Dictionary Construction for Patch-to-Tensor Embedding

Moshe Salhov Amit Bermanis Guy Wolf Amir Averbuch
School of Computer Science, Tel Aviv University, Tel Aviv 69978
Israel

Abstract

The incorporation of a matrix relations, which encompass multidimensional similarities between local neighborhoods of data points in the underlying manifold of a data, improves the utilization of kernel based data analysis methodologies. However, the utilization of multidimensional similarities results in a larger kernel and hence the computational complexity of the corresponding spectral decomposition increases dramatically. In this paper, we propose an efficient approximation to a spectral decomposition of a multidimensional similarity based kernel. Furthermore, we propose a dictionary construction that approximates the oversized kernel in this case and its associated embedding. The performance of the proposed dictionary construction is demonstrated on an example of a super-kernel that utilizes the Diffusion Maps methodology together with linear-projection operators between tangent spaces in the manifold.

I. Introduction

Recent methods for advanced massive high dimensional data analysis utilize a manifold structure on which data points are assumed to lie. This manifold is immersed (or submersed) in an ambient space that is defined by observable parameters. Kernel methods such as k-PCA and Diffusion Maps (DM) [4] have provided good results in analyzing such massive high dimensional data. The defined kernel can be thought of as an adjacency matrix of a graph whose vertices are the data points in the dataset. The analysis of the eigenvalues and the corresponding eigenvectors of this matrix reveals many properties and connections in the graph. These methods are based on the spectral decomposition of a kernel that was designed to incorporate a scalar similarity measure between data points. The resulting embedding of the data points into an Euclidean space preserves the qualities represented by the designed kernel. This approach extends the core of the classical Multi-Dimensional Scaling (MDS) method [6], [9] by considering non-linear relations instead of just a linear one in its original Gram matrix.

Recently, DM was extended in several different ways to handle the orientation in local tangent spaces [10]–[13]. The relation between two patches 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 can be used to analyze local areas around data points instead of analyzing their specific locations. For example, this analysis can be beneficial in image processing (analyzing regions instead of individual pixels) and when the data points are perturbed so that their surrounding area is more important than their specific position. Since the constructions of these similarities are based on local tangent spaces, they provide methods to manipulate tangential vector fields (e.g., perform out-of-sample extensions). 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 [2] is to recover missing data in images utilizing interpolation of the appropriate vector field. Another example is the utilization of tangential vector fields interpolation on $\mathcal{S}^2$ for modeling atmospheric air flow and oceanic water velocity [8].

The discussed enrichments increase considerably the kernel size. Kernel size is a limiting factor in the applicability of spectral decomposition based data analysis methods to real problems and considerable efforts have been invested for example in [1], [7] and others in approximating the spectral decomposition operator. The dictionary approach presented in [7] constructs a dictionary and the corresponding scalar

kernel plus the necessary extension coefficients that approximate the full scalar kernel. The number of dictionary members depends on the given data, kernel configuration and on the designed parameter that controls the quality of the full kernel approximation.

In this paper, we utilize the dictionary construction approach from [7] to approximate the spectral decomposition of a non-scalar kernel that utilizes the underlying patch structure. We describe the necessary condition for updating a non-scalar dictionary for achieving a bound on the approximation error. Although the proposed method is applicable to many such kernels, we focus on the linear-projection super-kernel construction described in [10]. The super-kernel construction there analyzes patches in the manifold instead of analyzing single data points on 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 two patches is described by a matrix, which represents both the affinity between data points at the centers of these patches and the similarity between their local coordinate systems. Then, the constructed matrices between all patches are combined into a block matrix that is called super-kernel.

The paper has the following structure. Preliminaries are presented in Section II. Section III formulates the problem. The dictionary construction and its properties are presented in Section IV. Finally, Section V displays the experimental results on image segmentation derived from the utilization of a dictionary based analysis.

II. Preliminaries

A. Manifold Setup

Let $M \subseteq \mathbb{R}^m$ be a set of n data points sampled from a manifold $\mathcal{M} \subseteq \mathbb{R}^m$ that lies in the ambient space $\mathbb{R}^m$. Let $d \ll m$ be the intrinsic dimension of $\mathcal{M}$, then, at every 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 tangent 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 $o_x^i = (o_x^{i1}, \ldots, o_x^{im})^T$, $i = 1, \ldots, d$, 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:

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

From now on we assume that the vectors in $T_x(\mathcal{M})$ are expressed by their d coordinates according to the presented basis $o_x^1, \ldots, o_x^d$. For each vector $u \in T_x(\mathcal{M})$, the vector $\tilde{u} = O_x u \in \mathbb{R}^m$ 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 the linear projection of v on the tangent space $T_x(\mathcal{M})$.

B. Diffusion Maps

The original Diffusion Maps method [4] 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

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

where ε is a meta-parameter of the algorithm. This kernel represents the affinities between data points in the manifold. The kernel can be viewed as a construction of a weighted graph over the dataset M. The data points in M are the vertices and the weights (for example, we can use the weight in Eq. II.2) 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

$$\bar{p}(x,y) = \frac{k(x,y)}{\sqrt{q(x)q(y)}} = \sqrt{q(x)}p(x,y)\frac{1}{\sqrt{q(y)}} \qquad x, y \in M. \tag{II.3}$$

The DM method embeds the manifold into an Euclidean space whose dimensionality is usually significantly lower than the original dimensionality. This embedding is a result from the spectral analysis of the diffusion affinity kernel $\bar{P}$. The eigenvalues $1 = \sigma_0 \geq \sigma_1 \geq \ldots$ 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 $\bar{\Phi}(x) = (\sigma_i \bar{\phi}_i(x))_{i=0}^{\delta}$ 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 $\bar{P}$.

C. Linear-Projection Diffusion Super-Kernel

For $x, y \in M$, define $O_{xy} = O_x^T O_y \in \mathbb{R}^{d \times d}$ where O_x and O_y were defined in Eq. II.1. The matrices O_x and O_y represent bases for the tangent spaces $T_x(\mathcal{M})$ and $T_y(\mathcal{M})$, respectively. Thus, the matrix O_{xy} represents a linear-projection between these tangent spaces, and, in some sense, the similarity between them. As in [10], it is referred to as a tangent similarity matrix.

Let $\Omega \in \mathbb{R}^{n \times n}$ be a symmetric and positive semi-definite affinity kernel defined on $M \subseteq \mathbb{R}^m$, i.e., 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. In addition, assume that $\omega(x,y) \geq 0$ for every $x, y \in M$. The diffusion affinity kernel is an example of such an affinity kernel. Definition II.1 uses the tangent similarity matrices and the affinity kernel Ω to define the Linear-Projection *super-kernel*. When the diffusion affinities in $\bar{P}$ are used, instead of using the general affinities in Ω, this super-kernels is called a Linear-Projection Diffusion (LPD) super-kernel.

Definition II.1 (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 $N(y)$. Each block $G_{(x,y)} \in \mathbb{R}^{d \times d}$ of G is defined as $G_{(x,y)} \triangleq \bar{p}(x,y)O_{xy} = a(x,y)O_x^T O_y,\ x, y \in M$.*

The super-kernel in Definition II.1 encompasses both the diffusion affinities between data points on the manifold $\mathcal{M}$ and the similarities between their tangent spaces. The latter are expressed by the linear-projection operators between tangent spaces. Specifically, for two tangent 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 tangent 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 [10], [13].

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 $[G_{(x,y)}]_{ij}$, $1 \leq i, j \leq d$, in the block $G_{(x,y)}$, where $x, y \in M$. 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 $N(x)$, $x \in M$, is a manifold, into a tensor space. Let $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_\ell|$ 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

can also be regarded as 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

$$\Phi(o_x^j) = (\lambda_1^t \phi_1(o_x^j), \ldots, \lambda_\ell^t \phi_\ell(o_x^j)), \tag{II.4}$$

which is similar to the one used in DM and where 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

$$\mathcal{T}_x \triangleq \begin{pmatrix} | & & | \\ \Phi(o_x^1) & \cdots & \Phi(o_x^d) \\ | & & | \end{pmatrix} \qquad x \in M. \tag{II.5}$$

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 $\mathcal{T}_x$ represents an embedding of the patch $N(x)$, $x \in M$, into the tensor space $\mathbb{R}^\ell \otimes \mathbb{R}^d$.

III. Problem Formulation

Under the manifold settings of Section II-A, we assume to have an access to a dataset of n data points that are sampled from a manifold $\mathcal{M} \subseteq \mathbb{R}^m$, which lies in the ambient space $\mathbb{R}^m$, whose intrinsic dimension $d \ll m$. In this paper, we consider two related tasks: 1. How to approximate the spectral decomposition of a super-kernel? 2. How to perform out-of-sample extension of vector fields?

A. The Approximation of the Spectral Decomposition of Super-Kernel

The main goal of this paper is to efficiently approximate the embedded tensors of the super-kernel G (definition II.1) without computing the entire spectral decomposition of this large matrix. In other words, let G be the super-kernel that is constructed on the dataset M according to [10]. We aim to find two matrices $E \in \mathbb{R}^{\eta_s d \times nd}$ and $\tilde{G} \in \mathbb{R}^{\eta d \times \eta d}$ such that

$$G \approx E^T \hat{G} E, \tag{III.1}$$

where $\hat{G}$ is a super-kernel that was constructed by utilizing $\eta \ll n$ representative samples from the dataset M. In order to find a solution to Eq. III.1, we will construct a dictionary of representatives and then use an out-of-sample extension method to extend the results, which are achieved for them, to the entire dataset. Furthermore, the presented methodology provides an out-of-sample extension method of the achieved patch-to-tensor embedding from [10] to new data points $x \notin M$ that are not in the original dataset.

Since the dataset M is finite then we can assume that the data points in it are sequentially ordered as $M = \{x_1, \ldots, x_n\}$ and define $X_t = \{x_1, \ldots, x_t\}$ to be the first t data points in M for every $s = 0, \ldots, n$ where $X_0 \triangleq \emptyset$ and $X_n \triangleq M$. Then, we use an iterative approach, where each iteration $t = 1, 2, \ldots, n$, considers the data point x_t and compares it to the set $X_{s-1} \subset M$ of the data points that were already considered in previous iterations. We also assume that previous iterations already found a dictionary D_{s-1} of η_{s-1} representatives that are sufficient to represent the embedding of X_{s-1}. Then, the new data point $x_s \in M \backslash X_{s-1}$ is considered and its embedded tensor is approximated based on an out-of-sample extension of the PTE of X_{s-1} (or rather the dictionary D_{s-1}). If the approximation is not sufficiently accurate, the dictionary is updated to contain the new data point. Finally, when all the data points were encountered, a dictionary subset D_n of η_n representatives is formed and the super-kernel G can be approximated by this dictionary set instead of the whole dataset M. Given matrices E_s and $\hat{G}_s$, which approximate G as described in Eq. III.1, an efficient spectral decomposition can be formulated that approximates the spectral decomposition of G.

B. Out-of-Sample Extension for Vector Fields

The patch-to-tensor embedding in [10] is based on the spectral analysis of the super-kernel G. In our case, we want to use a dictionary (i.e., a set of representatives) to approximate this spectral decomposition and extend it (using an out-of-sample extension) to the entire dataset. This extension method can also be utilized to extend this decomposition either from the dictionary or from the dataset to new data points. According to [13], the super-kernel G can be regarded as an operator on tangent vector fields of the manifold $\mathcal{M}$ restricted to a dataset M. Therefore, the spectral decomposition of G consists of eigen vector fields which span the range of G. Hence, an out-of-sample extension of the eigen vector fields is equivalent to the out-of-sample extension of vector fields in the range of G.

Out-of-sample extension of vector fields assumes an a priori knowledge of a set of points M and a corresponding vector field where each vector lies on the respective local tangent space. Consider a tangent vector field $\vec{v} : M \to \mathbb{R}^d$ such that $\vec{v}(x) \in T_x(\mathcal{M})$ for all $x \in M$. Then, the given data points are used to construct the super-kernel G. Since $\bar{p}$ is positive-definite, then G is also positive-definite[1], thus it is invertible and its range consists of all these vector fields.

The out-of-sample extension of a new data point under the PTE settings aims to find the new corresponding vector in the local tangent space of the new point. The extension coefficients $\vec{u}$ are designed to minimize $\|G\vec{u} - \vec{v}\|_2$ over the given set of training points. These coefficients, which minimize the l_2 norm, are computed by using the inverse of G such that

$$\vec{u} = G^{-1}\vec{v}. \tag{III.2}$$

The coefficient $\vec{u}$ can be interpreted as a vector field $\vec{u} : M \to \mathbb{R}^d$ over the set of training points or, equivalently,

$$\vec{v}(x) = \sum_{y \in M} G_{(x,y)}\vec{u}(y),\ x \in M, \tag{III.3}$$

where $\vec{u}(y)$, $y \in M$, are considered as the coefficients of the vector field $\vec{v}$ according to the super-kernel G. Consider a new data point $x' \in \mathcal{M} \backslash M$ with the matrix $O_{x'}$ whose columns $o^1_{x'}, \ldots, o^d_{x'}$ form an orthonormal basis for the tangent space $T_{x'}(\mathcal{M})$. We can extend the vector field to a new data point x' by setting the value $\vec{v}(x')$ to be

$$\vec{v}(x') \triangleq \sum_{y \in M} \tilde{G}_{(x',y)}\vec{u}(y), \tag{III.4}$$

where $\tilde{G}_{(x',y)} = \bar{p}(x', y)O^T_{x'}O_y$, $y \in M$, are the non-scalar affinity blocks between the new point and the data points in the dataset. The extension in Eq. III.4 is consistent with the values $\vec{v}(x)$, $x \in M$, in Eq. III.3.

While the new affinity blocks in Eq. III.4 are not known in advance as part of the super-kernel, they are easily computed for any new point. This approximation only considers values of the vector field $\vec{u}$ at the data points in M, which can be computed in advance by using the pseudo inverse of the super-kernel G. This phase is not complicated, but it is beyond the scope of this paper since it is not crucial for the presented dictionary construction. Therefore, this provides a feasible out-of-sample extension of a vector field, which is similar to the methods shown in [3], [5] for the scalar case.

The extension in Eq. III.4 can be interpreted geometrically by separately considering projections and the scalar weights in the affinity blocks of the super-kernel. First, the extension projects the coefficient vector field $\vec{u}$ from the manifold M to the tangent space $T_{x'}(\mathcal{M})$ of the new data point x'. This projection expresses the coefficient vectors in local terms of the manifold around x'. Then, the value of the vector field $\vec{v}$ at x' is computed by using a weighted sum of the projected coefficient vectors on the tangent space $T_{x'}(\mathcal{M})$.

[1]See Theorem 3.1 in [10], where the weak inequalities in Eqs. 3.1 and 3.2 in the proof are replaced by strict inequalities.

IV. Patch-Based Dictionary

According to Lemma 3.3 in [10], the sum in Eq. III.3 can be rephrased in terms of the embedded tensors to be

$$\vec{v}(x) = \sum_{y \in M} \mathcal{T}_x^T \mathcal{T}_y \vec{u}(y),\ x \in M. \tag{IV.1}$$

However, due to linear dependencies between the embedded tensors, this sum may contain redundant elements. Indeed, if $\mathcal{T}_z = \sum_{z \neq y \in M} c_y^z \mathcal{T}_y$ for some scalar coefficients $c_y^z \in \mathbb{R}$, $z \neq y \in M$, then Eq. IV.1 becomes

$$\vec{v}(x) = \sum_{z \neq y \in M} \mathcal{T}_x^T \mathcal{T}_y (\vec{u}(y) + c_y^z \vec{u}(z)),\ x \in M. \tag{IV.2}$$

This enables us to eliminate the redundant tensors. By taking an iterative approach, we obtain a small subset of tensors, which is a set of linearly independent tensors that are sufficient for computing Eqs. III.3 and III.4.

Similarly, we can use matrix coefficients instead of scalar ones to incorporate reacher relations between tensors. Therefore, $\mathcal{T}_z$ is tensorialy dependent in $\{\mathcal{T}_y\}_{z \neq y \in M}$ if it satisfies

$$\mathcal{T}_z = \sum_{z \neq y \in M} \mathcal{T}_y C_y^z, \tag{IV.3}$$

for some matrix coefficients $C_y^z \in \mathbb{R}^{d \times d}$, $z \neq y \in M$. The defined dependency expresses more redundancies than the standard linear dependency. As a result, we obtain a sparser set of tensorialy independent tensors that enables us to efficiently compute Eqs. III.3 and III.4. This set of representative tensors constitutes a dictionary that compactly represents the embedded tensor space.

A. Dictionary Construction

We use an iterative approach to construct the described dictionary by sequential scann of the data points in M. At the first iteration, we define the scanned set $X_1 = \{x_1\}$ 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, the dictionary D_{s-1} that represents X_{s-1}. The dictionary D_{s-1} is in fact 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 $X_s = X_{s-1} \cup \{x_s\}$. 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 do this, a dependency criterion has to be imposed. If this criterion is satisfied, then the the dictionary remains the same $D_s = D_{s-1}$. Otherwise, it is updated to include the new data point $D_s = D_{s-1} \cup \{x_s\}$.

We use a dependency criterion similar to the approximated linear dependency (ALD) criterion used in KRLS [7]. The ALD measures the distance between vector candidates and the span by the dictionary vectors, to determine if the dictionary should be updated. In our case, we want to approximate the tensorial dependency (see Eq. IV.3) of the examined tensor $\mathcal{T}_{x_s}$ on 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

$$\delta_s \triangleq \min_{C_1, \ldots, C_{\eta_{s-1}}} \left\| \sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{y_j} C_j - \mathcal{T}_{x_s} \right\|_F^2, \qquad C_1, \ldots, C_{\eta_{s-1}} \in \mathbb{R}^{d \times d}, \tag{IV.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and $C_1, \ldots, C_{\eta_{s-1}}$ are matrix coefficients in Eq. IV.3. Then, 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 achieve the minimum in Eq. IV.4. Otherwise, the dictionary has to be updated by adding x_s to it.

Lemma IV.1. *Let $\hat{G}_{s-1} \in \mathbb{R}^{d\eta_{s-1} \times d\eta_{s-1}}$ be the 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 j-th $d \times d$ block is $G_{(y_j,x_s)}$, $j = 1, \ldots, \eta_{s-1}$. Then, the optimal matrix coefficients (from Eq. IV.4) are $\{C_j^{(s)}\}_{j=1,\ldots,\eta_{s-1}}$, where $C_j^{(s)}$ is the j-th $d \times d$ block of the $\eta_{s-1} \times 1$ $d \times d$ blocks matrix $\hat{G}_{s-1}^{-1} H_s$. The corresponding error δ_s in Eq. IV.4 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. IV.4 can be rephrased as

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

Algebraic simplifications yield

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

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 [10]. We perform these substitutions and get

$$\delta_s = \min_A \mathrm{tr} \left[A^T \hat{G}_{s-1} A - H_s^T A - A^T H_s + G_{(x_s,x_s)} \right], \tag{IV.5}$$

where $A \in \mathbb{R}^{d\eta_{s-1} \times d}$ is a $\eta_{s-1} \times 1$ block matrix whose j-th $d \times d$ block is C_j, $j = 1, \ldots, \eta_{s-1}$. Solving the optimization (i.e., minimization) in Eq. IV.5 yields the solution

$$A_s = \hat{G}_{s-1}^{-1} H_s, \tag{IV.6}$$

which proves the lemma. ■

Lemma IV.1, which provides an expression for the dictionary-based approximation, is expressed in super-kernel terms. Essentially, this eliminates the need for 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 just approximate its tensor. The threshold μ is anyway given in advance as a meta-parameter and δ_t can be computed by using the expression in Lemma IV.1, which does not depend on the embedded tensors. Therefore, the dictionary construction process only requires knowledge of the super-kernel blocks that are required to compute this expression at 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.

B. The Dictionary Construction Algorithm

In this section, we specify the operations that take place by each iteration t of the construction algorithm. Assume that E_s is a $\eta_s \times d$ by $s \times d$ block matrix at iteration s whose (i, j) entry (block) is the optimal coefficient matrix $C_i^{(j)}$, $1 \leq i \leq \eta_s$, $1 \leq j \leq s$. The coefficient matrix $C_i^{(j)}$ has two solutions that depend on the ATD criterion calculated at iteration s. First, Lemma IV.1 is applied to compute δ_s. Then, two possible scenarios are considered:

1. $\delta_s \leq \mu$, hence $\mathcal{T}_{x_s}$ is 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 coefficient 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 IV.1) to the coefficient matrix from the previous $s - 1$ iteration.

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

$$\hat{G}_s = \begin{bmatrix} \hat{G}_{s-1} & H_s \\ H_s^T & G_{(x_s,x_s)} \end{bmatrix}. \tag{IV.7}$$

The computation of the ATD conditions during these iterations requires to get $\hat{G}_s^{-1}$ of the dictionary based super-kernels. The block matrix inversion is utilized to derive

$$\hat{G}_s^{-1} = \begin{bmatrix} \hat{G}_{s-1}^{-1} + A^t \Delta_s^{-1} (A^t)^T & -A^t \Delta_s^{-1} \\ -\Delta_s^{-1} (A^t)^T & \Delta_s^{-1} \end{bmatrix}, \tag{IV.8}$$

where $\Delta_s = G_{(x_s,x_s)} - H_s^T \hat{G}_{s-1}^{-1} H_s$ is computed through 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 this set. This spectral decomposition is approximated by using the extension matrix E_s and the dictionary related super-kernel $\hat{G}_s$ (Eq. IV.7).

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 $R \in \mathbb{R}^{d\eta_s \times d\eta_s}$ 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 $E_s^T \hat{G}_s E_s = (Q_s \tilde{U}_s) \Sigma_s Q_s \tilde{U}_s^T$. Thus, the SVD of G_s i approximated by SVD such that $G_s \approx U_s \Sigma_s U_s^T$, where

$$U_s = Q_s \tilde{U}_s. \tag{IV.9}$$

Algorithm IV.1: Patch-to-Tensor Dictionary Construction and Embedding Approximation (PTEA)

Input: Data points: $x_1, ..., x_n \in R^m$.

Parameters: Patch size ρ, max approximation tolerance μ, ℓ, ν, G, D_s, n

1: Initialize: $\hat{G}_1 = G_{(x_1,x_1)}$ (block, definition II.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 $H_s = \left[G_{(x_1,x_s)}, ..., G_{(X_{\eta_s},x_s)}\right]$
- 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. IV.7
    - Update $\hat{G}_s^{-1}$ according to Eq. IV.8
    - Set $\eta_s = \eta_{s-1} + 1$

3: Approximate U_n according Eq. IV.9.

4: Use U_n to compute the approximated spectral map $\Phi(o_x^j)$, $x \in M$, $j = 1, \ldots, d$, according to Eq. **??** (considering the first ℓ eigenvalues and 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. II.5.

The main computational cost of this approach lies in the computation of the QR decomposition that is estimated to be $O(d^3\eta_s^2(s-\eta_s/3))$. The computational cost is substantially smaller in comparison with the full SVD computation cost that is $O(d^3s^3)$.

The dictionary construction followed by the patch-to-tensor embedding approximation process is described in Algorithm IV.1.

C. The Super-Kernel Approximation Error Bound

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

$$G_s \approx E_s^T \hat{G}_s E_s, \tag{IV.10}$$

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 quantification of Eq. IV.10, let $\Psi_s \triangleq [\mathcal{T}_{x_1}, ..., \mathcal{T}_{x_s}]$ be the $\ell \times sd$ matrix that aggregates all the embedded tensors up to step s, let $\hat{\Psi}_s \triangleq [\mathcal{T}_{y_1}, ..., \mathcal{T}_{y_{\eta_s}}]$ be the $\ell \times \eta_s d$ matrix that aggregates all embedded tensors of the dictionary members D_s and let $\Psi_s^{\text{res}} \triangleq \Psi_s - \hat{\Psi}_s E_s = [\psi_1^{\text{res}}, \ldots, \psi_s^{\text{res}}]$, where $\psi_s^{\text{res}} \triangleq \mathcal{T}_{x_s} - \sum_{j=1}^{\eta_{s-1}} \mathcal{T}_{y_j} C_j^{(s)}$. Then, due to Eq. IV.4 and the ATD condition, $\|\Psi_s^{\text{res}}\|_F^2 \leq s\mu$. From the definition 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 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 in step s, $R_s = G_s - E_s \hat{G}_s E_s^T = (\Psi_s^{\text{res}})^T \Psi_s^{\text{res}}$, satisfies $\|R_s\|_F^2 \leq s\nu$. In particular, for $s = n$, $\|R_n\|_F^2 \leq n\nu$.

D. Out-of-Sample Extension for Vector Fields

The presented dictionary can be utilized to estimate a tangential vector field by using a recursive algorithm similar to the well known Kernel Recursive Least Squares (KRLS) in [7]. In a supervised learning scheme, the predictor in Eq. III.4 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

$$J(\vec{w}) = \sum_{i=1}^{s} \|\hat{\nu}(x_i) - \nu(x_i)\|_2^2 = \sum_{i=1}^{s} \left\| \sum_{j=1}^{t} G_{(x_i,x_j)} \vec{w}_j - \nu(x_i) \right\|_2^2 = \|G\vec{w} - \vec{\nu}\|_2^2, \tag{IV.11}$$

where $\vec{w}$ is the predictor weights vector and $\vec{\nu}$ is the concatenation of all the given training values of the vector field[2]. The Recursive Least Squares solution, which minimizes $J(\vec{w})$, is given by $\vec{w}_o = G^{-1}\vec{\nu}$. 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 cost and memory requirements. Furthermore, the length of the predictor weight vector $\vec{w}_o$ depends on the number of training samples. Hence, redundant samples, which generate linearly-dependent rows in the super-kernel, will cause over-fitting by the predictor. One possible remedy for these problems is to utilize the sparsification procedure from Section IV-B in order to design the predictor. The optimizer $\vec{w}_o$ can be formulated by introducing the dictionary estimated super-kernel as $J(\vec{w}) = \|G\vec{w} - \vec{\nu}\|_2^2 \approx \|E_s^T \hat{G}_s E_s w - \vec{\nu}\|_2^2$ by Eq. IV.10. Let $\alpha = E_s w$, then the predictor is reformulated as $\hat{\vec{\nu}}(x_i) = \sum_{j=1}^{s} C_j^{(s)} G_{(x_j,x_i)} \alpha_j$, and the corresponding predictor's l_2 error is given by $J(\alpha) = \|E_s^T \hat{G}_s E_s w - f\| = \|E_s^T \hat{G}_s \alpha - \vec{\nu}\|$, which can be minimized by

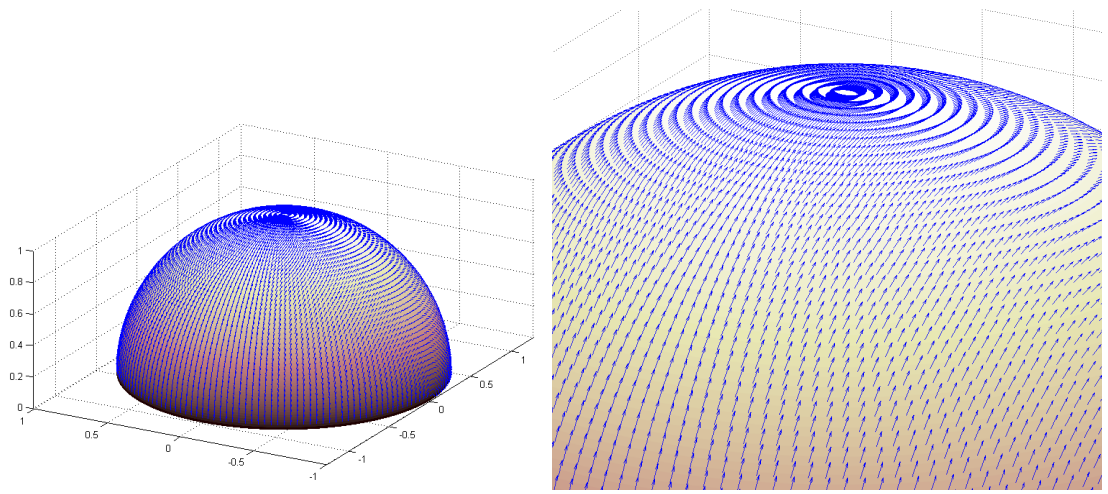
$$\alpha_o = (E_s^T \hat{G}_s)^\dagger f = \hat{G}_s^{-1} (E_s E_s^T)^{-1} f. \tag{IV.12}$$

Now, 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 corresponding kernel plus vectors sampled from the given vector field. In this case, the extenstion/predictor coefficients can be directly designed according to Eq. III.2.

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

V. Illustration of Data Analysis via Dictionary Based Patch-to-Tensor Embedding

The PTE methodology in [10] provides a general framework that can be utilized to a wide spectrum of data analysis tasks such as clustering, classification, anomaly detection and related manifold learning tasks. In this section, we demonstrate the utilization of the proposed dictionary construction based PTE to two interesting applications: I. Vector field extension. II. Image segmentation.



(a) (b)

Fig. V.1. The generated vector field. (a) Full view. (b) Zoom.

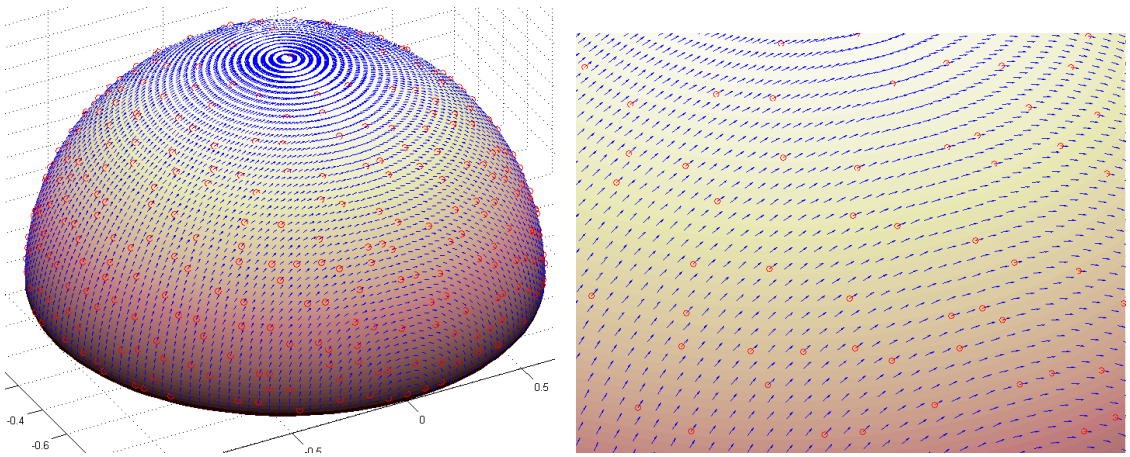
Example I: Vector field extension: In this example, we utilize a synthetic vector field F for the demonstration. The synthetic vector field consists of $|M| = N = 8050$ points sampled from a two dimensional half sphere $\mathcal{M}$ that is immersed in $\mathbb{R}^3$ by $F : M \to \mathbb{R}^3$ such that for any $x \in M$ we have $F(x) \in T_x(\mathcal{M})$. For each data point, we generate a vector that lies on the tangent plane of the sphere at the corresponding data point. Figure V.1 illustrates the described vector field by two views.

The dictionary in this example is constructed using Algorithm IV.1 with the meta-parameters $\varepsilon = 0.0218$ and $\mu = 0.0005$. The resulting dictionary contains $\eta_N = 414$ dictionary members. Figure V.2 presents the dictionary members, which provides an almost uniform sampling of the sphere.

Under the above settings, the vector $O_x^T F(x)$ is a coordinate vector of $F(x)$ for 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 tangent space O_x is given by the SVD of $[S_1, S_2]^T$. Furthermore, we choose the vectors of the vector field to be the corresponding vector $F(x) = (1, 0, -x_1\sqrt{1 - x_1^2 - x_2^2})$. Equation III.2 provides the solution to the extension problem given the dictionary members and the corresponding super-kernel. For each point y that is not in the dictionary, we compute Eq. III.4 to find the extended vector field. The resulting vector field is compared to the original vector field in Fig. V.3.

In order to evaluate the performance of the extension, we compared both the length of the resulting vector and its direction. Figure V.4 describes the cumulative distribution function of these squared errors. The cumulative distribution functions in Fig. V.4 suggest that, compared to 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 $2 \cdot 10^{-2}$ radians.

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 such as object recognition, image compression, image editing, image retrieval, to name some.



(a) (b)

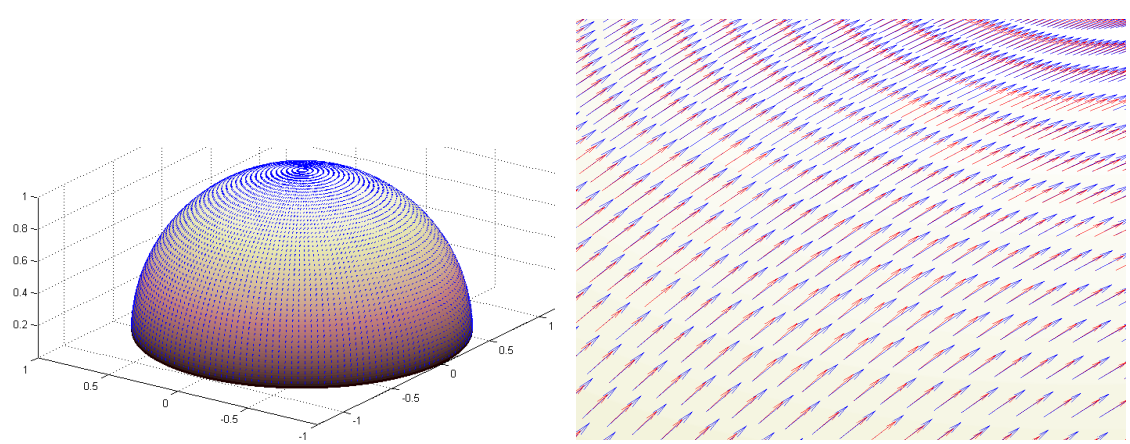
Fig. V.2. The generated vector field (arrow) and chosen dictionary members (circle). (a) Full view. (a) Zoom.

Under the PTE framework [10], the image is viewed as a super-kernel constructed to reflect the affinities between the pixels and the projection of the related tangent spaces. The PTE construction translates a given pixel related features into tensors in the embedded space. The image segmentation is achieved through tensors clustering into similar groups in the embedding space.

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

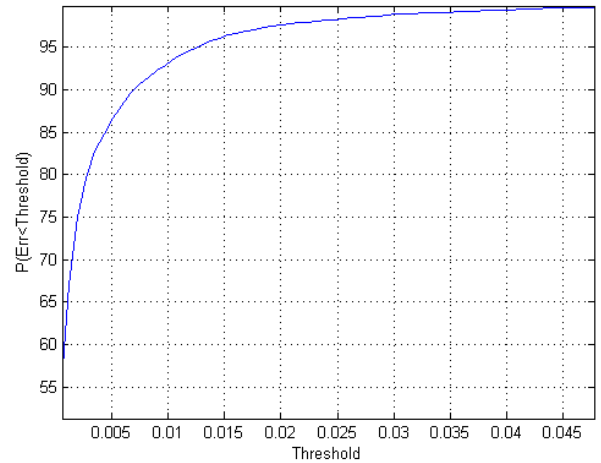
Algorithm IV.1 is used to construct an embedding of X in a tensor space. The first step in Algorithm IV.1 constructs local patches. Each generated patch captures the relevant neighborhood and considers both color and spatial similarities. Hence, a patch is more likely to include attributes related to spatially close pixels. It is important to note that the affinity kernel is computed according to Eq. II.2 where ε equals 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 was used to cluster the tensors into similar groups. The final clustering results from the embedded tensors as a function of the diffusion transition time t are presented in Figs. V.5-V.6.

These figures present the outputs from the application of the PTE segmentation that are based on the dictionary construction. In each figure, (a) is the original image. The images' sizes are given in Table V.1.

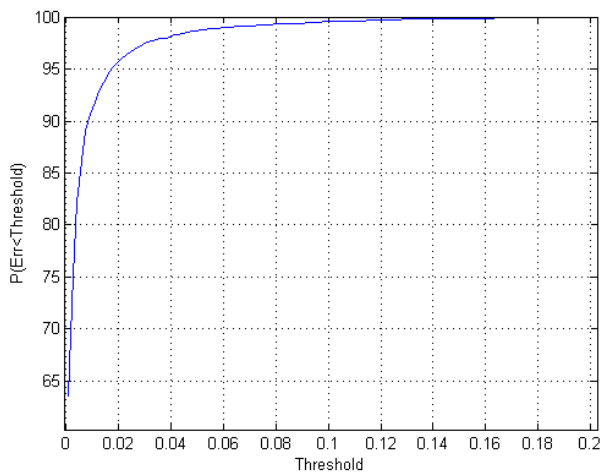


(a) (b)

Fig. V.3. The extended vector field (red) and the given vector field (blue). (a) Full view. (b) Zoom.



(a)



(b)

Fig. V.4. The cumulative distribution function of MSE of (a) Estimated vector length and (b) Estimated vector direction.

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 is significant for the 'Hand' image. For example, we can see the first three images in Fig. V.5 that correspond to $t = 1$ and $t = 3$, respectively. They provide poor segmentation results. As t increases, the segmentation becomes more homogenous thus the main structures in the original image can be separated. For example, see $t = 4$ 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 distance between similar tensors

Image	Size	d	SVD Cost - Full G	SVD Cost - Approx. G	Dict. Size
Hand	104x128	2	$O\left(26624^3\right)$	$O\left(26624 \times 16\right)$	2
Sport	40x77	2	$O\left(6160^3\right)$	$O\left(6160 \times 16\right)$	2

TABLE V.1

COMPARING THE ESTIMATED PERFORMANCE COST OF THE DICTIONARY, WHERE **d** IS THE ESTIMATED INTRINSIC DIMENSION, **SVD Cost - Full** G IS THE COMPUTATIONAL COST ESTIMATE FOR A FULL KERNEL DECOMPOSITION, **SVD Cost - Approx.** G IS THE COMPUTATIONAL COST ESTIMATE FOR THE DECOMPOSITION OF THE APPROXIMATED KERNEL ACCORDING TO EQ. IV.9 AND **Dict. Size** IS THE NUMBER OF DICTIONARY MEMBERS.

is reduced while he distances between dissimilar tensors increases.

The dictionary construction enables us to practically utilize the PTE for segmentation of reasonably sized images. The computational costs were significantly reduced from the application of the SVD decomposition that utilizes the full super-kernel. These performances were achieved by using the dictionary-based SVD/QR of the extension coefficient matrix E, which is efficiently computed using the methods described in the paper.

VI. Conclusions

The proposed construction extends the dictionary construction in [7] to use the LPD super-kernel from [10], [13]. This is done by an efficient dictionary based construction that assumes the data is sampled from an underlying manifold and utilizes non-scalar relations and similarities between patches of the manifold instead of utilizing individual data points. The utilized dictionary contains these patches of the underlying manifold, which are represented by the embedded tensors from [10], instead of individual data points. Therefore, it encompasses multidimensional similarities between local areas of the data. It alleviates the computational costs of the spectral analysis of such data (e.g., the Patch-to-Tensor Embedding presented in [10]).

Acknowledgements: This research was supported by the Israel Science Foundation (Grant No. 1041/10), the Israel Ministry of Science & Technology (Grant No. 3-9096). The third author was supported by the Eshkol Fellowship from the Israel Ministry of Science & Technology.

References

[1] C.T.H. Baker. *The Numerical Treatment of Integral Equations*. Oxford: Clarendon Press, 1977.
[2] C. Ballester, M. Bertalmio, G. Sapiro, and J. Verdera. Filling-in by joint interpolation of vector fields and gray levels. *IEEE Trans. Image Processing*, 10:1200–1211, 2001.
[3] A. Bermanis, A. Averbuch, and R.R. Coifman. Multiscale data sampling and function extension. *Applied and Computational Harmonic Analysis*, 2012. DOI:10.1016/j.acha.2012.03.002.
[4] R.R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, 2006.
[5] R.R. Coifman and S. Lafon. Geometric harmonics: A novel tool for multiscale out-of-sample extension of empirical functions. *Applied and Computational Harmonic Analysis*, 21(1):31–52, 2006.
[6] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman and Hall, London, UK, 1994.
[7] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *Signal Processing, IEEE Transactions on*, 52(8):2275 – 2285, aug. 2004.

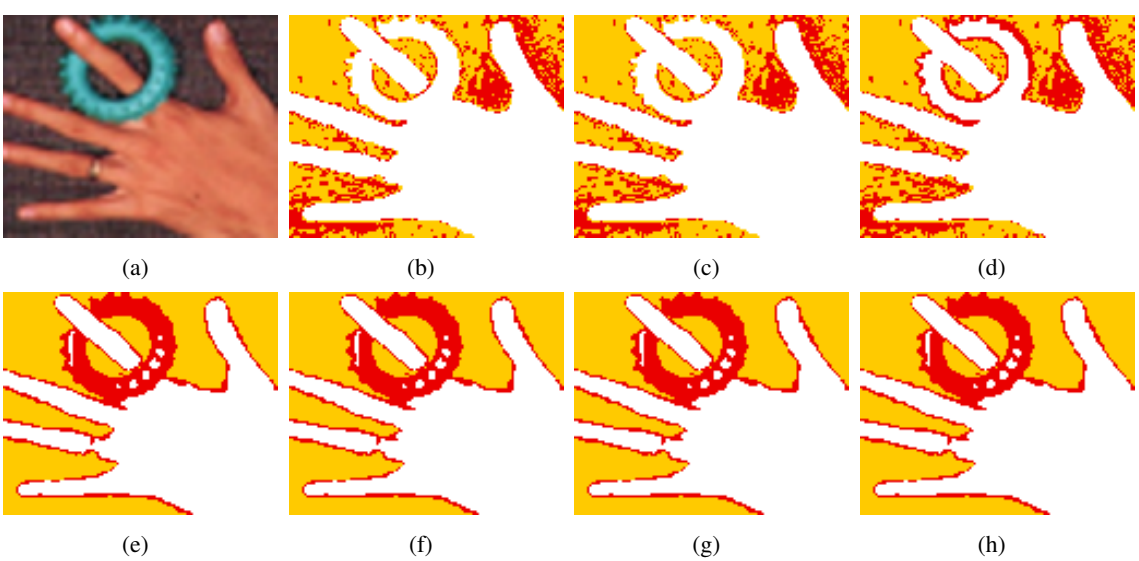
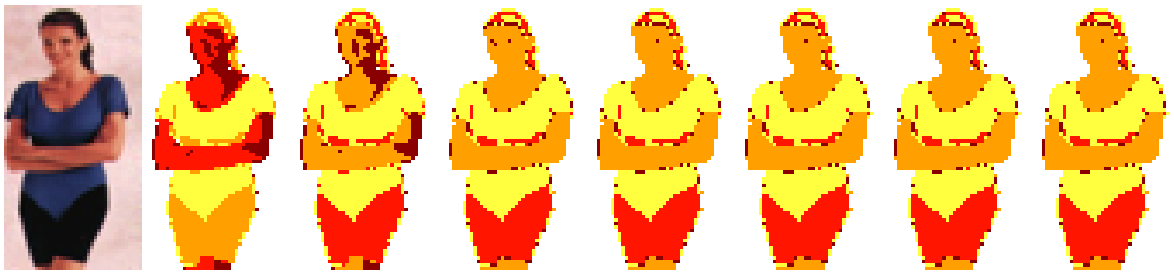


Fig. V.5. Segmentation results from the application of the PTEA algorithm to 'Hand' with $l = 10$ and $d = 10$. (a) Original image. (b) Segmentation with $t = 1$. (c) Segmentation with $t = 2$. (d) Segmentation with $t = 3$. (e) Segmentation with $t = 4$. (f) Segmentation with $t = 5$. (g) Segmentation with $t = 6$. (h) Segmentation with $t = 7$.



(a) (b) (c) (d) (e) (f) (g) (h)

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

[8] E. J. Fuselier and G.B. Wright. Stability and error estimates for vector field interpolation and decomposition on the sphere with rbfs. *SIAM J. Numer. Anal.*, 47(5):3213–3239, October 2009.

[9] J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

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

[11] A. Singer and H.-t. Wu. Orientability and diffusion maps. *Applied and Computational Harmonic Analysis*, 31(1):44–58, 2011.

[12] A. Singer and H.-t. Wu. Vector diffusion maps and the connection laplacian. *Communications on Pure and Applied Mathematics*, 65(8):1067–1144, 2012.

[13] G. Wolf and A. Averbuch. Linear-projection diffusion on smooth Euclidean submanifolds. *Applied and Computational Harmonic Analysis*, 2012. DOI:10.1016/j.acha.2012.03.003.