approach in Engel et al. (2004) to approximate the spectral decomposition of a non-scalar PTE based kernel that utilizes the underlying patch structure inside the ambient space. The contributions of this paper are twofold:

1. We utilize the structure of the super-kernel from a PTE construction to find a necessary condition for updating a non-scalar based dictionary. This dictionary is used to approximate the spectral decomposition of the super-kernel and to derive the required conditions for achieving a bound on the super-kernel approximation error. In addition, we analyze its computational complexity and estimate its efficiency in comparison to the computation of a spectral decomposition of a full super-kernel.
2. We show how to efficiently extend the PTE embedding via an out-of-sample extension to a newly arrived data point that did not participate in the dictionary construction.

[1] All the data points in this paper are multidimensional. They will be denoted as data points.

This includes the case when the analyzed dataset consists of tangential vector fields and provides an out-of-sample extension method to process vector fields.

The PTE based kernel utilizes the DM methodology (Coifman and Lafon 2006) that is described next. The DM kernel, which is based on a scalar similarity measure between individual data points, captures their connectivity. The kernel is represented by a graph where each data point corresponds to a node and the weight of an edge between any pair of nodes reflects the similarity between the corresponding data points. The analysis of the eigenvalues and the corresponding eigenvectors of the kernel matrix reveal many properties and connections in the graph. More specifically, the diffusion distance metric (Eq. 1) represents the transition probabilities of a Markov chain that advances in time (Coifman and Lafon 2006). Unlike the geodesic distance or the shortest path in a graph, the diffusion distance is robust to noise. The diffusion distance has proved to be useful in clustering (David and Averbuch 2012), anomaly detection (David 2009), parametrization of linear systems (Talmon et al. 2012), shape recognition (Bronstein and Bronstein 2011) to name some.

Formally, the original DM is used to analyze a dataset M by exploring the geometry of the manifold $\mathcal{M}$ from which it is sampled. This method is based on defining an isotropic kernel $K \in \mathbb{R}^{n\times n}$ whose elements are defined by

$$k(x,y) \triangleq e^{-\frac{\|x-y\|^2}{\varepsilon}}, \quad x,y \in M, \tag{1}$$

where ε is a meta-parameter of the algorithm that describes the kernel's computational neighborhood. This kernel represents the affinities between data points in the manifold. The kernel is viewed as a construction of a weighted graph over the dataset M. Data points in M are the vertices and the weights (for example, we can use the weight in Eq. 1) of the edges are defined by the kernel K. The degree of each data point (i.e., vertex) $x \in M$ in this graph is $q(x) \triangleq \sum_{y\in M} k(x,y)$. Normalization of the kernel by this degree produces an $n \times n$ row stochastic transition matrix P whose elements are $p(x,y) = k(x,y)/q(x)$, $x, y \in M$, which defines a Markov process (i.e., a diffusion process) over the data points in M. A symmetric conjugate $\bar{P}$ of the transition operator P defines the diffusion affinities between data points by

$$\bar{p}(x,y) = \frac{k(x,y)}{\sqrt{q(x)q(y)}} = \sqrt{q(x)}\,p(x,y)\frac{1}{\sqrt{q(y)}} \qquad x,y \in M. \tag{2}$$

The DM method embeds a manifold into an Euclidean space whose dimensionality may be significantly lower than the original dimensionality. This embedding is a result from the spectral analysis of the diffusion affinity kernel $\bar{P}$. The eigenvalues $1 = \sigma_0 \geq \sigma_1 \geq \cdots$ of $\bar{P}$ and their corresponding eigenvectors $\bar{\phi}_0, \bar{\phi}_1, \ldots$ are used to construct the desired map, which embeds each data point $x \in M$ into the data point $\bar{\Phi}(x) = (\sigma_i^t \bar{\phi}_i(x))_{i=0}^{\delta}$ for a sufficiently small δ, which is the dimension of the embedded space. The exact value of δ depends on the decay of the spectrum of $\bar{P}$.

DM extends the classical Multi-Dimensional Scaling (MDS) core method (Cox and Cox 1994; Kruskal 1964) by considering nonlinear relations between data points instead of considering the original linear Gram matrix relations. Furthermore, DM has been utilized for a wide variety data types and for pattern analysis such as improving audio quality by suppressing transient interference (Talmon et al. 2013), detecting moving vehicles (Schclar et al. 2010), scene classification (Jingen et al. 2009), gene expression analysis (Rui et al. 2007), source localization (Talmon et al. 2011) and fusion of different data sources (Lafon et al. 2006; Keller et al. 2010).

DM methodology is also used in finding both a parametrization and an explicit metric, which reflects the intrinsic geometry of a given dataset. This is done by considering the entire pairwise connectivity between data points in a diffusion process (Lafon et al. 2006).

Recently, DM has been extended in several directions to handle orientation in local tangent spaces (Singer and Wu 2011, 2012; Salhov et al. 2012; Wolf and Averbuch 2013). Under the manifold assumption, the relation between two local patches of neighboring data points is described by a matrix instead of a scalar value. The resulting kernel captures enriched similarities between local structures in the underlying manifold. These enriched similarities are used to analyze local areas around data points instead of analyzing their specific individual locations. For example, this approach is beneficial in image processing for analyzing regions instead of individual pixels or when data points are perturbed, thus, their surrounding area is more important than their specific individual positions. Since the constructions of these similarities are based on local tangent spaces, they provide methods to manipulate tangential vector fields. For example, they allow to perform an out-of-sample extension for vector fields. These manipulations are beneficial when the analyzed data consists of directional information in addition to positional information on the manifold. For example, the goal in Ballester et al. (2001) is to recover missing data in images while interpolating the appropriate vector field. Another example is in the utilization of tangential vector fields interpolation on $\mathcal{S}^2$ for modeling atmospheric air flow and oceanic water velocity (Fuselier and Wright 2009).

The main motivation of this paper is to show the beneficial aspects in using PTE based construction while reducing its computational complexity when spectral decomposition of a PTE based kernel is used.

The spectral decomposition computational complexity can be reduced in several ways. Kernel matrix sparsification is a widely used approach that utilizes a sparse eigensolver such as Lanczos to compute the relevant eigenvectors (Cullum and Willoughby 2002). Another sparsification approach is to translate the dense similarity matrix into a sparse matrix by selectively truncating elements outside a given neighborhood radius of each data point. More sparsification approaches are given in von Luxburg (2007). An alternative approach to reduce the computational complexity is based on Nyström extension for estimating the required eigenvectors (Fowlkes et al. 2004). The Nyström based method has three steps: (1) The given dataset is subsampled uniformly over its indices set. (2) The subsamples define a kernel that is smaller than the dataset size. Then, the corresponding spectral decomposition is achieved by the application of a SVD solver to this kernel. (3) The solution of the small spectral problem is extended to the entire dataset via the application of Nyström extension method. The result is an approximated spectral method. Although this solution is much more computationally efficient than the application of the spectral decomposition to the original kernel matrix, the resulting approximated spectral method suffers from several major problems originated from the subsampling effect on the spectral approximation and from the ill-conditioned effects in the Nyström extension method.

Recently, a novel multiscale scheme is suggested in Bermanis et al. (2013) for sampling scattered data and for extending functions defined on sampled data points. The suggested approach, also called Multiscale Extension (MSE), overcomes some limitations of the Nyström method. For example, it overcomes the ill-conditioned effect in the Nyström extension method and accelerates the computation. The MSE method is based on mutual distances between data points. It uses a coarse-to-fine hierarchy of a multiscale decomposition of a Gaussian kernel. Another interesting approach is suggested in Engel et al. (2004), Ouimet and Bengio (2005) to identify a subset of data points (called dictionary) and a corresponding extension coefficients that enable us to approximate the full scalar kernel. The number of dictionary atoms depends on the given data, kernel configuration and on the designed para-

meter that controls the quality of the full kernel approximation. These are efficient methods designed to process general kernels. However, they are not designed to utilize and preserve the inherent structure that resides in kernels such as a super-kernel (Salhov et al. 2012).

In summary, methods such as Nyström and kernel recursive least squares, to name some, have been suggested to approximate large kernels decomposition. However, neither the suggested methods utilize the inherent structure of the PTE kernel nor suggest an out-of-sample extension method for vector fields. In this paper, we propose a dictionary construction that approximates the oversized PTE kernel and its associated spectral decomposition. Furthermore, the approximation error is analyzed and the advantages of the proposed dictionary construction are demonstrated on a several image processing tasks.

The paper has the following structure: Sect. 2 formulates the problem. The patch-to-tensor embedding is described in Sect. 3. The patch-based dictionary construction and its properties are described in Sect. 4. Finally, Sect. 5 describes the experimental results for image segmentation and for handwritten digit classification that are based on the utilization of a dictionary-based analysis.

2 Problem formulation

In this paper, we approximate the spectral decomposition of a large and structured kernel. We assume that we have a dataset of n multidimensional data points that are sampled from a manifold $\mathcal{M}$ that lies in the ambient space $\mathbb{R}^m$. We also assume that the intrinsic dimension of $\mathcal{M}$ is $d \ll m$. We consider two related tasks: (1) How to approximate the spectral decomposition of a super-kernel? This is explained in Sect. 2.2. (2) How to perform an out-of-sample extension for vector fields? This is explained in Sect. 2.3.

2.1 Manifold setup

At every multidimensional data point $x \in M$, the manifold has a d-dimensional tangent space $T_x(\mathcal{M})$, which is a subspace of $\mathbb{R}^m$. We assume that the manifold is densely sampled thus, the tangential space $T_x(\mathcal{M})$ can be approximated by a small enough patch (i.e., neighborhood) $N(x) \subseteq M$ around $x \in M$. Let $o_x^1, \ldots, o_x^d \in \mathbb{R}^m$, where $o_x^i = (o_x^{i,1}, \ldots, o_x^{i,m})^T$, $i = 1, \ldots, d$, form an orthonormal basis of $T_x(\mathcal{M})$ and let $O_x \in \mathbb{R}^{m \times d}$ be a matrix whose columns are these vectors

$$O_x \triangleq \begin{pmatrix} | & & | \\ o_x^1 & \cdots & o_x^d \\ | & & | \end{pmatrix} \qquad x \in M. \tag{3}$$

The matrix O_x can be analyzed spectrally by applying to it the local PCA method (Singer and Wu 2011). We do it differently. From now on we assume that the vectors in $T_x(\mathcal{M})$ are expressed by their d coordinates according to the basis $o_x^1, \ldots, o_x^d$. For each vector $u \in T_x(\mathcal{M})$, $\tilde{u} = O_x u \in \mathbb{R}^m$ is the same vector as u represented by m coordinates, according to the basis of the ambient space. For each vector $v \in \mathbb{R}^m$ in the ambient space, the vector $v' = O_x^T v \in T_x(\mathcal{M})$ is a linear projection of v on the tangent space $T_x(\mathcal{M})$. This setting was used in Salhov et al. (2012). In Sect. 2.2, we formulate the spectral decomposition of a super kernel.

2.2 Approximation of the super-kernel spectral decomposition

The super-kernel G, which is constructed in Salhov et al. (2012), on a dataset M, is based on d orthonormal basis vectors in each O_x, $x \in M$. G is a $nd \times nd$ structured matrix. Each

$d \times d$ block in G is a similarity-index matrix between a pair of data points and their corresponding neighborhoods. More details of the structure of G are given in Sect. 3 (specifically, in Definition 1). Given a dataset M, the PTE method (Salhov et al. 2012) uses the spectral decomposition of G to embed each data point (or neighborhood around it) to a tensor. This embedding is computed by constructing the coordinate matrices of the embedded tensors using the eigenvalues and the eigenvectors of G.

We want to efficiently approximate the PTE embedded tensors without computing directly the spectral decomposition of G. Under the manifold assumption, the super-kernel G has a small number of significant singular values in comparison with the dimensionality m of the given data. Hence, a low rank approximation of the super-kernel G exists. We aim to find a dictionary D_{η_n} of η_n representative data points that will be used to construct a small super-kernel $\hat{G}_n$ and a corresponding extension matrix $E_n \in \mathbb{R}^{\eta_n d \times nd}$ that approximates the entire super-kernel G by

$$G \approx E_n^T \hat{G}_n E_n. \tag{4}$$

The approximation of the spectral decomposition of G is computed by the application of QR decomposition (see Golub and Loan 2012, p. 246) to E_n. In order to construct a dictionary of representatives, we design a criterion that will identify data points and their corresponding embedded tensors that are approximately linearly independent. This criterion is utilized to control the approximation error bound.

2.3 Out-of-sample extension method for vector fields

Consider a tangential vector field $\mathbf{v} : M \to \mathbb{R}^d$ such that for all $x \in M$, $\mathbf{v}(x) \in T_x(\mathcal{M})$. The goal is to find an out-of-sample extension method for such tangential vector fields. More specifically, given a new data point on the manifold, we aim to find the vector in its local tangential space that corresponds smoothly to the known values of the vector field. This out-of-sample extension method is used to extend the coordinate matrices of the embedded tensors of PTE and to approximate the embedding of a new data point $x \notin M$.

A tangential vector field can be technically described in two forms. The standard intuitive form describes it as a set of separate vectors or, equivalently, as a $n \times d$ matrix whose rows are these vectors. An alternative form is to concatenate its tangential vectors to represent the vector field by vector of length nd. Figure 1 illustrates these two forms.

Let $x' \in \mathcal{M} \backslash M$ be a new data point associated with the matrix $O_{x'}$ whose columns $o^1_{x'}, \ldots, o^d_{x'}$ form an orthonormal basis for the tangential space $T_{x'}(\mathcal{M})$. The out-of-sample extension problem is formulated as finding an extension vector field $\mathbf{u}$ that enables us to compute a vector $\mathbf{v}(x')$ for any new data point x'.

The super-kernel G defines an integral operator on tangential vector fields of the manifold $\mathcal{M}$ that are restricted to the dataset M (see Wolf and Averbuch 2013, Definition 4.1). This integral operator can be formally described by the application of a super-kernel matrix to a long-vector (second representation form) that represents vector fields. By abuse of notation, if we treat $\mathbf{u}$ as its vector representation in $\mathbb{R}^{nd}$, we define the relation between $\mathbf{u}$ and $\mathbf{v}(x')$ using the operator G to be

$$\mathbf{v}(x') \triangleq \sum_{y \in M} \tilde{G}_{(x',y)} \mathbf{u}(y), \tag{5}$$

where $\tilde{G}_{(x',y)}$ are the non-scalar similarity blocks between the new data point $x' \notin M$ and the data points $y \in M$. We will use Eq. 5 to provide an interpolation scheme for finding the

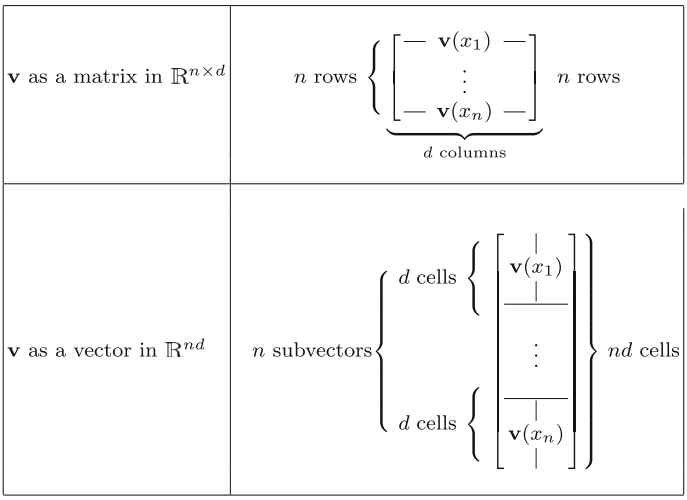

Fig. 1 $x_1, \ldots, x_n$ denote all the data points in M. *Top* a viewing illustration of a vector field $\mathbf{v} : M \to \mathbb{R}^d$ as a $n \times d$ matrix. *Bottom* a vector of length nd

described vector field $\mathbf{u}$ and for using it for the out-of-sample extension of tangential vector fields in general and for the PTE based embedded tensors in particular. The latter is achieved by extending the eigenvectors of G, which are vectors in $\mathbb{R}^{nd}$ that are regarded as tangential vector fields.

3 Patch-to-tensor embedding

The main goal of this paper is to enable to have a practical application of the PTE method from Salhov et al. (2012), Wolf and Averbuch (2013). In these works, the PTE methodology was introduced either in continuous settings or with a dense sampling of the manifold, which approximates the continuous settings. While these settings may not be suitable for efficient application to process modern datasets (indeed, this is the main focus of this paper), they enabled a thorough analysis regarding the geometrical interpretation of the obtained embedding and its metric structure, in relation to the manifold geometry. In order to make the presentation of this paper self-contained, we review in this section the main results from these previous works before introducing the proposed dictionary construction and out-of-sample extension method.

For $x, y \in M$, define $O_{xy} \stackrel{\Delta}{=} O_x^T O_y \in \mathbb{R}^{d\times d}$, where O_x and O_y were defined in Eq. 3. The matrices O_x and O_y represent the bases for the tangential spaces $T_x(\mathcal{M})$ and $T_y(\mathcal{M})$, respectively. Thus, the matrix O_{xy}, which will be referred to as the tangential similarity matrix, represents a linear-projection between these tangential spaces. According to Salhov et al. (2012), Wolf and Averbuch (2013), this linear-projection quantifies the similarity between these tangential spaces via the angle between their relative orientations.

Let Ω be an $n \times n$ symmetric and positive semi-definite affinity kernel defined on $M \subseteq \mathbb{R}^m$. Each row or each column in Ω corresponds to a data point in M, and each element $[\Omega]_{xy} = \omega(x, y)$, $x, y \in M$, represents the affinity between x and y such that $\omega(x, y) \geq 0$ for every $x, y \in M$. The diffusion affinity kernel is an example for such an affinity kernel.

Definition 1 uses the tangent similarity matrices and the affinity kernel Ω to define the Linear-Projection *super-kernel*. When the diffusion affinities in $\bar{P}$ (Eq. 2) are used instead of using the general affinities in Ω, then this structured kernel is called a Linear-Projection Diffusion (LPD) super-kernel.

Definition 1 (Salhov et al. 2012) [Linear-Projection Diffusion Super-Kernel] A Linear-Projection Diffusion (LPD) super-kernel is a block matrix $G \in \mathbb{R}^{nd\times nd}$ of size $n \times n$ where each block in it is a $d \times d$ matrix. Each row and each column of blocks in G correspond to a data point in M. A single block $G_{(x,y)}$, $x, y \in M$, represents an affinity (similarity) between the patches $N(x)$ and $N(y)$ around x and y, respectively. Each block $G_{(x,y)} \in \mathbb{R}^{d\times d}$ in G is defined by $G_{(x,y)} \triangleq \bar{p}(x, y)O_{xy} = a(x, y)O_x^T O_y,\ x, y \in M$.

The super-kernel in Definition 1 encompasses both the diffusion affinities between data points from the manifold $\mathcal{M}$ and the similarities between their tangential spaces. The latter are expressed by the linear-projection operators between tangential spaces. Specifically, for two tangential spaces $T_x(\mathcal{M})$ and $T_y(\mathcal{M})$ at $x, y \in M$ of the manifold, the operator $O_x^T O_y$ (i.e., their tangential similarity matrix) expresses a linear projection from $T_y(\mathcal{M})$ to $T_x(\mathcal{M})$ via the ambient space $\mathbb{R}^m$. The obvious extreme cases are the identity matrix, which indicates the existence of a complete similarity, and a zero matrix, which indicates the existence of orthogonality (i.e., a complete dissimilarity). These linear projection operators express some important properties of the manifold structure, e.g., curvatures between patches and differences in orientation. More details on the properties of this super-kernel are given in Salhov et al. (2012), Wolf and Averbuch (2013).

It is convenient to use the vectors o_x^i and o_y^j to apply a double-indexing scheme by using the notation $g(o_x^i, o_y^j) \triangleq [G_{(x,y)}]_{ij}$ that considers each single cell in G as an element $[G_{(x,y)}]_{ij}$, $1 \le i, j \le d$, in the block $G_{(x,y)}$, $x, y \in M$. It is important to note that $g(o_x^i, o_y^j)$ is *only a convenient notation* and a single element of a block in G does not necessarily have any special meaning. The block itself, as a whole, holds meaningful similarity information.

Spectral decomposition is used to analyze the super-kernel G. It is utilized to embed the patches $N(x)$, $x \in M$, into a tensor space. Let $|\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_\ell|$ be the ℓ most significant eigenvalues of G and let $\phi_1, \phi_2, \ldots, \phi_\ell$ be their corresponding eigenvectors. Each eigenvector ϕ_i, $i = 1, \ldots, \ell$, is a vector of length nd. We denote each of its elements by $\phi_i(o_x^j)$, $x \in M$, $j = 1, \ldots, d$. An eigenvector ϕ_i is a vector of n blocks, each of which is a vector of length d that corresponds to a data point $x \in M$ on the manifold. To express this notion, we use the notation $\varphi_i^j(x) = \phi_i(o_x^j)$. Thus, the block, which corresponds to $x \in M$ in ϕ_i, is the vector $(\varphi_i^1(x), \ldots, \varphi_i^d(x))^T$.

The eigenvalues and the eigenvectors of G are used to construct a spectral map

$$\Phi(o_x^j) = [\lambda_1^t \phi_1(o_x^j), \ldots, \lambda_\ell^t \phi_\ell(o_x^j)]^T, \tag{6}$$

which is similar to the one used in DM and t is the diffusion transition time. This spectral map is then used to construct the embedded tensor $\mathcal{T}_x \in \mathbb{R}^\ell \otimes \mathbb{R}^d$ for each $x \in M$. These tensors are represented by the $\ell \times d$ matrices

$$\mathcal{T}_x \triangleq \begin{pmatrix} | & & | \\ \Phi(o_x^1) & \cdots & \Phi(o_x^d) \\ | & & | \end{pmatrix} \qquad x \in M. \tag{7}$$

In other words, the coordinates of $\mathcal{T}_x$ (i.e., the elements in this matrix) are $[\mathcal{T}_x]_{ij} = \lambda_i^t \varphi_i^j(x)$, $i = 1, \ldots, \ell$, $j = 1, \ldots, d$. Each tensor $\mathcal{T}_x$, which is an $l \times d$ matrix, represents an embedding of the patch $N(x)$, $x \in M$, into the tensor space.

The computational complexity of the direct spectral decomposition of a super-kernel is in the order of $O\left(n^3d^3\right)$. However, the structure of the super-kernel can be utilized for simplifying the decomposition. The super kernel can be viewed as the Khatri–Rao product (Rao 1968) of the diffusion affinity matrix Ω with the $nd \times nd$ matrix $B = OO^T$, where the matrix $O \in \mathbb{R}^{nd\times d}$ is a block matrix with O_i^T as its $d \times d$ ith sub-block Hence, let $G = \Omega \odot B$ be the corresponding Khatri–Rao product where the blocks of Ω are the affinity scalars and the blocks of B are the $d \times d$ respected matrices $O_i^T O_j$. It can be shown that the symmetric partition of B can be used to reformulate the Khatri–Rao product in terms of a Kronecker product as $G = Z\left(\Omega * B\right) Z^T$, where Z is a proper $nd \times n^2d^2$ selection matrix (Liu and Trenkler 2008), $Z^T Z = I$ and $\Omega * B$ is the Kronecker product of Ω and B. Let $\Omega = U_\Omega \Sigma_\Omega V_\Omega^T$ be the SVD of Ω and $B = U_B \Sigma_B V_B^T$ be the SVD of B, then the spectral decomposition of G can be reformulated using the spectral decomposition characterization of a Kronecker products as $G = U\Sigma V = Z\left(U_\Omega * U_B\right)\left(\Sigma_\Omega * \Sigma_B\right)\left(V_\Omega * V_B\right) Z^T$. Hence, the computation of the left eigenvectors $U = Z\left(U_\Omega * U_B\right)$ requires $O\left(n^3\right)$ operations to decompose Ω and additional operations that are needed for the decomposition of O. The total computational complexity is more efficient than what is needed to decompose $nd \times nd$ super-kernel. However, computational cost of $O\left(n^3\right)$ to process large datasets is still impractical. Hence, we are looking for a reduction of at least one order of magnitude in the spectral decomposition of large matrices that represent the super kernel.

The previous works, which is described in this section, mainly focused on geometric interpretations of the embedded space obtained by PTE and its metric structure. In this paper, on the other hand, we focus on the practical application of this analysis, which requires a dictionary construction and an out-of-sample extension method, as described in Sect. 2. To this end, we consider the numerical and algebraic sides of the PTE super-kernel and the resulting embedded space to augment its extensively studied geometric side. In particular, the dictionary construction introduced in Sect. 4 aims to reduce redundancies by considering (and eliminating) approximate linear dependencies between dictionary members, while ensuring that the span of the their embedded tensors approximates the embedded space of PTE. By ensuring that the metric structure of the embedded space is approximated by the presented construction, the geometrically-oriented results from Salhov et al. (2012), Wolf and Averbuch (2013) remain valid for this analysis as well. They provide meaningful relations between the approximate embedding and the manifold geometry.

4 Patch-based dictionary construction

According to Lemma 3.3 in Salhov et al. (2012), the sum in Eq. 5 can be rephrased in terms of the embedded tensors $x \in M$ to be

$$\mathbf{v}(x) = \sum_{y\in M} \mathcal{T}_x^T \mathcal{T}_y \mathbf{u}(y). \tag{8}$$

However, due to linear dependencies between the embedded tensors, this sum may contain redundant terms. Indeed, if a tensor $\mathcal{T}_z$ can be estimated as a weighted sum of other tensors such that $\mathcal{T}_z = \sum_{z\neq y\in M} C_y^z \mathcal{T}_y$, where $C_y^z \in \mathbb{R}$, $z \neq y \in M$, then there are scalar weights that relates a tensor $\mathcal{T}_z$ to any other tensor $\mathcal{T}_y$. For $x \in M$, Eq. 8 becomes

$$\mathbf{v}(x) = \sum_{z\neq y\in M} \mathcal{T}_x^T \mathcal{T}_y(\mathbf{u}(y) + C_y^z \mathbf{u}(z)). \tag{9}$$

This enables us to eliminate the redundant tensors. By applying an iterative approach, we obtain a small subset of tensors, which is a set of linearly independent tensors, that are sufficient to compute Eqs. 8 and 5.

Similarly, we can use the matrix coefficients $C_y^z \in \mathbb{R}^{d\times d}$ instead of individual scalars to incorporate more detailed relations between the tensor $\mathcal{T}_z$ and any tensors $\mathcal{T}_y$. Therefore, $\mathcal{T}_z$ is tensorialy dependent in $\{\mathcal{T}_y\}_{z\neq y\in M}$ if it satisfies for some matrix coefficients C_y^z the following

$$\mathcal{T}_z = \sum_{z\neq y\in M} \mathcal{T}_y C_y^z. \tag{10}$$

The defined dependency described in Eq. 9 expresses more redundancies than what a standard linear dependency expresses. As a result, we obtain a sparse set of independent tensors that enables us to efficiently compute Eqs. 8 and 5. This set of representative tensors constitutes a dictionary that represents the embedded tensor space.

4.1 Dictionary construction

We use an iterative approach to construct the described dictionary by one sequential scan of the entire data points in M. In the first iteration, we define the scanned set to be $X_1 = \{x_1\}$ and the dictionary $D_1 = \{x_1\}$. At each iteration $s = 2, \ldots, n$, we have a new data point x_s, the scanned set $X_{s-1} = \{x_1, \ldots, x_{s-1}\}$ from the previous iteration and the dictionary D_{s-1} that represents X_{s-1}. The dictionary D_{s-1} is a subset of η_{s-1} data points from X_{s-1} that are sufficient to represent its embedded tensors. We define the scanned set to be $X_s = X_{s-1}\cup\{x_s\}$. Our goal is to define the dictionary D_s of X_s based on the dictionary D_{s-1} with the new data point x_s. To achieve it, a dependency criterion has to be established. If this criterion is satisfied, then the the dictionary remains the same such that $D_s = D_{s-1}$. Otherwise, it is updated to include the new data point such that $D_s = D_{s-1} \cup \{x_s\}$.

We use a dependency criterion, which is similar to the approximated linear dependency (ALD) criterion, used in KRLS (Engel et al. 2004). The ALD measures the distance between candidates vector and the span by the dictionary vectors, to determine if the dictionary has to be updated. In our case, we want to approximate the tensorial dependency (Eq. 10) of the examined tensor $\mathcal{T}_{x_s}$ with the tensors in the dictionary D_{s-1}. Therefore, we define the distance of $\mathcal{T}_{x_s}$ from the dictionary D_{s-1} by

$$\delta_s \triangleq \min_{C_1^s,\ldots,C_{\eta_{s-1}}^s} \left\| \sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{y_j} C_j^s - \mathcal{T}_{x_s} \right\|_F^2, \qquad C_1^s, \ldots, C_{\eta_{s-1}}^s \in \mathbb{R}^{d\times d}, \tag{11}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $C_1^s, \ldots, C_{\eta_{s-1}}^s$ are the matrix coefficients in Eq. 10. We define the approximated tensorial dependency (ATD) criterion to be $\delta_s \leq \mu$, for some accuracy threshold $\mu > 0$. If the ATD criterion is satisfied, then the tensor $\mathcal{T}_{x_s}$ can be approximated by the dictionary D_{s-1} using the matrix coefficients $C_1^s, \ldots, C_{\eta_{s-1}}^s$ that minimizes Eq. 11. Otherwise, the dictionary has to be updated by adding x_s to it.

Lemma 1 *Let $\hat{G}_{s-1} \in \mathbb{R}^{d\eta_{s-1}\times d\eta_{s-1}}$ be a super-kernel of the data points in the dictionary D_{s-1} and let $H_s \in \mathbb{R}^{d\eta_{s-1}\times d}$ be a $\eta_{s-1}\times 1$ block matrix whose jth $d\times d$ block is $G_{(y_j,x_s)}$, $j = 1, \ldots, \eta_{s-1}$. Then, the optimal matrix coefficients (from Eq. 11) are C_j^s, $j = 1, \ldots, \eta_{s-1}$, where C_j^s is the jth $d\times d$ block of the $\eta_{s-1}\times 1$ of $d\times d$ blocks matrix $\hat{G}_{s-1}^{-1} H_s$. The error δ_s in Eq. 11 satisfies $\delta_s = \mathrm{tr}[G_{(x_s,x_s)} - H_s^T \hat{G}_{s-1}^{-1} H_s]$.*

Proof The minimizer in Eq. 11 is rephrased as

$$\delta_s = \min_{C_1^s,\ldots,C_{\eta_{s-1}}^s} \left\{ \text{tr} \left[\left(\sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{y_j} C_j^s - \mathcal{T}_{x_s} \right)^T \left(\sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{y_j} C_j^s - \mathcal{T}_{x_s} \right) \right] \right\}.$$

Algebraic simplifications yield

$$\delta_s = \min_{C_1^s,\ldots,C_{\eta_{s-1}}^s} \left\{ \text{tr} \left[\sum_{i=1}^{\eta_{s-1}} \sum_{j=1}^{\eta_{s-1}} {C_i^s}^T \mathcal{T}_{y_i}^T \mathcal{T}_{y_j} C_j^s - \sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{x_s}^T \mathcal{T}_{y_j} C_j^s \right. \right.$$
$$\left. \left. - \sum_{j=1}^{\eta_{s-1}} {C_j^s}^T \mathcal{T}_{y_j}^T \mathcal{T}_{x_s} + \mathcal{T}_{x_s}^T \mathcal{T}_{x_s} \right] \right\}.$$

The products between the embedded tensors (e.g., $\mathcal{T}_{y_j}^T \mathcal{T}_{x_s}$, $j = 1, \ldots, \eta_{s-1}$) can be replaced with the corresponding super-kernel blocks via Lemma 3.3 in Salhov et al. (2012). We perform these substitutions to get

$$\delta_s = \min_A \text{tr} \left[A^T \hat{G}_{s-1} A - H_s^T A - A^T H_s + G_{(x_s,x_s)} \right], \tag{12}$$

where $A \in \mathbb{R}^{d\eta_{s-1} \times d}$ is a $\eta_{s-1} \times 1$ block matrix whose jth $d \times d$ block is C_j^s, $j = 1, \ldots, \eta_{s-1}$. Solving the optimization (i.e., minimization) in Eq. 12 yields the solution

$$A_s = \hat{G}_{s-1}^{-1} H_s, \tag{13}$$

that proves the lemma. □

Lemma 1 provides an expression for a dictionary-based approximation in super-kernel terms. Essentially, this eliminates the need for having a prior knowledge of the embedded tensors during the dictionary construction. At each iteration s, we consider the criterion $\delta_s < \mu$. Based on this condition, we decide whether to add x_s to the dictionary or to approximate its tensor. The threshold μ is given anyway in advance as a meta-parameter and δ_s is computed by using the expression in Lemma 1, which does not depend on the embedded tensors. Therefore, the dictionary construction processes only the required knowledge of the super-kernel blocks that are needed to compute this expression in every iteration. In fact, the number of required blocks is relatively limited since it is determined by the size of the dictionary and not by the size of the dataset.

4.2 The dictionary construction algorithm

In this section, we specify the steps that take place in each iteration s in the dictionary construction algorithm. Let E_s be a $(\eta_s \times d) \times (s \times d)$ block matrix at iteration s whose (i, j) entry (block) is the optimal coefficient matrix C_i^j, $1 \le i \le \eta_s$, $1 \le j \le s$ computed in Lemma 1. The coefficient matrix C_i^j has two solutions that depend on the ATD criterion calculated in iteration s. First, Lemma 1 is applied to compute δ_s. Then, two possible scenarios are considered:

1. $\delta_s \le \mu$, hence, $\mathcal{T}_{x_s}$ (Eq. 7) satisfies the ATD on D_{s-1}. The dictionary remains the same, i.e., $D_s = D_{s-1}$ and $\hat{G}_s = \hat{G}_{s-1}$. The extension matrix $E_s = [E_{s-1}\ A_s]$ is computed by concatenating the matrix $A_s = \hat{G}_{s-1}^{-1} H_s$ (from the proof of Lemma 1) with the coefficient matrix from the previous iteration $s - 1$.

2. $\delta_s > \mu$, hence, $\mathcal{T}_{x_s}$ does not satisfy the ATD on D_{s-1}. The vector x_s is added to the dictionary, i.e., $D_s = D_{s-1} \cup \{x_s\}$. The computation of E_s is done by adding the identity matrix and the zero matrix in the appropriate places. The dictionary related super-kernel $\hat{G}_s$ becomes

$$\hat{G}_s = \begin{bmatrix} \hat{G}_{s-1} & H_s \\ H_s^T & G_{(x_s,x_s)} \end{bmatrix}. \tag{14}$$

The computation of the ATD conditions during these iterations requires to have $\hat{G}_s^{-1}$ of the dictionary based super-kernels. The block matrix inversion in Eq. 14 becomes

$$\hat{G}_s^{-1} = \begin{bmatrix} \hat{G}_{s-1}^{-1} + A^s \Delta_s^{-1} (A^s)^T & -A^s \Delta_s^{-1} \\ -\Delta_s^{-1} (A^s)^T & \Delta_s^{-1} \end{bmatrix}, \tag{15}$$

where $\Delta_s = G_{(x_s,x_s)} - H_s^T \hat{G}_{s-1}^{-1} H_s$ is computed through the applicability of the ATD test.

At each iteration s, the PTE of the data points X_s (or, more accurately, their patches) is based on the spectral decomposition of the super-kernel G_s of X_s. This spectral decomposition is approximated by using the extension matrix E_s and the dictionary related super-kernel $\hat{G}_s$ (Eq. 14).

Let $E_s^T = Q_s R_s$ be the QR decomposition of this extension matrix, where $Q_s \in \mathbb{R}^{ds \times d\eta_s}$ is an orthogonal matrix and $R_s \in \mathbb{R}^{d\eta_s \times d\eta_s}$ is the upper triangular matrix. Additionally, let $R_s \hat{G}_s R_s^T = \tilde{U}_s \Sigma_s \tilde{U}_s^T$ be the SVD of $R_s \hat{G}_s R_s^T$. Then, the SVD of $E_s^T \hat{G}_s E_s$ is $E_s^T \hat{G}_s E_s = (Q_s \tilde{U}_s) \Sigma_s Q_s^T \tilde{U}_s^T$. Thus, the SVD of G_s is approximated by $G_s \approx U_s \Sigma_s U_s^T$, where

$$U_s = Q_s \tilde{U}_s. \tag{16}$$

The dictionary construction followed by the patch-to-tensor embedding approximation process is described in Algorithm 4.1.

The computational complexity of Algorithm 4.1 is given in Table 1. When $n \gg \eta_n$, the computational complexity $O(nd^3\eta_n^2)$ from the application of the proposed method is substantially lower than the computational complexity $O(d^3n^3)$ from the application of the straightforward SVD. Table 2 presents the computation complexity in comparison to relevant embedding techniques. Hence, the PTEA is significantly more efficient than the PTE (Salhov et al. 2012). Furthermore, the PTEA is more efficient than DM (Lafon 2004) due to the SVD computational complexity of $O(n^3)$. The KRLS (Engel et al. 2004) achieves a reduced computational complexity by a factor d, however, in general the intrinsic dimension d is assumed to be small. Finally, embedding via Tensor Voting (Mordohai and Medioni 2010) is more cost effective.

4.3 The super-kernel approximation error bound

The dictionary construction allows us to approximate the entire super-kernel without a direct computation of every block in it. Given the dictionary construction product E_s, then the super-kernel G_s of the data points in X_s is approximated by

$$G_s \approx E_s^T \hat{G}_s E_s, \tag{17}$$

where $\hat{G}_s \in \mathbb{R}^{\eta_s d \times \eta_s d}$ is the super-kernel of the data points in the dictionary D_s.

For the quantification of Eq. 17, let $\Psi_s \triangleq [\mathcal{T}_{x_1}, \ldots, \mathcal{T}_{x_s}]$ be the $\ell \times sd$ matrix that aggregates all the embedded tensors up to a step s, where $\mathcal{T}_{x_i} \in \mathbb{R}^{l \times d}$, $i = 1, \ldots, s$.

Algorithm 4.1: Patch-to-Tensor Dictionary Construction and Embedding Approximation (PTEA)

Input: Data points: $x_1, \ldots, x_n \in R^m$ and their corresponding local tangential basis $O_{x_i}, i = 1, \ldots, n$.
Parameters: Max approximation tolerance μ, tensor of length ℓ and dataset of size n

Output: The Estimated Tensors $\mathcal{T}_{x_i}, i = 1, \ldots, n$, the extension matrix E_n and the super-kernel $\hat{G}_n$

1: Initialize: $\hat{G}_1 = G_{(x_1,x_1)}$ (block, definition 1), $\hat{G}_1^{-1} = G_{(x_1,x_1)}^{-1}$, $E_1 = I_d$, $D_1 = \{x_1\}$, $\eta_1 = 1$

2: **for** $s = 2$ *to* n **do**

Compute $H_s = \left[G_{(x_1,x_s)}, \ldots, G_{(X_{\eta_s},x_s)}\right]$

ATD Test:

Compute $A_s = \hat{G}_{s-1}^{-1} H_s$

Compute $\Delta = G_{(x_s,x_s)} - H_s^T A_s$

Compute $\delta = \mathrm{tr}[\Delta]$

If $\delta < \mu$

Set $E_s = \begin{bmatrix} E_{s-1} & A_s \end{bmatrix}$

Set $D_s = D_{s-1}$

Else (update dictionary)

Set $E_s = \begin{bmatrix} E_{s-1} & 0 \\ 0 & I_d \end{bmatrix}$

Set $D_s = D_{s-1} \cup \{x_s\}$

Update $\hat{G}_s$ according to Eq. 14

Update $\hat{G}_s^{-1}$ according to Eq. 15

Set $\eta_s = \eta_{s-1} + 1$

3: Approximate U_n according Eq. 16.

4: Use U_n to compute the approximated spectral map $\Phi(o_x^j)$, $x \in M$, $j = 1, \ldots, d$, according to Eq. 6 (considering the first ℓ eigenvalues and the corresponding eigenvectors).

5: Use the spectral map Φ to construct the tensors $\mathcal{T}_x \in \mathbb{R}^\ell \otimes \mathbb{R}^d$, $x \in M$, according to Eq. 7.

Table 1 PTEA computational complexity

Operation	Complexity
Computation of δ_s	$O(\eta_s^2 d^3)$
Update of G_s,E_s and G_s^{-1}	$O(\eta_s^2 d^3)$
QR of E_s	$O(s\eta_s^2 d^3)$
SVD of $R_s \hat{G}_s R_s^T$	$O(\eta_s^3 d^3)$
$Q_s \tilde{U}_s$ that approximates $\mathcal{T}_{x_i}$	$O(s\eta_s^2 d^3)$

Table 2 Computational complexity comparison

Method	Complexity
PTEA	$O(n\eta_n^2 d^3)$
PTE	$O(\eta_n^3 d^3)$
DM	$O(n^3 + n^2 m)$
KRLS	$O(n\eta_s^2 + n^2 m)$
Tensor voting	$O(nmlog(n))$

Let $\hat{\Psi}_s \triangleq [\mathcal{T}_{y_1}, \ldots, \mathcal{T}_{y_{\eta_s}}]$ be the $\ell \times \eta_s d$ matrix that aggregates all the embedded tensors of the dictionary members D_s and let $\Psi_s^{\mathrm{res}} \triangleq \Psi_s - \hat{\Psi}_s E_s = [\psi_1^{\mathrm{res}}, \ldots, \psi_s^{\mathrm{res}}]$, where $\psi_s^{\mathrm{res}} \triangleq \mathcal{T}_{x_s} - \sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{y_j} C_j^s$. Then, due to Eq. 11 and the ATD condition, $\left\|\Psi_s^{\mathrm{res}}\right\|_F^2 \leq s\mu$.

From Definition 1 of the super-kernel, we get $G_s = \Psi_s^T \Psi_s$ and $\hat{G}_s = \hat{\Psi}_s^T \hat{\Psi}_s$. Therefore, by substituting Ψ_s into Eq. 17 we get $G_s = E_s \hat{G}_s E_s^T + (\Psi_s^{\text{res}})^T \Psi_s^{\text{res}}$ where all the cross terms vanish by the optimality of E_s. As a consequence, the approximation error $R_s = G_s - E_s \hat{G}_s E_s^T = (\Psi_s^{\text{res}})^T \Psi_s^{\text{res}}$ in step s satisfies $\|R_s\|_F^2 \leq s\mu$. In particular, $\|R_n\|_F^2 \leq n\mu$ for $s = n$.

4.4 Out-of-sample extension for vector fields

The constructed dictionary estimates a tangential vector field by using a recursive algorithm similar to the well known Kernel Recursive Least Squares (KRLS) (Engel et al. 2004). In a supervised learning scheme, the predictor in Eq. 5 is designed to minimize the l_2 distance between the predicted vector field at each iteration s and the actual given vector field (as part of the training set) by

$$J(\mathbf{w}) = \sum_{i=1}^{s} \|\hat{v}(x_i) - v(x_i)\|_2^2 = \sum_{i=1}^{s} \left\| \sum_{j=1}^{t} G_{(x_i,x_j)} \mathbf{w}_j - v(x_i) \right\|_2^2 = \|G\mathbf{w} - \boldsymbol{v}\|_2^2, \quad (18)$$

where $\mathbf{w}$ is the predictor weights vector and $\boldsymbol{v}$ is the concatenation of all the given training values of the vector field.[2] The Recursive Least Squares solution, which minimizes $J(\mathbf{w})$, is given by $\mathbf{w}_o = G^{-1}\boldsymbol{v}$. In the case when the number of vector examples is large, the complexity of inverting the super-kernel tends to be expensive in terms of computational complexity and memory requirements. Furthermore, the length of the predictor weight vector $\mathbf{w}_o$ depends on the number of training samples. Hence, redundant samples, which generate linearly-dependent rows in the super-kernel, will generate over fitting by using the predictor. One possible remedy for these problems is to utilize the sparsification procedure from Sect. 4.2 in order to design the predictor. The optimizer $\mathbf{w}_o$ is formulated by introducing the dictionary estimated super-kernel as $J(\mathbf{w}) = \|G\mathbf{w} - \boldsymbol{v}\|_2^2 \approx \|E_s^T \hat{G}_s E_s w - \boldsymbol{v}\|_2^2$ using Eq. 17. Let $\alpha = E_s w$ and let α_j be the jth element of vector α, then the predictor is reformulated as $\hat{\mathbf{v}}(x_i) = \sum_{j=1}^{s} C_j^s G_{(x_j,x_i)} \alpha_j$, and the corresponding predictor's l_2 error is given by $J(\alpha) = \|E_s^T \hat{G}_s E_s w - f\| = \|E_s^T \hat{G}_s \alpha - \boldsymbol{v}\|$, which can be minimized by

$$\alpha_o = (E_s^T \hat{G}_s)^{\dagger} f = \hat{G}_s^{-1} (E_s E_s^T)^{-1} f. \quad (19)$$

The predictor coefficients α_o is computed using the dictionary-based super-kernel $\hat{G}_s$ and the corresponding extension matrix E_s. It is important to note that in some applications, it is sufficient to consider only the dictionary members and the corresponding kernel plus vectors sampled from the given vector field. In this case, the relevant extension matrix E_s is the identity matrix. Reformulating Eq. 19 accordingly yields that the extension/predictor coefficients design is

$$\alpha_o = \hat{G}_s^{-1} f. \quad (20)$$

5 Data analysis illustration via dictionary based PTE

The PTE methodology provides a general framework that can be applied to a wide varieties of data analysis tasks such as clustering, classification, anomaly detection and manifold

[2] For the sake of simplicity, we use this slight abuse of notation. Namely, the same notation is used for the vector field and for the aggregation of its known values.

learning. In this section, we demonstrate how the proposed dictionary construction based approximated PTE is utilized to solve several relevant data analysis applications such as: I. MNIST Handwritten digit classification. II. Image segmentation. III. Vector field extension over a sphere. The experiments were done on an off-the-shelf PC with a I7-2600 quad core CPU and a 32 GB of DDR3 memory.

5.1 Example I: MNIST handwritten digit classification

The computerized handwritten character recognition has been extensively studied. Relevant papers such as Dimond (1958), Bomba (1959) can be found from the late 50s of the last century. Since then, this challenge has remained relevant and many more studies have been published (Suen et al. 1980). Handwritten characters can be recognized in many different ways, such as K-Nearest Neighbors (Keysers et al. 2007), linear and nonlinear classifiers, neural nets and SVMs (Lecun et al. 1998), to name some.

The MNIST database of handwritten digits (Lecun et al. 1998) (available from http://yann.lecun.com/exdb/mnist/) consists of a training set of 60,000 examples and a test set of 10,000 examples. Each digit is given as a gray level image of size 28×28 pixels. The digit images were centered by computing the center of mass of the pixels. Then, the image was translated to the position of this data point at the center of the 28×28 field. MNIST is a subset of a larger available set from NIST.

Currently, convolutional networks achievements are considered as the state-of-the-art in recognition accuracy with an error of 0.23 % (Ciresan et al. 2012). For our purpose, the MNIST dataset provides 70,000 data points of very high dimensional measurements having 728 pixels per a measured digit. In our experiments, the images were used as is.

The PTEA Algorithm 4.1 was applied to embed the MNIST dataset of 70,000 examples (data points) using the following steps: 1. For each data point, we identified the 300 nearest neighbors and computed the corresponding local PCA og this neighborhood. For each local tangential space, we kept the 2 significant eigenvectors. 2. The diffusion affinity (Eq. 1) was computed with $\varepsilon = 105$ which is the Euclidean distance mean for all the pairwise data points. The proposed dictionary construction with $\mu = 0.0002/3$ identified 381 important data points and their corresponding local tangential spaces. 3. The approximated tensors were constructed utilizing a tensor length $l = 14$. The labeling of each test data point was determined by using the label of the nearest training data point where the pairwise distance was computed by the Frobenius norm of the difference between the corresponding embedded tensors. The software-based implementation of the PTEA algorithm is described in Salhov et al. (2015).

The resulting labeling error from the application of the PTEA based recognition method (Algorithm 4.1) is 7 %. Table 3 compares between the computational complexities from the application of the straightforward SVD algorithm and from the application of the PTEA algorithm where both apply to the MNIST dataset.

The actual running times are summarized in the Table 4. Total time of 443 min or 7.4 h is short in comparison to the expected time of a standard spectral decomposition.

Although we are not far away from the state-of-the-art result in digit recognition, the proposed method has the following advantages: 1. It shows that patch processing can be utilized for recognition and data analysis tasks. 2. High dimensional big datasets can be processed on a "cheap" hardware such as in our case where the algorithm was executed on less than 1000$ worth of hardware.

Table 3 Performance comparison between the application of the approximated SVD step in the PTEA Algorithm 4.1 and the application of the full SVD to the NIST dataset

Dataset	Size	d	SVD complexity—full G	SVD complexity—approx. G PTEA	Dict. size
MNIST	70,000	2	$O\left(70,000^3 \times 2^3\right)$	$O\left(70,000 \times 145,161 \times 2^3\right)$	381

d is the estimated intrinsic dimension, SVD complexity—full G is the computational complexity to estimate a full kernel decomposition, SVD complexity—approx. G is the computational coomplexity to estimate the decomposition of the approximated kernel according to Eq. 16 and dict. size is the number of dictionary members

Table 4 Comparisons between the computational complexities of different methods

Operation	Measured time (min)
300NN	126
Local PCA	50
PTEA	267
Total	443

5.2 Example II: Image segmentation

Image segmentation aims to cluster pixels into image regions that correspond to individual surfaces, objects or natural parts of objects. Image segmentation plays a key rule in many computer vision tasks.

Under the PTEA framework, the image is viewed as a super-kernel construction that reflects the affinities between the pixels and the projection of the related tangential spaces. The PTE construction translates a given pixel related features into tensors in the embedded space. The image segmentation is achieved through tensorial clustering in the embedded space.

For the image segmentation examples, we utilized the pixel color information and its spatial (x,y) location multiplied by a scaling factor $w = 0.1$. Hence, given a RGB image with $Ix \times Iy$ pixels, we generated the dataset X of size $5 \times (Ix \cdot Iy)$.

Algorithm 4.1 embeds X into a tensorial space. The first step in Algorithm 4.1 constructs local patches. Each generated patch captures the relevant neighborhood and considers both color and spatial similarities. Hence, a patch is more likely to include attributes related to spatially close pixels. It is important to note that the affinity kernel is computed according to Eq. 1 where ε equals to the mean Euclidean distance between all pairs in X. The PTE parameters l and ρ were chosen to generate homogeneous segments. The dictionary's approximation tolerance μ was chosen arbitrarily to be $\mu = 0.001$. The k-means algorithm with sum of square differences clustered the tensors into similar clusters. The final clustering results from the embedded tensors as a function of the diffusion transition time t (Eq. 6) are presented in Figs. 2 and 3.

Figures 2 and 3 are the segmented outputs from the application of the PTEA Algorithm 4.1. In each figure, (a) is the original image. The images' sizes are 40×77 and 104×128, respectively. Each figure describes the output from the segmentation as a function of the diffusion parameter t. The effect of the diffusion transition time on the segmentation of the 'Hand' image is significant. For example, the first three images in Fig. 2, which correspond to $t = 1$, $t = 2$ and $t = 3$, respectively, show a poor segmentation results. As t increases, the

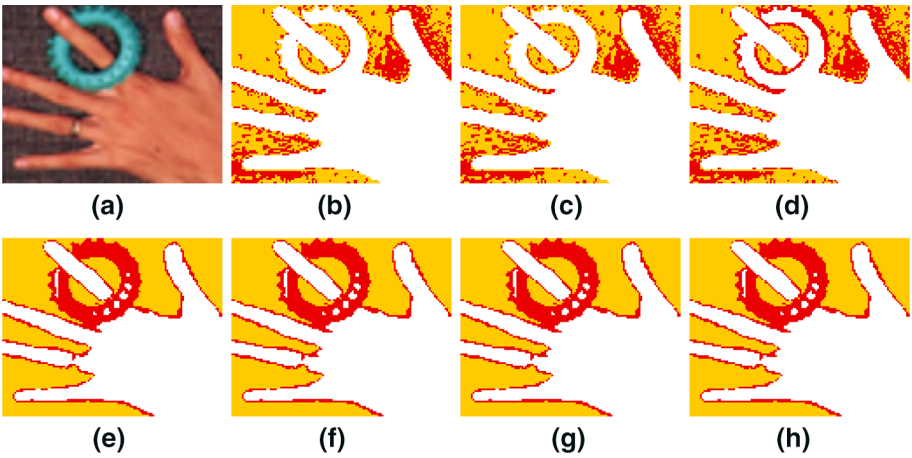

Fig. 2 Segmentation results from the application of the PTEA Algorithm 4.1 to 'Hand' when $l = 10$ and $d = 10$. **a** Original image. **b** Segmentation when $t = 1$. **c** Segmentation when $t = 2$. **d** Segmentation when $t = 3$. **e** Segmentation when $t = 4$. **f** Segmentation when $t = 5$. **g** Segmentation when $t = 6$. **h** Segmentation when $t = 7$

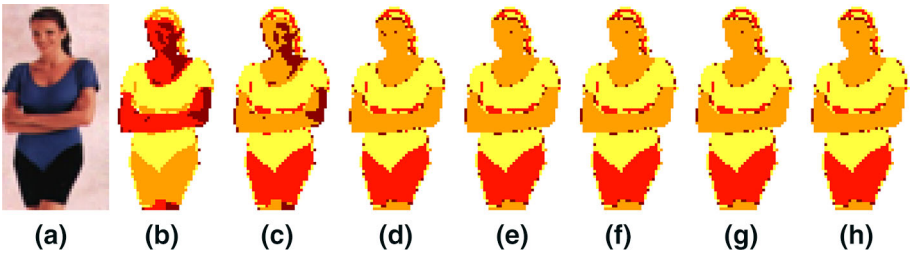

Fig. 3 Segmentation results from the application of the PTEA Algorithm 4.1 to 'Sport' with $l = 10$ and $d = 20$. **a** Original image. **b** Segmentation when $t = 1$. **c** Segmentation when $t = 2$. **d** Segmentation when $t = 3$. **e** Segmentation when $t = 4$. **f** Segmentation when $t = 5$. **g** Segmentation when $t = 6$. **h** Segmentation when $t = 7$

segmentation becomes more homogeneous, thus the main structures in the original image can be separated such as $t = 4$ in (e). Another consequence from the increase in the diffusion transition time parameter t is the smoothing effect on the pairwise distances between data points in the embedded space. By increasing t, the pairwise distances between similar tensors are reduced while he distances between dissimilar tensors increase.

Table 5 compares between the estimated computational complexity from the application of the approximated SVD by the PTEA Algorithm 4.1 and the computational complexity from the application of the full SVD by the PTE. The generated dataset for each image consists of 13312 and 3080 data points, respectively. The computational complexity from the application of the approximated SVD is significantly lower from the application of the full SVD in both cases.

The constructed dictionary enables us to practically utilize the PTE to segment medium sized images. The computational complexities were significantly lower in comparison to the application of the SVD decomposition that utilizes the full super-kernel. The reduction in the computational complexity of image segmentation was achieved by the application of the dictionary-based SVD/QR step of the extension coefficient matrix E in the PTEA algorithm.

Table 5 Performance comparison between the application of the approximated SVD step in the PTEA Algorithm 4.1 and the application of the full SVD to imaging datasets

Image	Size	d	SVD complexity—full G	SVD complexity—approx. G PTEA	Dict. size
Hand	104×128	2	$O\left(26624^3\right)$	$O\,(26624 \times 16)$	2
Sport	40×77	2	$O\left(6160^3\right)$	$O\,(6160 \times 16)$	2

d is the estimated intrinsic dimension, SVD complexity—full G is the computational complexity to estimate a full kernel decomposition, SVD complexity—approx. G is the computational complexity to estimate the decomposition of the approximated kernel according to Eq. 16 and dict. size is the number of dictionary members

5.3 Example III: Vector field extension over a sphere

In this example, we utilize a synthetic vector field F. The synthetic vector field consists of $|M| = N = 8050$ data points sampled from a two dimensional half sphere $\mathcal{M}$ that is immersed in $\mathbb{R}^3$ by $F : M \to \mathbb{R}^3$ such that for any $x \in M$, we have $F(x) \in T_x(\mathcal{M})$. For each two-dimensional data point, we generate a vector that lies on the tangent plane of the sphere at a corresponding data point. Figure 4 illustrates the described vector field from two views.

The dictionary in this example is constructed using Algorithm 4.1 with the meta-parameters $\varepsilon = 0.0218$ and $\mu = 0.0005$. The resulting dictionary contains $\eta_N = 414$ members. Figure 5 presents the dictionary members that provide an almost uniform sampling of the sphere.

Under the above settings, the vector $O_x^T F(x)$ is a coordinate vector of $F(x)$ in the local basis of $T_x(\mathcal{M})$. Our goal is to extend F to any data point $y \in \mathcal{M}\backslash M$. In our example, for each data point $x \in M$, we have a closed form for its local patch. Each patch at the data point $x = (x_1, x_2)$ is spanned by two vectors $S_1 = (1, 0, -x_1\sqrt{1 - x_1^2 - x_2^2})$ and $S_2 = (0, 1, -x_2\sqrt{1 - x_1^2 - x_2^2})$. Hence, the corresponding tangential space O_x is given by the application of SVD to $[S_1, S_2]^T$. Furthermore, we choose the vectors from the vector field

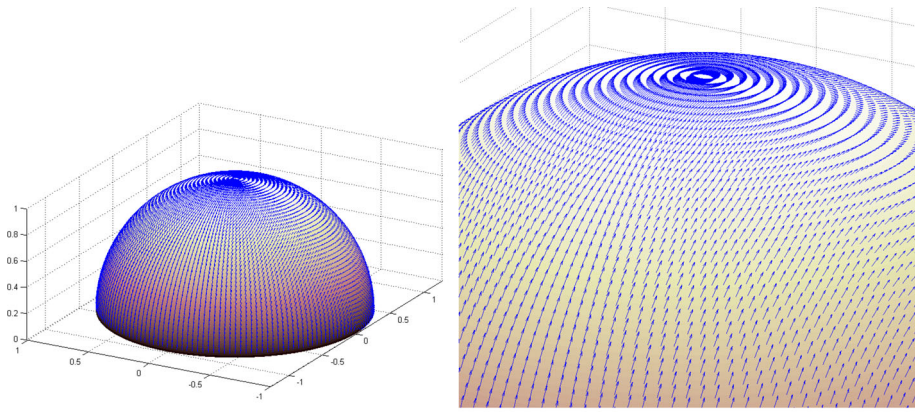

Fig. 4 The generated vector field. **a** Full view. **b** Zoom

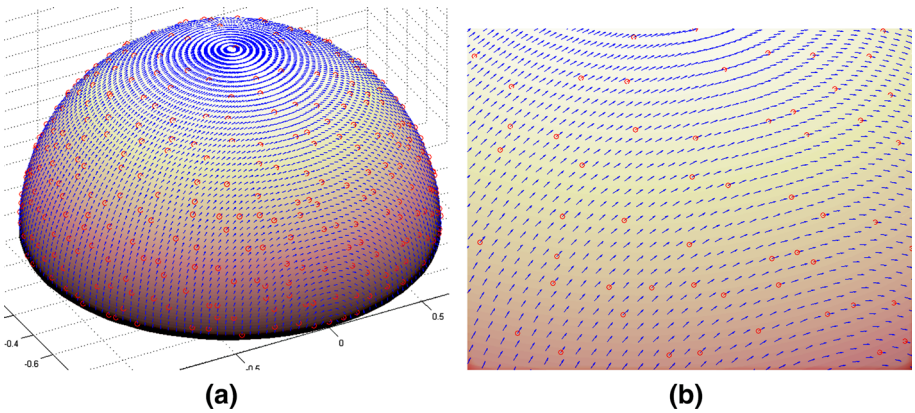

Fig. 5 The generated vector field (*arrows*) and the chosen dictionary members (*circles*). **a** Full view. **a** Zoom

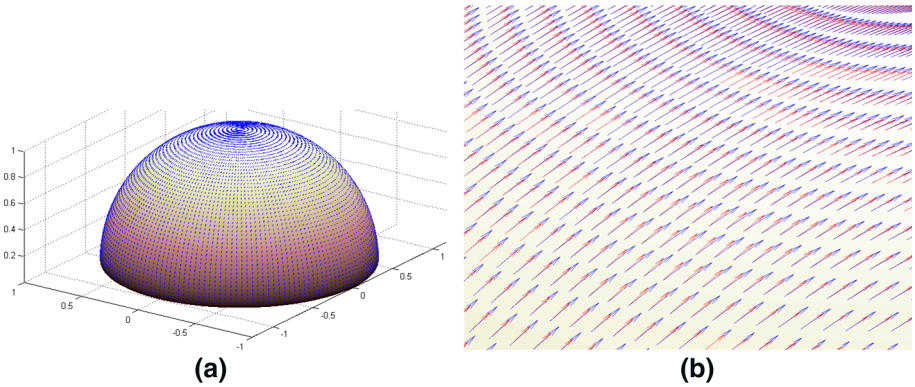

Fig. 6 The extended vector field (*red*) and the given vector field (*blue*). **a** Full view. **b** Zoom (Color figure online)

to be the corresponding vectors $F(x) = (1, 0, -x_1\sqrt{1 - x_1^2 - x_2^2})$. Equation 20 provides the solution to the out-of-sample extension problem given the vector field members f that corresponds to the dictionary members and the corresponding super-kernel G_s. For each data point y, which is not in the dictionary, we apply Eq. 5 to find the extended vector field. The resulting vector field is compared with the original vector field in Fig. 6.

In order to evaluate the performance of this out-of-sample extension, we compared between the length and the direction of the corresponding vector in the original vector field. Figure 7 displays the cumulative distribution function (CDF) of these squared errors. The CDF in Fig. 7 suggests that, in comparison with the ground truth vector field, about 93 % of the estimated vectors have a vector length square error of less than 10^{-2}, and 95 % have vector direction square error of less than 2×10^{-2} radians.

6 Conclusions

The proposed construction in this paper extends the dictionary construction in Engel et al. (2004) by using the LPD super-kernel from Salhov et al. (2012), Wolf and Averbuch

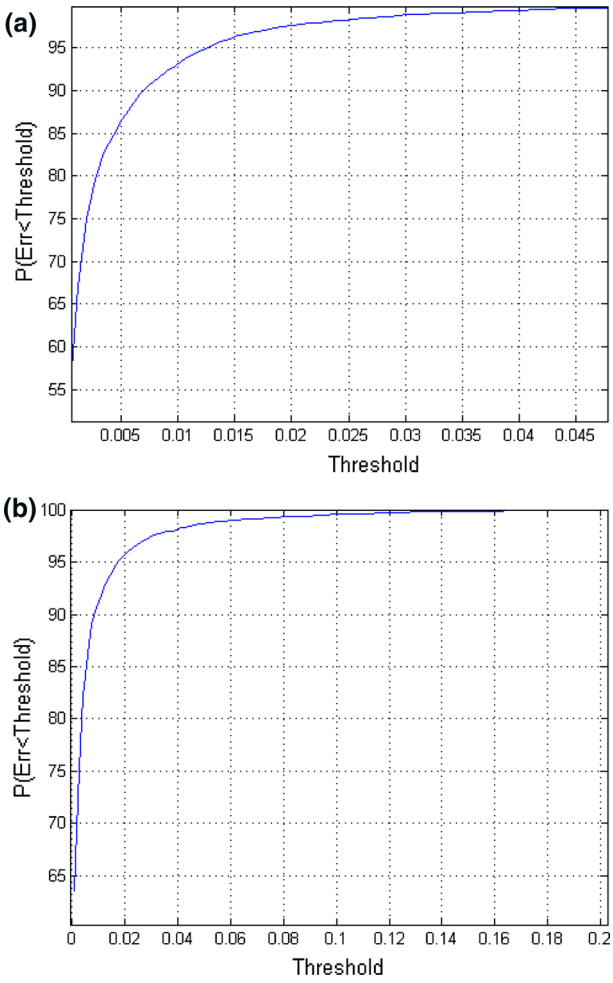

Fig. 7 The CDF (y-axis) of MSE of **a** estimated vector length. **b** Estimated vector direction

(2013). This is achieved by an efficient dictionary-based construction assuming the data is sampled from an underlying manifold while utilizing the non-scalar relations and the similarities among manifold patches instead of representing them by individual data points. The constructed dictionary contains these patches from the underlying manifold, which are represented by the embedded tensors from Salhov et al. (2012), instead of representing them by individual data points. Therefore, it encompasses multidimensional similarities between local areas in the data. Furthermore, the dictionary insertion criterion is based on tensorial linear dependencies that incorporate both distances on the manifold as well as relations between its tangential spaces. Therefore, dictionary representatives are expected to either be sufficiently far from each other on the manifold or have sufficiently different tangent spaces. As a result, this criterion is able to adapt the dictionary selection to the curvatures of the manifold, since it allows a denser selection in high-curvature areas and a sparser one in low-curvature areas. The PTEA Algorithm reduces the computational complexities of the spectral analysis in comparison to the regular use of PTE based algorithm.

Acknowledgments This research was supported by the the Israel Ministry of Science & Technology (Grants No. 3-9096, 3-10898), by US–Israel Binational Science Foundation (BSF 2012282), by Blavatnik Funds and by a Fellowship from University of Jyväskylä. The third author was supported by the Eshkol Fellowship from the Israel Ministry of Science & Technology.

References

Ballester, C., Bertalmio, M., Sapiro, G., & Verdera, J. (2001). Filling-in by joint interpolation of vector fields and gray levels. *IEEE Transactions on Image Processing*, *10*, 1200–1211.

Belkin, M., & Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, *15*(6), 1373–1396.

Bermanis, A., Averbuch, A., & Coifman, R. (2013). Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, *34*, 182–203.

Bomba, J. S. (1959). *Alpha-numeric character recognition using local operations*. In Papers presented at the December 1–3, 1959, eastern joint IRE-AIEE-ACM computer conference, ACM, New York, NY, USA, IRE-AIEE-ACM '59 (Eastern) (pp. 218–224).

Bronstein, M., & Bronstein, A. (2011). Shape recognition with spectral distances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(5), 1065–1071.

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3642–3649).

Coifman, R., & Lafon, S. (2006). Diffusion maps. *Applied and Computational Harmonic Analysis*, *21*(1), 5–30.

Cox, T., & Cox, M. (1994). *Multidimensional scaling*. London: Chapman and Hall.

Cullum, J. K., & Willoughby, R. A. (2002). *Lanczos algorithms for large symmetric eigenvalue computations*, (Vol. 1). Society for Industrial and Applied Mathematics.

David, G. (2009). Anomaly detection and classification via diffusion processes in hyper-networks. PhD thesis, School of Computer Science, Tel Aviv University.

David, G., & Averbuch, A. (2012). Hierarchical data organization, clustering and denoising via localized diffusion folders. *Applied and Computational Harmonic Analysis*, *33*(1), 1–23.

Dimond, T. L. (1958). *Devices for reading handwritten characters*. In Papers and discussions presented at the December 9–13, 1957, eastern joint computer conference: Computers with deadlines to meet, ACM, New York, NY, USA, IRE-ACM-AIEE '57 (Eastern) (pp. 232–237).

Engel, Y., Mannor, S., & Meir, R. (2004). The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, *52*(8), 2275–2285.

Fowlkes, C., Belongie, S., Chung, F., & Malik, J. (2004). Spectral grouping using the Nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 214–225.

Fuselier, E. J., & Wright, G. (2009). Stability and error estimates for vector field interpolation and decomposition on the sphere with RFBs. *SIAM Journal on Numerical Analysis*, *47*(5), 3213–3239.

Golub, G., & Van Loan, C. (2012). *Matrix computations* (4th ed.). Baltimore: John Hopkins University Press.

Jingen, L., Yang, Y., & Shah, M. (2009). Learning semantic visual vocabularies using diffusion distance. In *IEEE conference on computer vision and pattern recognition, 2009. CVPR 2009* (pp. 461–468).

Keller, Y., Coifman, R., Lafon, S., & Zucker, S. (2010). Audio-visual group recognition using diffusion maps. *IEEE Transactions on Signal Processing*, *58*(1), 403–413.

Keysers, D., Deselaers, T., Gollan, C., & Ney, H. (2007). Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*, 1422–1435.

Kruskal, J. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, *29*, 1–27.

Lafon, S. (2004). Diffusion maps and geometric harmonics. PhD thesis, Yale University.

Lafon, S., Keller, Y., & Coifman, R. (2006). Data fusion and multicue data matching by diffusion maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(11), 1784–1797.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

Liu, S., & Trenkler, G. (2008). Hadamard, Khatri-Rao, Kronecker and other matrix products. *International Journal of Information & Systems Sciences*, *4*(1), 160–177.

Mordohai, P., & Medioni, G. G. (2010). Dimensionality estimation, manifold learning and function approximation using tensor voting. *Journal of Machine Learning Research*, *11*, 411–450.

Ouimet, M., & Bengio, Y. (2005). Greedy spectral embedding. In *Proceedings of the tenth international workshop on artificial intelligence and statistics* (pp. 253–260).

Rao, C. K. C. (1968). Solutions to some functional equations and their applications to characterization of probability distributions. *Sankhya: The Indian Journal of Statistics, Series A (1961–2002)*, *30*(2), 167–180.

Rui, X., Damelin, S., & Wunsch, D. (2007). Applications of diffusion maps in gene expression data-based cancer diagnosis analysis. In: *Engineering in medicine and biology society, 2007. EMBS 2007. 29th annual international conference of the IEEE* (pp. 4613–4616).

Salhov, M., Wolf, G., & Averbuch, A. (2012). Patch-to-tensor embedding. *Applied and Computational Harmonic Analysis*, *33*(2), 182–203.

Salhov, M., Bermanis, A., Wolf, G., & Averbuch, A. (2015). PTEA matlab implementation. http://www.cs.tau.ac.il/~moshesa/HA/PAtchDiffusion.rar.

Schclar, A., Averbuch, A., Rabin, N., Zheludev, V., & Hochman, K. (2010). A diffusion framework for detection of moving vehicles. *Digital Signal Processing*, *20*(1), 111–122.

Singer, A., & Coifman, R. (2008). Non-linear independent component analysis with diffusion maps. *Applied and Computational Harmonic Analysis*, *25*(2), 226–239.

Singer, A., & Wu, H. (2011). Orientability and diffusion maps. *Applied and Computational Harmonic Analysis*, *31*(1), 44–58.

Singer, A., & Wu, H. (2012). Vector diffusion maps and the connection laplacian. *Communications on Pure and Applied Mathematics*, *65*(8), 1067–1144.

Suen, C., Berthod, M., & Mori, S. (1980). Automatic recognition of handprinted characters; the state of the art. *Proceedings of the IEEE*, *68*(4), 469–487.

Talmon, R., Cohen, I., & Gannot, S. (2011). Supervised source localization using diffusion kernels. In *2011 IEEE workshop on applications of signal processing to audio and acoustics (WASPAA)* (pp. 245–248).

Talmon, R., Kushnir, D., Coifman, R., Cohen, I., & Gannot, S. (2012). Parametrization of linear systems using diffusion kernels. *IEEE Transactions on Signal Processing*, *60*(3), 1159–1173.

Talmon, R., Cohen, I., & Gannot, S. (2013). Single-channel transient interference suppression with diffusion maps. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(1–2), 132–144.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*, 395–416.

Wolf, G., & Averbuch, A. (2013). Linear-projection diffusion on smooth Euclidean submanifolds. *Applied and Computational Harmonic Analysis*, *34*, 1–14.