Manifold Kernel Sparse Representation of Symmetric Positive-Definite Matrices and Its Applications

Yuwei Wu, Yunde Jia, *Member, IEEE*, Peihua Li, Jian Zhang, *Senior Member, IEEE*, and Junsong Yuan, *Senior Member, IEEE*

***Abstract*— The symmetric positive-definite (SPD) matrix, as a connected Riemannian manifold, has become increasingly popular for encoding image information. Most existing sparse models are still primarily developed in the Euclidean space. They do not consider the non-linear geometrical structure of the data space, and thus are not directly applicable to the Riemannian manifold. In this paper, we propose a novel sparse representation method of SPD matrices in the data-dependent manifold kernel space. The graph Laplacian is incorporated into the kernel space to better reflect the underlying geometry of SPD matrices. Under the proposed framework, we design two different positive definite kernel functions that can be readily transformed to the corresponding manifold kernels. The sparse representation obtained has more discriminating power. Extensive experimental results demonstrate good performance of manifold kernel sparse codes in image classification, face recognition, and visual tracking.**

***Index Terms*— Kernel sparse coding, Riemannian manifold, region covariance descriptor, symmetric positive definite matrices, visual tracking, image classification, face recognition.**

I. Introduction

SPARSE representation (SR) has been an important subject in signal processing and computer vision communities with a wide range of applications including visual tracking [1]–[3], face recognition [4], [5], and image classification [6], [7]. Given a set of data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$, the sparse model attempts to find a dictionary $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N\}$, where $\mathbf{d}_i$ is the so-called *base* or *atom*, such that each $\mathbf{x}_i$ can be linearly reconstructed by a relatively small subset of atoms from $\mathcal{D}$, meanwhile keeping the reconstruction error as small as possible. The underlying linear process significantly depends on the assumption that the data points and the atoms lie on a vector space $\mathbb{R}^d$. In many applications, however, data points actually belong to known Riemannian manifolds such as the space of symmetric positive-definite (SPD) matrices [8], [9], Stiefel and Grassmann manifolds [10], [11]. Most existing sparse models in $\mathbb{R}^d$ fail to consider the non-linear geometrical structure of the manifold space $\mathcal{M}$, and hence are not directly applicable to the Riemannian manifold. In this paper, we tackle the problem of the SR in the space of $d \times d$ SPD matrices, denoted by Sym_d^+. Unlike the Euclidean space, the space of Sym_d^+ lacks a global linear structure. To formulate the sparse representation on Sym_d^+, one natural and crucial question arises: how to allow the SPD matrix to be reconstructed linearly by the atoms on Sym_d^+ using an appropriate metric which can measure the intrinsic distance between two SPD matrices?

Intuitively, a direct approach is to approximate a SPD matrix by the linear combination of other SPD matrices. In general, employing the linear combination of atomic matrices to represent a SPD matrix largely depends on an appropriate metric to measure the reconstruction error, *e.g.,* Logdet divergence and Frobenius norm. Sivalingam *et al.* [12] proposed a tensor sparse coding method, in which the Logdet divergence is used to measure the reconstruction error. The sparse decomposition of a SPD matrix is then formulated as a MAXDET optimization problem that can be solved by the interior-point (IP) algorithm. Sivalingam *et al.* [13] further introduced a dictionary learning method using the Logdet divergence. However, the solutions of the above two approaches are computationally expensive. Sra and Cherian [14] adopted the Frobenius norm as an error metric to learn a generalized dictionary of rank-1 atoms to sparsely represent a SPD matrix. However, several studies [15], [16] show that the Frobenius norm, which basically vectorizes SPD matrices and measures the norm between two matrices, is not a good metric, since it discards the manifold geometry.

In order to apply existing vector-based sparsity modeling approaches, an alternative scheme is to embed manifold data points into a vector space $\mathbb{R}^d$. One commonly used vector space is the tangent space at the mean of the data points in $\mathcal{M}$. The logarithmic and exponential maps are iteratively

Manuscript received September 2, 2014; revised February 12, 2015 and May 9, 2015; accepted June 22, 2015. Date of publication July 1, 2015; date of current version July 23, 2015. This work was supported in part by the National Natural Science Foundation of China under Grant 61375044, in part by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20121101110035, and in part by the Ministry of Education Tier-1, Singapore, under Grant M4011272.040. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Rebecca Willett. *(Corresponding author: Yunde Jia.)*

Y. Wu and Y. Jia are with the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China (e-mail: wuyuwei@bit.edu.cn; jiayunde@bit.edu.cn).

P. Li is with the School of Information and Communication Engineering, Dalian University of Technology, Dalian 116023, China (e-mail: peihuali@dlut.edu.cn).

J. Zhang is with the Faculty of Engineering and Information Technology, Advanced Analytics Institute, University of Technology at Sydney, Sydney, NSW 2007, Australia (e-mail: jian.zhang@uts.edu.au).

J. Yuan is with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore (e-mail: jsyuan@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2015.2451953

used to map the manifold data points to the tangent space, and vice-versa. Exploiting the Log-Euclidean mapping of SPD matrices, Zhang *et al.* [17] obtained the vectorized Log-Euclidean covariance features for sparse representation. Guo *et al.* [18] transformed the Riemannian manifold of SPD matrices into a vector space $\mathbb{R}^d$ under the matrix logarithm mapping. The log-covariance matrix is approximated by a sparse linear combination of the log-covariance matrices of training samples. Yuan *et al.* [19] also proposed to solve sparse representation for human action recognition by embedding manifolds into tangent spaces. Although log-Euclidean based approaches benefit from their simplicity, the iterative computation of the logarithmic and exponential maps results in a high computational cost. In addition, the tangent space preserves only the local structure of manifold data points, *i.e.*, the true geometry structure is not taken into account, which often results in sub-optimal performance.

To consider the local manifold structure of manifold data points, many attempts have been made to implicitly map these data into a high-dimensional Reproducing Kernel Hilbert Space (RKHS) by using a nonlinear map associated with a kernel function. Harandi *et al.* [20] tackled the problem of both SR and dictionary learning in Sym_d^+ by adopting the Stein kernel to map the SPD matrices to a RKHS. Following this line of work [20], Zhang *et al.* [21] proposed an online dictionary learning method on SPD manifolds using the Stein kernel. Nonetheless, the Stein divergence is only an approximation of the Riemannian metric as it is positive definite only for some values of the Gaussian bandwidth parameter. Barachant *et al.* [22] exploited a Riemannian-based kernel to model the SR of SPD matrices for brain-computer interface applications. Li *et al.* [23] also embedded Sym_d^+ into a RKHS and developed Log-E kernels for both SR and dictionary learning of SPD matrices based on the Log-Euclidean framework. Although Log-E kernels obtain satisfactory results in face recognition and image classification, their modeling does not explicitly reflect the geometrical structure of the data space.

The key issue of mapping SPD matrices into a RKHS while preserving the geometrical structure of the data is the construction of the kernel function. An essential criterion is that the kernel function should be positive definite. The Gaussian kernel is perhaps the most popular positive definite kernel on the $\mathbb{R}^d$. Both Jayasumana *et al.* [9] and Vemulapalli *et al.* [24] presented the Gaussian kernel based on the Log-Euclidean metric. In practice, however, the nonlinear structure captured by the data-independent kernels, *e.g.,* Gaussian kernel, may not be consistent with the intrinsic manifold structure.

In this paper, we construct a data-dependent manifold kernel function using the kernel deformation principle [25]. The SR on the space of SPD matrices can be performed by embedding the Sym_d^+ into a RKHS using the proposed manifold kernel, as shown in Fig. 1. Furthermore, the graph Laplacian as a smooth operator of manifold data points is incorporated into the kernel space to discover the manifold structure. Different positive definite kernel functions on the space of SPD matrices are introduced, which can be easily transformed to the corresponding manifold kernels to better

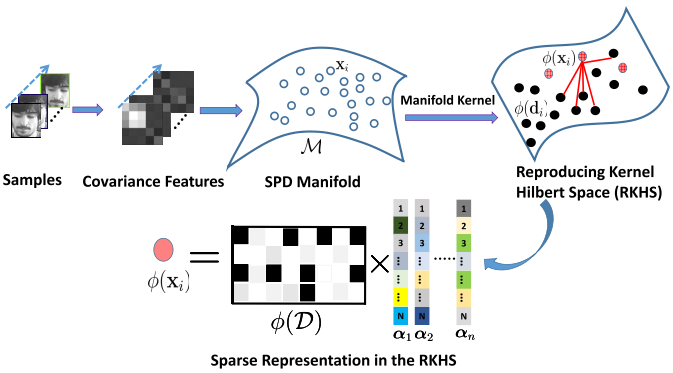


Fig. 1. Data points $\mathbf{x}_i$ on the manifold $\mathcal{M}$ of SPD matrices are mapped into RKHS using the data-dependent manifold kernel function. Since the RKHS is a linear space, $\phi(\mathbf{x}_i)$ can be naturally represented by a linear combination of atoms $\phi(\mathbf{d}_i)$.

characterize the underlying geometry structure of the manifold. These schemes have several advantages: (1) Since the RKHS is a complete vector space, the input data $\phi(\mathbf{x}_i)$ can be naturally approximated by using a sparse linear combination of *atoms* $\phi(\mathbf{d}_i)$ from the dictionary. (2) The high-dimensional RKHS typically yields a more discriminative representation which is potentially better suited for visual analysis.

The remainder of this paper is organized as follows. We discuss the preliminaries including Riemannian geometry on SPD matrices and kernel sparse representation in Sect. II. In Sect. III, we introduce the data-dependent manifold kernel on SPD matrices. Then we describe the details of the manifold kernel sparse representation on Sym_d^+, including its objective function and its implementation in Sect. IV. Experimental results are reported and analyzed in Sect. V and the conclusion is given in Sect. VI.

II. Preliminaries

A. Riemannian Geometry on SPD Matrices

SPD matrices usually emerge in the form of covariance features defined in ***Definition*** 1 [26]. The covariance matrix descriptor, as a special case of SPD matrices, captures feature correlations compactly in an object region, and therefore has been proven to be effective for pedestrian detection [27], face recognition [28], and texture classification [29] *etc.*

Definition 1: Given a region of interest R of an image, let $\boldsymbol{z}_i \in \mathbb{R}^d$, for $i = 1, 2, \cdots, N$, be feature vectors from R, then the covariance matrix descriptor $\boldsymbol{C}_R \in Sym_d^+$ is defined as

$$\boldsymbol{C}_R = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{z}_i - \mu_R)(\boldsymbol{z}_i - \mu_R)^\top, \tag{1}$$

where $\mu_R = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{z}_i$ is the mean vector, and N is the number of pixels in region R. The feature vector $\boldsymbol{z}_i$ may consist of the pixel coordinates, image gray level or color, image gradients, edge magnitude, edge orientation, filter responses, etc. For example, $\boldsymbol{z} = \left[x, y, I, |I_x|, |I_y|, \sqrt{I_x^2 + I_y^2}\right]^\top$.

In Sym_d^+, the SPD matrix lies on a connected Riemannian manifold. In this case, the geodesic distance induced by the Riemannian metric is a suitable choice for considering the manifold structure of SPD matrices. Two widely used distance measures in Sym_d^+ are the affine-invariant distance and the

Log-Euclidean distance [26]. Typically, the former requires eigenvalue computations, causing significant slowdowns for larger matrices. The latter is particularly simple to use and overcomes the computational limitations of the affine-invariant distance.

For any matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ in Sym_d^+, the logarithmic product $\boldsymbol{C}_1 \odot \boldsymbol{C}_2$ is defined as

$$\boldsymbol{C}_1 \odot \boldsymbol{C}_2 := \exp\big(\log(\boldsymbol{C}_1) + \log(\boldsymbol{C}_2)\big). \tag{2}$$

The logarithmic multiplication $\odot$ on Sym_d^+ is compatible with the structure of a smooth manifold: $(\boldsymbol{C}_1, \boldsymbol{C}_2) \longmapsto \boldsymbol{C}_1 \odot \boldsymbol{C}_2^{-1} \in \mathcal{C}^\infty$. Sym_d^+, therefore, is given a commutative Lie group structure $\mathcal{G}$ by $\odot$. The tangent space at the identity element in $\mathcal{G}$ forms a Lie algebra $\mathcal{H}$, a vector space. In a Lie algebra $\mathcal{H}$, the Riemannian manifold of SPD matrices can be mapped to the Euclidean space by matrix logarithm. Analogously, the results of the Euclidean space can be mapped back to the Riemannian space by the matrix exponential. Given a symmetric matrix $\boldsymbol{C} \in Sym_d^+$, $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$ is the eigen-decomposition of SPD matrix $\boldsymbol{C}$, where $\boldsymbol{U}$ is an orthonormal matrix and $\boldsymbol{\Sigma} = Diag\big(\lambda_1, \lambda_2, \cdots, \lambda_n\big)$ is a diagonal matrix composed of the eigenvalues. SPD matrix $\boldsymbol{C}$ has unique matrix logarithm $\log(\boldsymbol{C})$ and matrix exponential $\exp(\boldsymbol{C})$:

$$\begin{cases} \log(\boldsymbol{C}) = \boldsymbol{U} \cdot Diag\big(\log(\lambda_1), \log(\lambda_2), \cdots, \log(\lambda_d)\big) \cdot \boldsymbol{U}^\top \\ \exp(\boldsymbol{C}) = \boldsymbol{U} \cdot Diag\big(\exp(\lambda_1), \exp(\lambda_2), \cdots, \exp(\lambda_d)\big) \cdot \boldsymbol{U}^\top \end{cases} \tag{3}$$

The Log-Euclidean metric on the Lie group of SPD matrices corresponds to a Euclidean metric in the logarithmic domain. The distance between two matrices $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ is calculated by

$$d(\boldsymbol{C}_1, \boldsymbol{C}_2) = \| \log(\boldsymbol{C}_1) - \log(\boldsymbol{C}_2) \|_F, \tag{4}$$

where $\| \cdot \|_F$ denotes the matrix Frobenius norm induced by the Frobenius matrix inner product $\langle \cdot, \cdot \rangle$.

B. Kernel Sparse Representation

Let $\mathcal{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ be a data matrix with n d-dimensional features extracted from an image, $\mathcal{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N] \in \mathbb{R}^{d \times N}$ be a dictionary where each column represents an atom, and $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_n] \in \mathbb{R}^{N \times n}$ be the coding matrix. The goal of sparse representation is to learn a dictionary and corresponding sparse codes such that each input local feature $\mathbf{x}_i$ can be well approximated by the dictionary $\mathcal{D}$. The general formulation of the sparse representation is expressed as

$$\arg\min_{\mathcal{D}, \boldsymbol{\alpha}} \sum_{i=1}^{n} \|\mathbf{x}_i - \mathcal{D}\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1, \tag{5}$$

where $\|\mathbf{x}_i - \mathcal{D}\boldsymbol{\alpha}_i\|_2^2$ measures the approximation error, and $\|\boldsymbol{\alpha}_i\|_1$ enforces $\boldsymbol{\alpha}_i$ to have a small number of nonzero elements. Although the objective function in Eq. (5) is not convex in both variables, it is convex in either $\mathcal{D}$ or $\boldsymbol{\alpha}$. The ℓ_1 minimization problem can be solved efficiently [30].

Recently, Nguyen *et al.* [31], [32] suggested that each atom of the dictionary has a sparse representation over the feature space $\phi(\mathcal{X})$, which leads to a simple and flexible dictionary representation. Gao *et al.* [33] proposed a kernel version of sparse representation in the RKHS mapped by an implicit function ϕ. Mercer kernels are usually employed to carry out the mapping implicitly. The Mercer kernel is a function $\mathcal{K}(\cdot, \cdot)$ which can generate a kernel matrix $\boldsymbol{K}_{ij} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$ using pairwise inner products between mapped samples for all the input data points. The data points $\mathcal{X}$ and dictionary $\mathcal{D}$ are transformed to the corresponding feature space:

$$\begin{aligned} \mathcal{X} &= \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \xrightarrow{\phi} \{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \cdots, \phi(\mathbf{x}_n)\} \\ \mathcal{D} &= \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N\} \xrightarrow{\phi} \{\phi(\mathbf{d}_1), \phi(\mathbf{d}_2), \cdots, \phi(\mathbf{d}_N)\}. \end{aligned} \tag{6}$$

Then we substitute the mapped features and dictionary to the kernelized formulation of sparse representation:

$$\arg\min_{\mathcal{D}, \boldsymbol{\alpha}} \sum_{i=1}^{n} \|\phi(\mathbf{x}_i) - \phi(\mathcal{D})\boldsymbol{\alpha}_i\|_2^2 + \lambda \|\boldsymbol{\alpha}_i\|_1. \tag{7}$$

While the kernel sparse representation has been extensively developed, most algorithms [33]–[35] are still primarily developed for data points lying on the vector space. In this work, we focus on the sparse representation of SPD matrices, Sym_d^+. Motivated by the nonlinear generalization performance of kernel methods for sparse representation [20], [23], [33], we embed Sym_d^+ into the RKHS using the data-dependent manifold kernel, which better reflects the underlying geometry of the data.

III. Data-Dependent Manifold Kernel on Sym_d^+

A. Kernel Deformation

The choice of the kernel function is an essential issue of mapping SPD matrices into the RKHS while preserving the geometrical structure of the data. In this work, we adopt a kernel deformation principle [25] to learn a data-dependent kernel function. The goal of kernel deformation is to derive a data-dependent kernel by incorporating the estimated geometry prior of the underlying marginal distribution from the input matrix $\mathcal{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$. Since the resulting kernel considers the data distribution, it may achieve better performance than the original input kernel.

Let $\mathcal{H}$ denote the original RKHS reproduced by the kernel function $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \big\langle \mathcal{K}(\mathbf{x}_i, \cdot), \mathcal{K}(\cdot, \mathbf{x}_j) \big\rangle_{\mathcal{H}}$. $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$ can be built using the Riesz representation theorem for the decision function $f(\mathbf{x})$, *i.e.,* $f(\mathbf{x}) = \big\langle f, \mathcal{K}(\mathbf{x}, \cdot) \big\rangle_{\mathcal{H}}$. To take the geometry prior of the data distribution as a similarity measure in the deformed kernel, we define a linear space $\mathcal{V}$ with a positive semidefinite inner product, and let $\mathcal{S} : \mathcal{H} \longrightarrow \mathcal{V}$ be a bounded linear operator. The deformed RKHS $\widetilde{\mathcal{H}}$ can be defined by the modified inner product [25]

$$\langle f, g \rangle_{\widetilde{\mathcal{H}}} = \langle f, g \rangle_{\mathcal{H}} + \langle Sf, Sg \rangle_{\mathcal{V}}.$$

Given samples $\mathcal{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, let $\mathcal{S} : \mathcal{H} \longrightarrow \mathbb{R}^n$ be the decision map $S(f) = \big(f(\mathbf{x_1}), \cdots, f(\mathbf{x_n})\big)$. Denote $\boldsymbol{f} = \big(f(\mathbf{x_1}), \cdots, f(\mathbf{x_n})\big)$, $\boldsymbol{g} = \big(g(\mathbf{x_1}), \cdots, g(\mathbf{x_n})\big)$, and $\boldsymbol{f}, \boldsymbol{g} \in \mathcal{V}$, we have

$$\langle Sf, Sg \rangle_{\mathcal{V}} = \langle \boldsymbol{f}, \boldsymbol{g} \rangle = \boldsymbol{f}^\top M \boldsymbol{g}, \tag{8}$$

where M is a symmetric positive semi-definite matrix that captures the geometry relationship between all the data points. Thus we have the following relationship between the two Hilbert spaces $\mathcal{H}$ and $\widetilde{\mathcal{H}}$:

$$\langle f, g\rangle_{\widetilde{\mathcal{H}}} = \langle f, g\rangle_{\mathcal{H}} + \boldsymbol{f}^\top M \boldsymbol{g}. \tag{9}$$

Eq. (9) combines the original ambient smoothness with an intrinsic smoothness measure defined in the deformation term $\boldsymbol{f}^\top M \boldsymbol{g}$. With the modified data-dependent inner product, we can define a deformed kernel function $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j)$ associated with $\widetilde{\mathcal{H}}$ as follows.

Definition 2 [25]: Let $\mathcal{H}$ denote the original RKHS reproduced by the kernel function $\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)$, and $\widetilde{\mathcal{H}}$ denote the deformed RKHS. Given the relationship between the two Hilbert spaces, i.e., $\langle f, g\rangle_{\widetilde{\mathcal{H}}} = \langle f, g\rangle_{\mathcal{H}} + \boldsymbol{f}^\top M \boldsymbol{g}$, the deformed kernel function $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j)$ associated with $\widetilde{\mathcal{H}}$ is defined as

$$\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) - \mu \boldsymbol{k}_{\mathbf{x}_i}^\top (I + M\boldsymbol{K})^{-1} M \boldsymbol{k}_{\mathbf{x}_j}. \tag{10}$$

Here, I is an identity matrix. $\boldsymbol{K} = \big[\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j)\big]_{n\times n}$ is the original kernel matrix in $\mathcal{H}$. $\boldsymbol{k}_{x_i}$ and $\boldsymbol{k}_{x_j}$ denote the column vectors $\boldsymbol{k}_{\mathbf{x}_i} = \big[\mathcal{K}(\mathbf{x}_i, \mathbf{x}_1), \cdots, \mathcal{K}(\mathbf{x}_i, \mathbf{x}_n)\big]^\top \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{k}_{\mathbf{x}_j} = \big[\mathcal{K}(\mathbf{x}_j, \mathbf{x}_1), \cdots, \mathcal{K}(\mathbf{x}_j, \mathbf{x}_n)\big]^\top \in \mathbb{R}^{n\times 1}$, respectively. $\mu \geq 0$ is the kernel deformation parameter controlling the smoothness of the functions.

B. Data-Dependent Kernel With Graph Laplacian

From Eq. (10), preserving the geometrical structure of the data largely depends on M. The spectral graph theory [36] indicates that the geometrical structure can be approximated by the graph Laplacian associated with the data points. Considering a graph with n vertices where each vertex corresponds to a data point $\mathbf{x}_i \in Sym_d^+$, we define the edge weight matrix $W \in \mathbb{R}^{n\times n}$ as

$$W_{ij} = \begin{cases} 1, & if\ \mathbf{x}_i \in N_\varepsilon(\mathbf{x}_j)\ or\ \mathbf{x}_j \in N_\varepsilon(\mathbf{x}_i) \\ 0, & otherwise, \end{cases} \tag{11}$$

where $N_\varepsilon(\mathbf{x}_j)$ represents the set of ε nearest neighbors of $\mathbf{x}_j$, which can be effectively computed by Log-Euclidean distance defined in Eq. (4). In our experiments, the neighborhood size is empirically set to 5.

Let $L = D - W$, where D is a diagonal matrix whose elements are column (or row) sums of W, $D_{ii} = \sum_j W_{ij}$. L is called graph Laplacian. By setting $M = L$, we obtain the following manifold adaptive kernel:

$$\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j) = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) - \mu \boldsymbol{k}_{\mathbf{x}_i}^\top (I + L\boldsymbol{K})^{-1} L \boldsymbol{k}_{\mathbf{x}_j}. \tag{12}$$

When $\mu = 1$, we are able to better understand the kernel deformation. In this case, Eq. (12) can be rewritten as

$$\begin{aligned} \widetilde{\boldsymbol{K}} &= \boldsymbol{K} - \boldsymbol{K}^\top (I + L\boldsymbol{K})^{-1} L\boldsymbol{K} \\ &= \boldsymbol{K}\big[(I + L\boldsymbol{K})^{-1}(I + L\boldsymbol{K}) - (I + L\boldsymbol{K})^{-1} L\boldsymbol{K}\big] \\ &= \boldsymbol{K}(I + L\boldsymbol{K})^{-1} \\ &= \boldsymbol{K}(\boldsymbol{K}^{-1}\boldsymbol{K} + L\boldsymbol{K})^{-1} \\ &= \boldsymbol{K}\big((\boldsymbol{K}^{-1} + L)\boldsymbol{K}\big)^{-1} \\ &= (\boldsymbol{K}^{-1} + L)^{-1}. \end{aligned} \tag{13}$$

Here $\widetilde{\boldsymbol{K}} = \big[\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j)\big]_{n\times n}$ is the kernel matrix computed by the new kernel function $\widetilde{\mathcal{K}}(\cdot,\cdot)$. The new kernel matrix $\widetilde{\boldsymbol{K}}$ can be regarded as the "reciprocal mean" of matrix $\boldsymbol{K}^{-1}$ and L [37]. Since Eq. (11) reflects the relationships between data points, each element of W can be used to effectively evaluate how a data point $\mathbf{x}_i$ resembles another data point $\mathbf{x}_j$. In this case, the graph Laplacian L measures the variation of the decision function f along the graph built from all samples. In other words, the geometry prior of the data points is included by L. Based on Eq. (13), $\widetilde{\boldsymbol{K}}$ is likely to be governed by L when strong geometrical relationships exist between all the data points. That is, $\widetilde{\boldsymbol{K}}$ is significantly deformed by the geometrical relationships. In contrast, for the extreme case in which there are no relationships between the data points, the adjacency matrix W will reduce to an identity matrix. Therefore, L will be a *zero* matrix, and Eq. (13) is equivalent to the original (undeformed) kernel.

C. Kernels for SPD Matrices

Since SPD matrices do not lie on the Euclidean space, an arithmetic subtraction would not measure the distance between two SPD matrices. Consequently, traditional kernels (*e.g.*, Gaussian kernel, polynomial kernel, and linear kernel) cannot be directly transformed to manifold adaptive kernels. To address this issue, we adopt a more accurate geodesic distance on the manifold to define kernels on Sym_d^+. Nevertheless, not all geodesic distances yield positive definite kernels. In this paper, we state two positive definite kernels on Sym_d^+ through the true geodesic distance, as illustrated in Theorem 1 and Theorem 2.

Before the statement of Theorem 1 and Theorem 2, we introduce the definition of the positive definite kernel [38].

Definition 3: Let $\mathcal{X}$ be a nonempty set. A function $\mathcal{K}(\cdot,\cdot)$: $\mathcal{X}\times\mathcal{X} \longrightarrow \mathbb{R}$ is called a positive definite kernel if and only if $\mathcal{K}(\cdot,\cdot)$ is symmetric and for all $n \in \mathbb{N}$, $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \subseteq \mathcal{X}$ gives rise to a positive definite Gram matrix, i.e., for all $Z = \{z_1, z_2, \cdots, z_n\} \in \mathbb{R}^n$, we have

$$\sum_{i,j=1}^{n} z_i z_j \boldsymbol{K}_{i,j} \geq 0, \quad where \quad \boldsymbol{K}_{i,j} := \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j).$$

Theorem 1: Let $\mathcal{K}^G : Sym_d^+ \times Sym_d^+ \to \mathbb{R} : \mathcal{K}^G(\mathbf{x}_i, \mathbf{x}_j) = \exp\big(-\gamma \,\|\log(\mathbf{x}_i) - \log(\mathbf{x}_j)\|_F^2\big)$. $\mathcal{K}^G$ defines a positive definite kernel for all $\gamma \in \mathbb{R}$.

Proof: We use $\boldsymbol{K}^G = \big[\mathcal{K}^G(\mathbf{x}_i, \mathbf{x}_j)\big]_{n\times n}$ to denote the kernel matrix. Based on ***Definition*** 3, $\mathcal{K}^G$ is positive definite if and only if $Z^\top \boldsymbol{K}^G Z \geq 0, \forall Z \in \mathbb{R}^n$. Note that $\mathcal{K}^G(\mathbf{x}_i, \mathbf{x}_i) = 1$, *i.e.,* $\boldsymbol{K}^G_{i,j} = 1$ for $i = j$. Thus, expanding $Z^\top \boldsymbol{K}^G Z$ yields

$$\begin{aligned} Z^\top \boldsymbol{K}^G Z &= \sum_{i=1}^{n}\sum_{j=1}^{n} z_i \boldsymbol{K}^G_{i,j} z_j \\ &= \sum_{i=1}^{n}\sum_{j=i}^{n} z_i z_j \boldsymbol{K}^G_{i,j} + \sum_{i=1}^{n}\sum_{j\neq i} z_i z_j \boldsymbol{K}^G_{i,j} \\ &= \Big(\sum_{i=1}^{n} z_i\Big)^2 - \sum_{i=1}^{n}\sum_{j\neq i} z_i z_j + \sum_{i=1}^{n}\sum_{j\neq i} z_i z_j \boldsymbol{K}^G_{i,j} \\ &= \Big(\sum_{i=1}^{n} z_i\Big)^2 + \sum_{i=1}^{n}\sum_{j\neq i} z_i z_j (\boldsymbol{K}^G_{i,j} - 1). \end{aligned} \tag{14}$$

Since $\boldsymbol{K}^G_{i,j} \in (0,1]$, for $\forall z_i,\ z_j,\ \min\left(z_i z_j(\boldsymbol{K}^G_{i,j}-1)\right) = -z_i z_j$ holds. We get

$$\begin{aligned}\min\left(Z^\top \boldsymbol{K}^G Z\right) &= \Big(\sum_{i=1}^{n} z_i\Big)^2 - \sum_{i=1}^{n}\sum_{j\neq i} z_i z_j \\ &= \sum_{i=1}^{n}(z_i)^2 \geq 0.\end{aligned}$$

■

Theorem 2: Let $\mathcal{K}^L : Sym_d^+ \times Sym_d^+ \to \mathbb{R} : \mathcal{K}^L(\mathbf{x}_i,\mathbf{x}_j) = tr\big(\log(\mathbf{x}_i)\log(\mathbf{x}_j)\big)$, *where tr is the matrix trace operation.* $\mathcal{K}^L$ *defines a positive definite kernel.*

Proof: Using the notation $\log(\mathbf{x}_1) = A = [a_{ij}]_{d\times d}$, $\log(\mathbf{x}_2) = B = [b_{ij}]_{d\times d}$, we denote $C = AB = [c_{ij}]_{d\times d} = \big(\sum_{k=1}^{d} a_{ik}b_{kj}\big)_{d\times d}$. Since B is a symmetric matrix, we get

$$\begin{aligned}tr\big(\log(\mathbf{x}_i)\log(\mathbf{x}_j)\big) &= tr(C) = \sum_{i=1}^{d} c_{ii} = \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij}b_{ji} \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij}b_{ij} = \Big\langle \log(\mathbf{x}_i), \log(\mathbf{x}_j)\Big\rangle.\end{aligned}$$

Therefore, $tr\big(\log(\mathbf{x}_i)\log(\mathbf{x}_j)\big)$ is an inner product. The induced norm can be used to define the distance which is equal to the geodesic distance. Furthermore, to show that the kernel $\mathcal{K}^L$ is positive definite, based on ***Definition*** 3, we need to prove that $Z^\top \boldsymbol{K}^L Z \geq 0$ for $\forall z \in \mathbb{R}^n$, *i.e.,*

$$\begin{aligned}Z^\top \boldsymbol{K}^L Z &= \sum_{i=1}^{n}\sum_{j=1}^{n} z_i \boldsymbol{K}^L z_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} z_i\ tr\big[\log(\mathbf{x}_i)\cdot\log(\mathbf{x}_j)\big] z_j \\ &= tr\left[\left(\sum_{i=1}^{n} z_i \log(\mathbf{x}_i)\right)^2\right] \\ &= \left\|\sum_{i=1}^{n} z_i \log(\mathbf{x}_i)\right\|_F^2 \geq 0.\end{aligned}$$

■

Based on Theorem 1 and Theorem 2, positive definite kernels $\mathcal{K}^G$ and $\mathcal{K}^L$ can be directly transformed to manifold kernels $\widetilde{\mathcal{K}}^G$ and $\widetilde{\mathcal{K}}^L$ on the Riemannian manifold of SPD matrices, respectively,

$$\begin{cases}\widetilde{\mathcal{K}^G}(\mathbf{x}_i,\mathbf{x}_j) = \mathcal{K}^G(\mathbf{x}_i,\mathbf{x}_j) - \mu\big(\boldsymbol{k}^G_{\mathbf{x}_i}\big)^\top (I + L\boldsymbol{K}^G)^{-1} L\boldsymbol{k}^G_{\mathbf{x}_j} \\ \widetilde{\mathcal{K}^L}(\mathbf{x}_i,\mathbf{x}_j) = \mathcal{K}^L(\mathbf{x}_i,\mathbf{x}_j) - \mu\big(\boldsymbol{k}^L_{\mathbf{x}_i}\big)^\top (I + L\boldsymbol{K}^L)^{-1} L\boldsymbol{k}^L_{\mathbf{x}_j}.\end{cases} \tag{15}$$

In the remaining part of this article, notation $\widetilde{\mathcal{K}}$, instead of $\widetilde{\mathcal{K}^G}$ and $\widetilde{\mathcal{K}^L}$, is used to present the manifold kernel specified in Eq. (12) for brevity.

IV. Manifold Kernel Sparse Representation on Sym_d^+

In the space of Sym_d^+, we do not use the linear combination of atoms $\widehat{\mathbf{x}}_i = \sum_{j=1}^{N} \alpha_{ij}\mathbf{d}_j$ to represent the data $\mathbf{x}_i$, since the approximation $\widehat{\mathbf{x}}_i$ corresponding to $\mathbf{x}_i$ may not be on the Riemannian manifold. In this section, we perform the SR of SPD matrices by embedding Riemannian manifold into RKHS using the manifold kernels introduced in Sect. III.

A. Sparse Coding

Employing the manifold kernels specified in Eq. (15) induced by the feature mapping function ϕ: $\mathcal{R}^d \to \mathcal{R}^{\mathcal{F}}$, the data points $\mathcal{X}$ on Sym_d^+ are transformed to the corresponding feature space $\{\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \cdots, \phi(\mathbf{x}_n)\}$. The kernel similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined by $\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_j)$. The dictionary $\mathcal{D}$ in the feature space is denoted by $\{\phi(\mathbf{d}_1), \phi(\mathbf{d}_2), \cdots, \phi(\mathbf{d}_N)\}$. The similarity between dictionary atoms and the original data points can also be computed using the kernel function as $\phi(\mathbf{d}_i)^\top\phi(\mathbf{x}_j) = \widetilde{\mathcal{K}}(\mathbf{d}_i,\mathbf{x}_j)$. Similarly, the similarity among dictionary atoms is $\phi(\mathbf{d}_i)^\top\phi(\mathbf{d}_j) = \widetilde{\mathcal{K}}(\mathbf{d}_i,\mathbf{d}_j)$. For the Riemannian data points $\mathbf{x}$ on Sym_d^+, we solve a sparse vector $\boldsymbol{\alpha} \in \mathbb{R}^{N\times n}$ such that $\phi(\mathbf{x})$ admits the sparse representation $\boldsymbol{\alpha}$ over the dictionary $\phi(\boldsymbol{D})$. The kernelized sparse coding is given by

$$\min_{\boldsymbol{\alpha}} \|\phi(\mathcal{X}) - \phi(\mathcal{D})\boldsymbol{\alpha}\|_F^2 + \lambda\|\boldsymbol{\alpha}\|_1. \tag{16}$$

For each manifold point $\mathbf{x}_i$, Eq. (16) can be expanded as

$$\begin{aligned}&\Big\|\phi(\mathbf{x}_i) - \phi(\mathcal{D})\boldsymbol{\alpha}_i\Big\|_F^2 + \lambda\|\alpha_i\|_1 \\ &\quad= \Big\|\phi(\mathbf{x}_i) - \sum_{j=1}^{N}\phi(\mathbf{d}_j)\alpha_{j,i}\Big\|_F^2 + \lambda\|\boldsymbol{\alpha}_i\|_1 \\ &\quad= \widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{x}_i) - 2\sum_{j=1}^{N}\alpha_{j,i}\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{d}_j) \\ &\qquad+ \sum_{j=1}^{N}\sum_{t=1}^{N}\alpha_{j,i}\alpha_{t,i}\widetilde{\mathcal{K}}(\mathbf{d}_i,\mathbf{d}_j) + \lambda\|\boldsymbol{\alpha}_i\|_1 \\ &\quad= \phi(\mathbf{x}_i)^\top\phi(\mathbf{x}_i) - 2\boldsymbol{\alpha}_i^\top\phi(\mathcal{D})^\top\phi(\mathbf{x}_i) \\ &\qquad+\boldsymbol{\alpha}_i^\top\phi(\mathcal{D})^\top\phi(\mathcal{D})\boldsymbol{\alpha}_i + \lambda\|\boldsymbol{\alpha}_i\|_1 \\ &\quad= \widetilde{\boldsymbol{K}}_{\mathbf{x}_i\mathbf{x}_i} - 2\boldsymbol{\alpha}_i^\top\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i} + \boldsymbol{\alpha}_i^\top\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}}\boldsymbol{\alpha}_i + \lambda\|\boldsymbol{\alpha}_i\|_1.\end{aligned} \tag{17}$$

Here, $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}}$ is a $N\times N$ matrix. It contains the kernel similarities between all the dictionary atoms, *i.e.,* $\widetilde{\mathcal{K}}(\mathbf{d}_t,\mathbf{d}_j)$, where $t = 1,2,\cdots,N$ and $j = 1,2,\cdots,N$. $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i} \in \mathbb{R}^{N\times 1}$ consists of $\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{d}_j)$, $j = 1,2,\cdots,N$. $\boldsymbol{\alpha}_i \in \mathbb{R}^{N\times 1}$ corresponds to the sparse code of $\mathbf{x}_i$. The objective function in Eq. (17) is similar to the sparse coding problem except for the definitions of $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}}$ and $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i}$ which can be calculated by the manifold kernel functions in Eq. (15).

To derive an efficient solution to kernel sparse coding, we introduce the following theorem.

Theorem 3: In the RKHS $\widetilde{\mathcal{H}}$, *consider the least-square problem*

$$\begin{aligned}&\min_{\boldsymbol{\alpha}_i}\Big\|\phi(\mathbf{x}_i) - \sum_{j=1}^{N}\phi(\boldsymbol{d}_j)\alpha_{j,i}\Big\|_F^2 \\ &\qquad\equiv \min_{\boldsymbol{\alpha}_i}\widetilde{\boldsymbol{K}}_{\mathbf{x}_i\mathbf{x}_i} - 2\boldsymbol{\alpha}_i^\top\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i} + \boldsymbol{\alpha}_i^\top\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}}\boldsymbol{\alpha}_i.\end{aligned} \tag{18}$$

Let $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$ *be the singular value decomposition (SVD) of the the symmetric positive definite matrix* $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}}$. *Then the problem*

defined in Eq. (18) is equivalent to the least-square problem in $\mathbb{R}^N$

$$\min_{\boldsymbol{\alpha}_i} \|\Theta - \Lambda\boldsymbol{\alpha}_i\|_2^2,$$

where $\Lambda = (\boldsymbol{U}\Sigma^{1/2})^\top$, $\Theta = (\Lambda^\top)^{-1}\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i}$.[1]

Proof: The symmetric positive definite matrix $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}}$ is rewritten as $\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^\top$ through singular value decomposition (SVD). $\boldsymbol{U}$ is an an orthonormal matrix and $\boldsymbol{\Sigma}$ is a diagonal matrix. More specifically,

$$\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}} = \boldsymbol{U}\Sigma\boldsymbol{U}^\top = \boldsymbol{U}\Sigma^{1/2}(\Sigma^{1/2})^\top\boldsymbol{U}^\top.$$

For simplicity, let $\Lambda = (\boldsymbol{U}\Sigma^{1/2})^\top$, then

$$\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathcal{D}} = \Lambda^\top\Lambda.$$

Since $\Lambda^\top(\Lambda^\top)^{-1} = I$, $\widetilde{\boldsymbol{K}}_{\mathbf{x}_i}$ can be given by

$$\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i} = \Lambda^\top(\Lambda^\top)^{-1}\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i}.$$

Similarly, let $\Theta = (\Lambda^\top)^{-1}\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i}$, then

$$\widetilde{\boldsymbol{K}}_{\mathcal{D}\mathbf{x}_i} = \Lambda^\top\Theta.$$

Since the optimization of $\boldsymbol{\alpha}_i$ is independent on Θ, we add $\Theta^\top\Theta$ into Eq. (18) and omit constant $\widetilde{\boldsymbol{K}}_{\mathbf{x}_i\mathbf{x}_i}$ with no impact on minimizing Eq. (18). Thus, we get

$$\min_{\boldsymbol{\alpha}_i} \Theta^\top\Theta - 2\boldsymbol{\alpha}_i^\top\Lambda^\top\Theta + \boldsymbol{\alpha}_i^\top\Lambda^\top\Lambda\boldsymbol{\alpha}_i \equiv \min_{\boldsymbol{\alpha}_i} \|\Theta - \Lambda\boldsymbol{\alpha}_i\|^2.$$

■

Based on the ***Theorem*** 3, minimizing Eq. (17) is equivalent to solving

$$\min_{\boldsymbol{\alpha}_i} \|\Theta - \Lambda\boldsymbol{\alpha}_i\|^2 + \lambda\|\boldsymbol{\alpha}_i\|_1. \tag{19}$$

Eq. (19) is a standard Lasso problem [39], which can be solved efficiently with the SPAMS package [30]. Since we solve Eq. (19) by fixing the dictionary $\mathcal{D}$, both $\widetilde{\boldsymbol{K}}_{DD}$ and $(\Lambda^\top)^{-1}$ are computed only once. In our experiments, we set $\lambda = 0.01$.

B. Dictionary Learning

When the kernel sparse codes for the given manifold data points $\mathcal{X}$ are computed, the dictionary can be updated such that the reconstruction error for each $\mathbf{x}_i$ is minimized. The dictionary-learning problem, therefore, can be formulated as

$$\min_{\boldsymbol{\alpha}_i,\mathcal{D}} \sum_{i=1}^{n} \left\| \phi(\mathbf{x}_i) - \sum_{j=1}^{N} \phi(\mathbf{d}_j)\alpha_{j,i} \right\|_F^2 + \lambda\|\boldsymbol{\alpha}_i\|_1. \tag{20}$$

Writing the first term of the objective in Eq. (20) as a function of $\mathcal{D}$ for dictionary update, we have

$$f(\mathcal{D}) = \sum_{i=1}^{n}\Bigg[1 - 2\sum_{j=1}^{N}\alpha_{j,i}\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{d}_j) + \sum_{j=1}^{N}\sum_{l=1}^{N}\alpha_{j,i}\alpha_{l,i}\widetilde{\mathcal{K}}(\mathbf{d}_l,\mathbf{d}_j)\Bigg], \tag{21}$$

[1]According to the ***Definition*** 3, $\mathcal{K}^G$ (*resp.* $\mathcal{K}^L$) is positive definite if and only if $Z^\top\boldsymbol{K}^G Z \geq 0$ (*resp.* $Z^\top\boldsymbol{K}^L Z \geq 0$), for $\forall Z \in \mathbb{R}^n$. Therefore, $(\Lambda^\top)^{-1}$ is a pseudo inverse.

where $\alpha_{j,i}$ denotes the i-th element in the coefficient vector $\boldsymbol{\alpha}_j$. After initializing the dictionary $\mathcal{D}^1$, we solve the optimization by repeating two steps (*i.e.,* sparse coding and dictionary update). To update dictionary atoms $\{\mathbf{d}_j\}_{j=1}^N$, we compute the partial derivative of Eq. (21) with respect to $\mathbf{d}_j$:

$$\frac{\partial f(\mathcal{D})}{\partial \mathbf{d}_j} = \sum_{i=1}^{n}\Bigg[-2\alpha_{j,i}\frac{\partial\widetilde{\mathcal{K}}(\mathbf{x}_i,\mathbf{d}_j)}{\partial\mathbf{d}_j} + \sum_{l=1}^{N}\alpha_{j,i}\alpha_{l,i}\frac{\partial\widetilde{\mathcal{K}}(\mathbf{d}_l,\mathbf{d}_j)}{\partial\mathbf{d}_j}\Bigg]. \tag{22}$$

We take $\widetilde{\mathcal{K}^G}$ as an example to perform the dictionary learning ($\widetilde{\mathcal{K}^L}$ can be carried out by a similar scheme). We set Eq. (22) to 0 using the definition of manifold adaptive kernel functions $\widetilde{\mathcal{K}^G}(\mathbf{x}_i,\mathbf{x}_j)$ in Eq. (15) as follows:

$$\begin{aligned}
-4\gamma\boldsymbol{d}_j^{-1}\sum_{i=1}^{n}\Bigg[&-\alpha_{j,i}\Big(\mathcal{K}^G(\mathbf{x}_i,\mathbf{d}_j)\Big(\log(\mathbf{d}_j)-\log(\mathbf{x}_i)\Big)+\mu\\
&\cdot\sum_{s=1}^{n}\Phi_s\mathcal{K}^G(\mathbf{x}_s,\mathbf{d}_j)\Big(\log(\mathbf{d}_j)-\log(\mathbf{x}_s)\Big)\Big)\\
&+\sum_{l=1}^{N}\alpha_{j,i}\alpha_{l,i}\Big(\mathcal{K}^G(\mathbf{d}_l,\mathbf{d}_j)\Big(\log(\mathbf{d}_j)-\log(\mathbf{d}_l)\Big)\\
&-\mu\cdot\sum_{s=1}^{n}\Psi_s\mathcal{K}^G(\mathbf{x}_s,\mathbf{d}_j)\\
&\times\Big(\log(\mathbf{d}_j)-\log(\mathbf{x}_s)\Big)\Big)\Bigg]\\
=0,&
\end{aligned} \tag{23}$$

where

$$\begin{cases}\Phi = \left(\boldsymbol{k}_{x_i}^G\right)^\top(I + L\boldsymbol{K}^G)^{-1}L\\ \Psi = \left(\boldsymbol{k}_{d_l}^G\right)^\top(I + L\boldsymbol{K}^G)^{-1}L.\end{cases} \tag{24}$$

Here, $\Phi \in \mathbb{R}^{1\times n}$ and $\Psi \in \mathbb{R}^{1\times n}$. During updating, each dictionary atom is updated independently. At time $t+1$, $\mathbf{d}_j$ is updated using the results of time t. $\mathbf{d}_j^{(t)}$ represents the j-th atom in the t-th iteration. Eq. (23) can be rewritten as

$$\begin{aligned}
-4\sum_{i=1}^{n}\Bigg[&-\alpha_{j,i}\Big(\mathcal{K}^G(\mathbf{x}_i,\mathbf{d}_j^{(t)})\Big(\log(\mathbf{d}_j)^{(t+1)}-\log(\mathbf{x}_i)\Big)\\
&-\mu\cdot\sum_{s=1}^{n}\Phi_s\mathcal{K}^G(\mathbf{x}_s,\mathbf{d}_j^{(t)})\\
&\times\Big(\log(\mathbf{d}_j)^{(t+1)}-\log(\mathbf{x}_s)\Big)\Big)+\sum_{l=1}^{N}\alpha_{j,i}\alpha_{l,i}\\
&\times\Big(\mathcal{K}^G(\mathbf{d}_l,\mathbf{d}_j^{(t)})\Big(\log(\mathbf{d}_j)^{(t+1)}-\log(\mathbf{d}_l)\Big)-\mu\\
&\cdot\sum_{s=1}^{n}\Psi_s\mathcal{K}^G(\mathbf{x}_s,\mathbf{d}_j^{(t)})\Big(\log(\mathbf{d}_j)^{(t+1)}-\log(\mathbf{x}_s)\Big)\Big)\Bigg]\\
=0.&
\end{aligned} \tag{25}$$

By solving Eq. (25), we obtain the iterative formulation in Eq. (27), as shown at bottom of the next page. $diag[\cdot]$ denotes

Algorithm 1 Dictionary Learning on $Sym^+(n)$ Using *Kernel Trick*

Input:
- Original data points $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ on the Riemannian manifold of SPD matrices, where each $\mathbf{x}_i \in Sym^+(n)$ is a SPD matrix.
- The Riemannian kernel function $\widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_j)$.
- The number of iterations $iter$.

Output:
- The Riemannian dictionary $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_N\}$.

Compute $\widetilde{\boldsymbol{K}}^{(t)}_{\mathbf{x_i}\mathbf{x_i}} = \widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{x}_i)$, $i = 1, 2, \cdots, n$.
Initialize the dictionary $\mathcal{D}^1 = \{\mathbf{d}_j\}_{j=1}^N$ by Riemannian clustering using Karcher mean.
for $t = 1 \rightarrow iter$ **do**
  Compute $\widetilde{\boldsymbol{K}}^{(t)}_{\mathcal{D}\mathcal{D}} = \widetilde{\mathcal{K}}(\mathbf{d}_t^{(t)}, \mathbf{d}_j^{(t)})$, $t, j = 1, 2, \cdots, N$;
  Compute $\widetilde{\boldsymbol{K}}^{(t)}_{\mathcal{D}\mathbf{x}_i} = \widetilde{\mathcal{K}}(\mathbf{x}_i, \mathbf{d}_j^{(t)})$, $i = 1, 2, \cdots, n$; $j = 1, 2, \cdots, N$.
  Compute $\|\Theta - \Lambda\boldsymbol{\alpha}_i\|^2 + \lambda\|\boldsymbol{\alpha}_i\|_1$, $\forall i$, $\mathbf{x}_i \in \mathcal{X}$.
  for $j = 1 \rightarrow N$ **do**
    compute $\log(\mathbf{d}_j)^{(t+1)}$ according to Eq. (27);
    $\mathbf{d}_j = \exp(\log(\mathbf{d}_j)^{(t+1)})$;
    $\mathbf{d}_j^{(t+1)} \leftarrow \frac{\mathbf{d}_j}{\|\mathbf{d}_j\|_2}$.
  end
end

a diagonal matrix using the element as its diagonal. $\mathbf{1_n}$ and $\mathbf{1_N}$ are n-dimensional and N-dimensional 1s column vectors, respectively. $\boldsymbol{\alpha}_j \in \mathbb{R}^{1\times n}$ is a row vector containing the set of coefficients of data points corresponding to the dictionary atom $\mathbf{d}_j$. Note that $\boldsymbol{K}^G = \left[\mathcal{K}^G(\mathbf{x}_i, \mathbf{x}_j)\right]_{n\times n}$ is replaced with $\boldsymbol{K}$ in Eq. (27) for simplicity. Kernel matrix $\boldsymbol{K}^{(t)}_{\mathbf{d}_j,\mathcal{D}} \in \mathbb{R}^{1\times N}$ contains the kernel similarity elements $\mathcal{K}^G\left(\mathbf{d}_j^{(t)}, \mathbf{d}_k\right)$ between each atom $\mathbf{d}_j$ and the entire dictionary $\mathcal{D}$, where $k = 1, 2, \cdots, N$. Similarly, kernel matrix $\boldsymbol{K}^{(t)}_{\mathbf{d}_j,\mathcal{X}} \in \mathbb{R}^{1\times n}$ contains the kernel similarity elements $\mathcal{K}^G\left(\mathbf{d}_j^{(t)}, \mathbf{x}_i\right)$ between each atom $\mathbf{d}_j$ and all data points. Algorithm 1 gives the details for dictionary learning.

V. Experiments

In this section, we evaluate the proposed manifold kernel sparse representation using three applications: visual tracking, face recognition and image classification. Our visual tracking experiments are implemented in MATLAB2012b on an Intel Core2 2.5 GHz processor with 4GB RAM, and the image classification and face recognition experiments are run in MATLAB2012b on an Intel(R) Xeon(R) 2.93 GHz processor with 24GB RAM.

A. Visual Tracking

Motivated by recent advances of sparse coding for visual tracking [3], [40]–[42], we employ the sparse coding of SPD matrices as the object representation. Tracking is then carried out within a Bayesian inference framework, in which the binratio similarity function [43] of sparse histograms between the candidate and the template are used to construct the observation model.

1) Experimental Setup: To take the local appearance information of patches into consideration, we resize the object image to 32×32 pixels and extract 36 overlapped 12×12 sliding windows (or local patches) within the object region with four pixels as the step length. Following [27], the covariance descriptors are computed from the feature vector $\left[x, y, I, |I_x|, |I_y|, \sqrt{(I_x)^2 + (I_y)^2}, |I_{xx}|, |I_{yy}|\right]^\top$. The covariance matrix for each image patch, therefore, is an 8×8 SPD matrix. With the overlapped patches extracted from the object region in the first frame, k-means clustering is performed in the Log-Euclidean framework [26] to obtain the dictionary $\mathcal{D}$ with 72 atoms. The sparse coefficient vectors of patches are normalized and concatenated to form a histogram representation by $[\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_{36}]^\top$. The parameters γ and μ are set to 1 and 0.01, respectively. Due to space limitations, we only provide the corresponding tracking results of the $\widetilde{\mathcal{K}^G}$ kernel sparse coding of SDP matrices in this paper. The performance of the $\widetilde{\mathcal{K}^L}$ kernel is comparable to the $\widetilde{\mathcal{K}^G}$ in the visual tracking scenario.

We compare our tracker with the state-of-the-art sparsity-based tracking algorithms including L1 [44], APGL1 [45], LSK [42], MTT [2], LSST [41], SCM [40], and MLSAM [3]. We run our method on eight challenging video sequences which suffer from heavy occlusions, illumination changes, pose variations, motion blur, scale variations and complex backgrounds.

2) Quantitative Comparisons: One widely used evaluation method to measure tracking results is the center location error. It is based on the relative position errors (in pixels) between the central locations of the tracked object and those of the ground truth. From Table I, we can see that our algorithm achieves the lowest tracking error in almost all the sequences. However, when the tracker loses the object for several frames, the output location can be random and therefore the average center location errors may not evaluate the tracking performance correctly. In this paper, *precision plot* is also adopted to measure the overall tracking performance. This shows the percentage of frames whose estimated locations are within

$$
\begin{aligned}
&\log(\boldsymbol{d}_j)^{(t+1)} \\
&= \frac{\log(\mathcal{D})diag\left[\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{D}}\right]\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top - \log(\mathcal{X})diag\left[\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\right]\boldsymbol{\alpha}_j^\top + \mu\log(\mathcal{X})diag\left[\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\right]\Phi^\top\mathbf{1}_{\boldsymbol{n}}^\top\boldsymbol{\alpha}_j^\top - \mu\log(\mathcal{X})diag\left[\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\right]\Psi^\top\mathbf{1}_{\boldsymbol{N}}^\top\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top}{\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{D}}\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top - \boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\boldsymbol{\alpha}_j^\top + \mu\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\Phi^\top\mathbf{1}_{\boldsymbol{n}}^\top\boldsymbol{\alpha}_j^\top - \mu\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\Psi^\top\mathbf{1}_{\boldsymbol{N}}^\top\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top} \\
&= \frac{\log(\mathcal{D})diag\left[\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{D}}\right]\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top - \log(\mathcal{X})diag\left[\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\right]\left(\boldsymbol{\alpha}_j^\top - \mu\Phi^\top\mathbf{1}_{\boldsymbol{n}}^\top\boldsymbol{\alpha}_j^\top + \mu\Psi^\top\mathbf{1}_{\boldsymbol{N}}^\top\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top\right)}{\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{D}}\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top - \boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\boldsymbol{\alpha}_j^\top + \mu\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\Phi^\top\mathbf{1}_{\boldsymbol{n}}^\top\boldsymbol{\alpha}_j^\top - \mu\boldsymbol{K}^{(t)}_{\boldsymbol{d}_j,\mathcal{X}}\Psi^\top\mathbf{1}_{\boldsymbol{N}}^\top\boldsymbol{\alpha}\boldsymbol{\alpha}_j^\top}
\end{aligned}
\tag{27}
$$

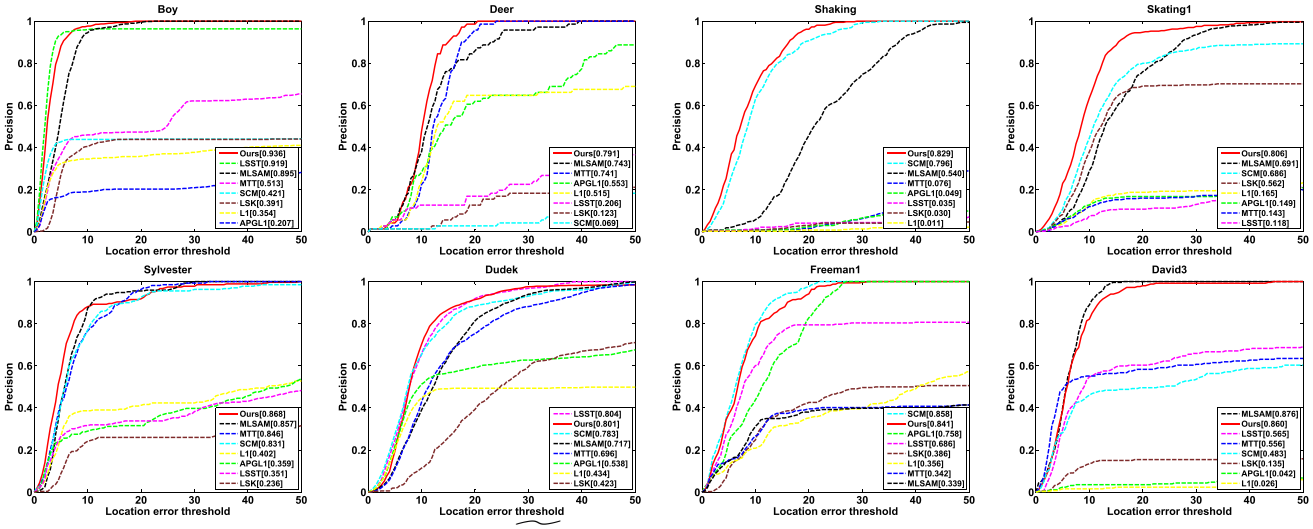


Fig. 2. Precision plot for 8 representative sequences using $\widetilde{\mathcal{K}^G}$ kernel. The performance score of each tracker is shown in the legend (best viewed on high-resolution display).

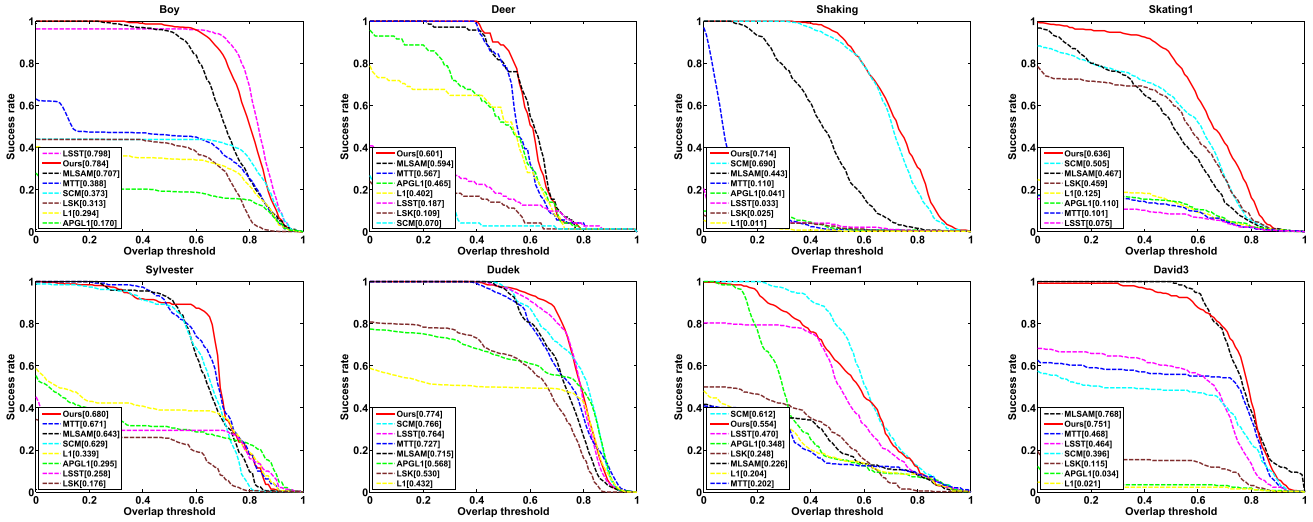


Fig. 3. Success rate curve for 8 representative sequences using $\widetilde{\mathcal{K}^G}$ kernel. The performance score of each tracker is shown in the legend (best viewed on high-resolution display).

TABLE I

CENTER LOCATION ERROR (CLE)(IN PIXELS). **BOLD** FONT INDICATES THE BEST PERFORMANCE AND *italic* FONT INDICATES THE SECOND BEST PERFORMANCE

	APGL1	LSST	MTT	SCM	MLSAM	L1	LSK	Ours
Sylvester	47.9	53.6	7.6	8.5	*6.9*	50.3	96.4	**6.4**
Freeman1	11.9	29.8	123.1	**6.9**	99.1	54.5	92.4	*7.8*
David3	237.0	71.5	58.7	73.1	**6.0**	250.1	227.0	*6.9*
Boy	*58.5*	284.8	492.0	98.7	98.7	93.5	176.3	**39.0**
Deer	88.1	5.2	43.0	51.1	*5.0*	83.8	180.6	**3.0**
Shaking	26.3	89.7	12.8	102.6	*12.7*	41.3	176.5	**10.3**
Skating1	122.3	121.6	55.1	*10.0*	23.0	174.2	114.2	**8.4**
Dudek	151.1	159.1	255.8	35.1	*15.4*	117.2	44.0	**9.5**

the given threshold (*e.g.*, 20 pixels). More accurate trackers have higher precision at lower thresholds. We provide the precision plot results of eight trackers over eight representative sequences, as shown in Fig. 2. We see that the proposed tracker achieves the most robust and accurate tracking performance in most video sequences.

The tracking overlap rate indicates the stability of each algorithm as it takes the size and pose of the target object into account. It is defined by $score = \frac{area(ROI_T \bigcap ROI_G)}{area(ROI_T \bigcup ROI_G)}$, where ROI_T is the tracking bounding box and ROI_G is the ground truth. This can be used to evaluate the success rate of any tracking approach. Table II gives the average overlap rates. Overall, the proposed tracker outperforms the state-of-the-art methods.

Generally, the tracking result is considered as a success when the $score$ is greater than the given threshold t_s. It may not be fair or representative for tracker evaluation merely using one specific threshold (*e.g.*, $t_s = 0.5$), We therefore count the number of successful frames at the thresholds which vary from 0 to 1 and plot the *success rate* curve for our tracker and the compared trackers. The *area under curve* (AUC) of each success rate plot is employed to rank the tracking algorithms. Robust trackers have higher success rates at higher thresholds. The success rate curve of eight representative sequences is illustrated in Fig. 3. We can see that the proposed method achieves the best tracking performance on most video sequences.

Fig. 4. The tracking results of eight trackers over the representative frames of the *Shaking*, *Skating1*, *Boy*, *Deer*, *Dudek*, *Freeman1*, *Sylvester* and *David3* sequences from top to bottom.

TABLE II
OVERLAP RATE (OR)(%). **BOLD** FONT INDICATES THE BEST PERFORMANCE AND *italic* FONT INDICATES THE SECOND BEST PERFORMANCE

	APGL1	LSST	MTT	SCM	MLSAM	L1	LSK	Ours
Sylvester	29.5	25.8	*67.3*	63.0	64.4	34.0	17.7	**68.2**
Freeman1	34.7	47.1	20.2	**61.3**	22.6	20.4	24.8	*55.5*
David3	3.4	46.6	47.0	39.7	**77.2**	2.1	11.5	*75.3*
Boy	*48.1*	21.0	21.1	32.3	32.3	47.6	9.4	**68.8**
Deer	17.0	**80.1**	38.9	37.5	70.9	29.5	31.4	*78.6*
Shaking	46.5	18.7	56.7	6.9	*59.5*	40.2	10.9	**60.2**
Skating1	4.1	3.3	10.6	*69.2*	44.3	1.1	2.5	**71.6**
Dudek	11.0	7.5	10.1	*50.6*	46.7	12.5	46.0	**63.8**

3) Qualitative Comparisons: We report the tracking results of eight trackers (highlighted by bounding boxes in different colors) over the representative frames of the eight video sequences, as shown in Fig. 4. In the "Shaking" sequence, the target undergoes pose variation, illumination change, and partial occlusion. The SCM, L1, and LSK trackers drift from the object quickly when the spotlight blinks suddenly (*e.g.,* frame ♯60). MLSAM and our trackers can successfully track the surfer throughout the sequence with relatively accurate bounding box sizes. MTT and APGL1 methods are able to track the object in this sequence but with less success than our method. In the "Skating1" sequence, the dancer continuously changes her pose on a stage with a complex background as well as drastic illumination variations. The L1, APGL1, LSST, MTT and LSK methods cannot track the object correctly. The MLSAM method performs slightly better. Our tracker loses the object at frame ♯359, but recovers at frame ♯368. Overall, the SCM and our method outperform the other trackers.

"Boy" and "Deer" sequences are used to evaluate whether our method is able to tackle fast motion. In the "Boy" sequences, a boy jumps irregularly as the object moves quickly. It is difficult to predict the boy's location. Most methods fail to track the object at the beginning of the sequence (*e.g.,* ♯95). In contrast, our method achieves relatively lower

TABLE III
AVERAGE CLASSIFICATION RATE (%) ON SCENE 15 DATASET

Num. of atoms	32	64	128	256	512
Log-E kernel [23]	$\mathbf{75.84 \pm 0.64}$	$\mathbf{79.27 \pm 0.65}$	80.92 ± 0.44	–	–
Our $\widetilde{\mathcal{K}^L}$ kernel	73.49 ± 0.58	77.99 ± 0.50	81.57 ± 0.46	82.46 ± 0.47	82.69 ± 0.43
Our $\widetilde{\mathcal{K}^G}$ kernel	74.21 ± 0.47	78.56 ± 0.56	$\mathbf{83.35 \pm 0.39}$	$\mathbf{84.18 \pm 0.42}$	$\mathbf{84.31 \pm 0.44}$

center location errors and higher success rates than other methods. In the "Deer" sequence, the appearance change caused by motion blur is very drastic. APGL1, MTT, SCM, L1 and LSK trackers do not perform well in some frames (*e.g.,* ♯40, ♯63). Though LSST and MLSAM trackers are able to keep track of the object to the end, the proposed approach achieves both the lowest tracking error and the highest overlap rate.

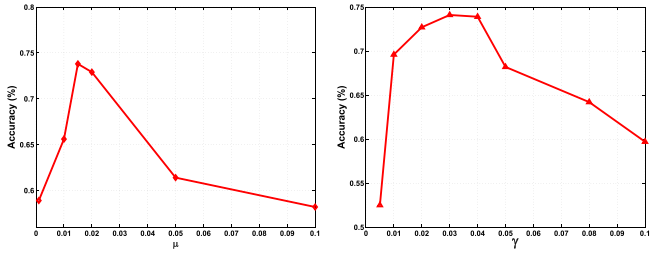
In the "Dudek", "Freeman1" and "Sylvester" sequences, the objects suffer from large pose and view changes. For the "Dudek" sequence, we see that the LSK tracker loses the target very quickly at the beginning of the sequence (*e.g.*, ♯363). The LSST, MTT, and APGL1 trackers fail when scale change occurs (*e.g.*, ♯853). In contrast, our method has both relatively low center location error and high overlap rate, as shown in Table I and Table II. For the "Freeman1" sequence, although the LSST tracker obtains slightly better results than MTT, MLSAM, LSK and L1 trackers, it loses the object after drastic pose change (*e.g.*, ♯265). In comparison, APGL1 and our trackers track the object successfully. We note that the SCM performs better than the other methods. For the "Sylvester" sequence, APGL1, LSST, L1 and LSK trackers are unable to locate the object in the whole sequence. In contrast, MTT, SCM, MLSAM and our method track the object well and provide tracking boxes that are much more accurate and consistent.

In the "David3" sequence, the person suffers from partial occlusion as well as drastic pose variations. It is difficult to handle these two challenges. The SCM, APGL1, L1 and MTT methods fail to track the object when the person walks behind a tree (*e.g.*, ♯84). The LSST and LSK methods lose the object when the person changes direction (*e.g.*, ♯159). In comparison, only MLSAM and our tracker succeed throughout this sequence.

B. Image Classification

We evaluate our method on the Scene 15 dataset [46] and Brodatz dataset [47]. The Scene 15 dataset contains 4485 images of 15 different scenes. The number of images per category ranges from 200 to 400. The dataset contains not only indoor scenes, *e.g.,* store and living-room, but also outdoor scenes, *e.g.,* streets and mountain. To be consistent with previous work [23], we use the same setting to extract 64 covariance descriptors. We randomly select 100, 000 covariance matrices from the total covariance descriptors of all images to learn the dictionary.

In the Brodatz dataset, each class corresponds to only one image. All 111 texture images are used to train the dictionary. Following the previous works in [20] and [23], we normalize each training image to 256×256 pixels, and obtain non-overlapping blocks of 32×32 pixels. A 5×5 covariance descriptor is then computed from each of these blocks using the feature vector $\left[I, |I_x|, |I_y|, |I_{xx}|, |I_{yy}|\right]^\top$. Twenty blocks from each image are randomly selected for the dictionary learning.



(a) (b)

Fig. 5. The effect of the parameters μ and γ on the Brodatz dataset. (a) The relationship between the classification accuracy and the parameter μ. (b) The relationship between the classification accuracy and the parameter γ.

1) Parameter Selection: The kernel deformation parameter μ is an important factor in the manifold kernel sparse representation. We experimentally test the effect of μ on the Brodatz dataset. In the manifold kernel $\widetilde{\mathcal{K}^G}$, $\mathcal{K}^G(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \| \log(\mathbf{x}_i) - \log(\mathbf{x}_j) \|_F^2\right)$, we fix the parameter $\gamma = 1/d$, where d is the dimensionality of the covariance descriptor. We list the results based on different μ ranging from 0.001 to 1 in Fig. 5. We observe that our method obtains good performance when μ is fixed at either 0.015 or 0.02. In addition, when $\widetilde{\mathcal{K}^G}$ is used in the sparse representation, kernel parameter γ is also very important and affects classification accuracy. To depict the relationship between γ and classification accuracy, we set $\gamma = 3^n/d$, where n ranges from -3 to 2 with step size 1. The relationship between γ and classification accuracy is shown in Fig. 5. We see that our method achieves promising results in a wide range of γ value. For simplicity, we set $\gamma = 1/d$ in the following experiments.

2) Result Comparisons: For the Scene 15 dataset, the Bag-of-Words image representation [46] is applied for training and classification. Following the common experimental settings, we randomly select 100 images per category as training data and use the rest as test data. The reported results of the dataset are the averages of 20 independent experiments. Table III shows that the proposed kernels have comparable performance with other competitive methods. Note that the $\widetilde{\mathcal{K}^G}$ ($\mu = 0.015$, $\gamma = 0.03$) kernel achieves the best results. We notice that as the number of atoms grows, the average classification rates increase, as illustrated in Table III, but the results are very close. The underlying reason is that our dictionary-learning method is generative without

TABLE IV
COMPARISON OF CLASSIFICATION ACCURACY (%) ON THE BRODATZ DATASET

Method	♯ Training samples per class		
	20	25	30
RSR [20]	60.65 ± 0.90	–	–
Log-E kernel [23]	71.34 ± 0.61	72.53 ± 0.74	74.87 ± 0.69
Our $\widetilde{\mathcal{K}^L}$ kernel	72.17 ± 0.65	74.03 ± 0.72	75.14 ± 0.56
Our $\widetilde{\mathcal{K}^G}$ kernel	$\mathbf{73.62 \pm 0.71}$	$\mathbf{75.19 \pm 0.47}$	$\mathbf{76.62 \pm 0.63}$

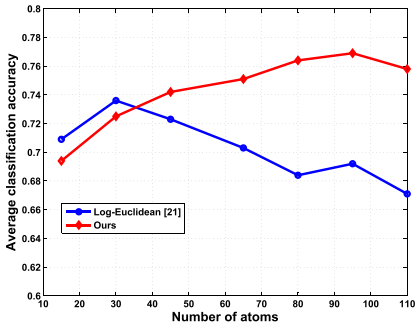


Fig. 6. The average classification accuracy curves vs. the number of atom matrices on the Brodatz dataset.

discriminative information, therefore, good representational capability does not necessarily signify promising discriminability.

We use the Brodatz dataset [47] to evaluate the performance of our method on texture classification. We evaluate classification performance with the number of training samples fixed at 20, 25, and 30 covariance matrices per class, with the remaining samples being used for testing. The reported performance is obtained by averaging over ten random splits of training and test sets. The KNN classifier is employed for the classification task ($k = 3$). The results are reported in Table IV. It can be seen that our $\widetilde{\mathcal{K}^L}$ kernel ($\mu = 0.015$) achieves comparable performance with the Log-Euclidean method [23]. $\widetilde{\mathcal{K}^G}$ kernel ($\mu = 0.015$, $\gamma = 0.02$) performs better than other approaches, because the radial basis function $\widetilde{\mathcal{K}^G}$ better reflects the geometrical structure of the manifold data than the polynomial kernel $\widetilde{\mathcal{K}^L}$. To further analyze our results, we plot the average classification accuracy curves using $\widetilde{\mathcal{K}^G}$ versus the number of atom matrices in Fig. 6. We see that as the number of atoms increases, the improvement of our method grows. Our method achieves the best result when the number of atoms is equal to 95.

C. Face Recognition

In this section, we present experimental results for face recognition on the FERET dataset [48] and the Extended Yale B dataset [49]. The sparse representation classifier [4] is adopted for the classification task. We use the "b" subset of the FERET dataset for the evaluation of recognition performance. The subset includes 1400 images from 198 subjects. In our experiments, the images are resized to 64×64 pixels. The Extended Yale B dataset contains about $2,414$ frontal face images of 38 subjects. The face images are taken under varying illumination conditions. We normalize face images of size 54×48.

TABLE V
FACE RECOGNITION RATE (%) ON THE FERET DATASET

	SRC [4]	GSRC [5]	TSC [12]	RSR [20]	Log-E kernel [23]	Ours	
						$\widetilde{\mathcal{K}^L}$	$\widetilde{\mathcal{K}^G}$
bg	26.0	79.0	44.5	86.0	94.5	91.5	**95.0**
bf	61.0	97.0	73.5	97.5	100	100	**100**
be	55.5	93.5	73.0	96.5	99.0	97.5	**99.5**
bd	27.5	77.0	36.0	79.5	91.5	89.0	91.0
ave.	42.5	86.6	56.8	89.9	96.3	94.5	**96.4**

TABLE VI
COMPARISON OF THE AVERAGE RECOGNITION ACCURACY (%) ON THE EXTENDED YALE B DATASET

Method	Dimensionality		
	30	84	150
SRC [4]	90.9	95.5	96.8
GSRC [5]	88.1	92.4	96.9
Log-E kernel [23]	95.4		
Our $\widetilde{\mathcal{K}^L}$ kernel	97.4		
Our $\widetilde{\mathcal{K}^G}$ kernel	**98.1**		

1) Experimental Setup: To obtain the manifold kernel sparse representation, a 43×43 covariance descriptor is used to describe a face image using the feature vector

$$\big[I(x,y), x, y, |G_{00}(x,y)|, \cdots, |G_{07}(x,y)|, \cdots, |G_{47}(x,y)|\big]^\top,$$

where $I(x,y)$ is the intensity value at position (x,y), and $G_{uv}(x,y)$ is the response of a 2D Gabor filter along five orientations and eight angles. For each pixel (x,y), the dimensionality of the Gabor features is 40. For the FERET dataset, images marked "ba", "bj" and "bk" are used as training data, and images with "bd", "be", "bf" and "bg" labels are test data. We randomly split the Extended Yale B dataset into two halves. One half containing 32 images for each person is used as the dictionary, and the other half is used for testing. On the Extended Yale B dataset, we conduct experiments 10 times by randomly selecting training and testing sets, and we report the average result.

2) Result Comparisons: Table V shows the recognition rates on the FERET dataset compared with SRC [4], GSRC [5], TSC [12], RSR [20], and Log-E kernel [23]. We see that neither SRC nor TSC is able to obtain satisfactory results, because the Riemannian manifold of Sym_d^+ lacks a global linear structure which allows the SPD matrix to be generated linearly by the atoms in $\mathcal{D}$. GSRC and RSR improve the recognition rates. Overall, the proposed method with $\widetilde{\mathcal{K}^L}$ achieves comparable performance with the Log-E kernel method, and $\widetilde{\mathcal{K}^G}$ ($\mu = 0.02$, $\gamma = 0.01$) is slightly better than the Log-E kernel method.

Table VI shows the recognition rates versus feature dimension by SRC and GSRC on the Extended Yale B dataset. Since the Log-E kernel and our method acquire the covariance descriptors which are independent of each image, it is not necessary for these two methods to reduce the dimensionality of the original face images. The results of both SRC and GSRC are from [5]. Our method with $\widetilde{\mathcal{K}^G}$ ($\mu = 0.02$, $\gamma = 0.015$) achieves better results than the other methods,

but the $\widetilde{\mathcal{K}^L}$ kernel is slightly worse than SRC and GSRC when the dimension is 150.

VI. Conclusion

In this paper, we have presented a manifold kernel sparse representation method for symmetric positive definite (SPD) matrices. The sparse representation on the space of SPD matrices can be performed by embedding the SPD matrices into a reproducing kernel Hilbert space (RKHS) using the data-dependent manifold kernel function. The graph Laplacian is incorporated into the manifold kernel space to discover the underlying geometrical structure of the manifold. Experimental results of visual tracking, face recognition and image classification show that our algorithm outperforms existing sparse coding based approaches, and compares favorably with state-of-the-art methods.

Acknowledgement

The authors would like to thank Zhen Dong and Shiye Zhang for their efforts in conducting the experiments, and greatly appreciate Sue Felix polishing our presentation. Yuwei Wu thanks Mehrtash Harandi and Shengping Zhang for the useful discussion on the revised version of our paper.

References

[1] Z. Xiao, H. Lu, and D. Wang, "L2-RLS based object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 8, pp. 1301–1309, Aug. 2014.

[2] T. Zhang, B. Ghanem, S. Liu, and N. Ahuja, "Robust visual tracking via structured multi-task sparse learning," *Int. J. Comput. Vis.*, vol. 101, no. 2, pp. 367–383, 2013.

[3] Y. Wu, B. Ma, M. Yang, Y. Jia, and J. Zhang, "Metric learning based structural appearance model for robust visual tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 5, pp. 865–877, May 2014.

[4] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, Feb. 2009.

[5] M. Yang and L. Zhang, "Gabor feature based sparse representation for face recognition with Gabor occlusion dictionary," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 448–461.

[6] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 92–104, Jan. 2013.

[7] M. Yang, L. Zhang, X. Feng, and D. Zhang, "Sparse representation based Fisher discrimination dictionary learning for image classification," *Int. J. Comput. Vis.*, vol. 109, no. 3, pp. 209–232, 2014.

[8] A. Cherian, S. Sra, A. Banerjee, and N. Papanikolopoulos, "Jensen–Bregman LogDet divergence with application to efficient similarity search for covariance matrices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 9, pp. 2161–2174, Sep. 2013.

[9] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel methods on the Riemannian manifold of symmetric positive definite matrices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 73–80.

[10] H. E. Cetingul and R. Vidal, "Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 1896–1902.

[11] M. Harandi, C. Sanderson, C. Shen, and B. Lovell, "Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 3120–3127.

[12] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Tensor sparse coding for positive definite matrices," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 592–605, Mar. 2014.

[13] R. Sivalingam, D. Boley, V. Morellas, and N. Papanikolopoulos, "Positive definite dictionary learning for region covariances," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Nov. 2011, pp. 1013–1019.

[14] S. Sra and A. Cherian, "Generalized dictionary learning for symmetric positive definite matrices with application to nearest neighbor retrieval," in *Machine Learning and Knowledge Discovery in Databases*. Berlin, Germany: Springer-Verlag, 2011, pp. 318–332.

[15] A. Cherian and S. Sra, "Riemannian sparse coding for positive definite matrices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 299–314.

[16] M. Harandi, R. Hartley, B. Lovell, and C. Sanderson, "Sparse coding on symmetric positive definite manifolds using Bregman divergences," *IEEE Trans. Neural Netw. Learn. Syst.*, doi: 10.1109/TNNLS.2014.2387383, 2015.

[17] X. Zhang, W. Li, W. Hu, H. Ling, and S. Maybank, "Block covariance based ℓ_1 tracker with a subtle template dictionary," *Pattern Recognit.*, vol. 46, no. 7, pp. 1750–1761, 2013.

[18] K. Guo, P. Ishwar, and J. Konrad, "Action recognition using sparse representation on covariance manifolds of optical flow," in *Proc. IEEE Int. Conf. Adv. Video Signal Based Surveill. (AVSS)*, Aug./Sep. 2010, pp. 188–195.

[19] C. Yuan, W. Hu, X. Li, S. Maybank, and G. Luo, "Human action recognition under Log-Euclidean Riemannian metric," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, 2010, pp. 343–353.

[20] M. T. Harandi, C. Sanderson, R. Hartley, and B. C. Lovell, "Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 216–229.

[21] S. Zhang, S. Kasiviswanathan, P. C. Yuen, and M. Harandi, "Online dictionary learning on symmetric positive definite manifolds with vision applications," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2015, pp. 1–9.

[22] A. Barachant, S. Bonnet, M. Congedo, and C. Jutten, "Classification of covariance matrices using a Riemannian-based kernel for BCI applications," *Neurocomputing*, vol. 112, pp. 172–178, Jul. 2013.

[23] P. Li, Q. Wang, W. Zuo, and L. Zhang, "Log-Euclidean kernels for sparse representation and dictionary learning," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1601–1608.

[24] R. Vemulapalli, J. K. Pillai, and R. Chellappa, "Kernel learning for extrinsic classification of manifold features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 1782–1789.

[25] V. Sindhwani, P. Niyogi, and M. Belkin, "Beyond the point cloud: From transductive to semi-supervised learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 824–831.

[26] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Geometric means in a novel vector space structure on symmetric positive-definite matrices," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 1, pp. 328–347, 2006.

[27] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on Riemannian manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 10, pp. 1713–1727, Oct. 2008.

[28] Y. Pang, Y. Yuan, and X. Li, "Gabor-based region covariance matrices for face recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 7, pp. 989–993, Jul. 2008.

[29] J. Y. Tou, Y. H. Tay, and P. Y. Lau, "Gabor filters as feature images for covariance matrix on texture classification problem," in *Advances in Neuro-Information Processing*. Berlin, Germany: Springer-Verlag, 2009, pp. 745–751.

[30] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.

[31] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 2021–2024.

[32] H. Van Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 5123–5135, Dec. 2013.

[33] S. Gao, I. W.-H. Tsang, and L.-T. Chia, "Sparse representation with kernels," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 423–434, Feb. 2013.

[34] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Multiple kernel sparse representations for supervised and unsupervised learning," *IEEE Trans. Image Process.*, vol. 23, no. 7, pp. 2905–2915, Jul. 2014.

[35] M. J. Gangeh, A. Ghodsi, and M. S. Kamel, "Kernelized supervised dictionary learning," *IEEE Trans. Signal Process.*, vol. 61, no. 19, pp. 4753–4767, Oct. 2013.

[36] F. R. K. Chung, *Spectral Graph Theory*, vol. 92. Providence, RI, USA: AMS, 1997.

[37] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu, "Semi-supervised SVM batch mode active learning for image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–7.

[38] C. van den Berg, J. P. R. Christensen, and P. Ressel, *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. New York, NY, USA: Springer-Verlag, 1984.
[39] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.
[40] W. Zhong, H. Lu, and M.-H. Yang, "Robust object tracking via sparse collaborative appearance model," *IEEE Trans. Image Process.*, vol. 23, no. 5, pp. 2356–2368, May 2014.
[41] D. Wang, H. Lu, and M.-H. Yang, "Least soft-threshold squares tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 2371–2378.
[42] B. Liu, J. Huang, C. Kulikowski, and L. Yang, "Robust visual tracking using local sparse appearance model and K-selection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2968–2981, Dec. 2013.
[43] N. Xie, H. Ling, W. Hu, and X. Zhang, "Use bin-ratio information for category and scene classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 2313–2319.
[44] X. Mei and H. Ling, "Robust visual tracking using ℓ_1 minimization," in *Proc. IEEE 12th Int. Conf. Comput. Vis. (ICCV)*, Sep./Oct. 2009, pp. 1436–1443.
[45] C. Bao, Y. Wu, H. Ling, and H. Ji, "Real time robust L1 tracker using accelerated proximal gradient approach," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 1830–1837.
[46] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. Jun. 2006, pp. 2169–2178.
[47] T. Randen and J. H. Husoy, "Filtering for texture classification: A comparative study," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 4, pp. 291–310, Apr. 1999.
[48] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000.
[49] K.-C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.

Yuwei Wu received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2014. He is currently a Research Fellow with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He has strong research interests in computer vision, medical image processing, and object tracking. He received a National Scholarship for Graduate Students and an Academic Scholarship for Ph.D. Candidates from the Ministry of Education, China, an Outstanding Ph.D. Thesis Award and XU TELI Excellent Scholarship from the Beijing Institute of Technology, and a CASC Scholarship from the China Aerospace Science and Industry Corporation.

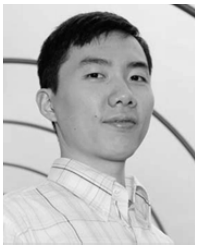
Yunde Jia (M'11) received the B.S., M.S., and Ph.D. degrees in mechatronics from the Beijing Institute of Technology (BIT), in 1983, 1986, and 2000, respectively. He was a Visiting Scientist with Carnegie Mellon University from 1995 to 1997, and a Visiting Fellow with the Australian National University in 2011. He has served as the Executive Dean of the School of Computer Science at BIT from 2005 to 2008. He is currently a Professor of Computer Science at BIT, and serves as the Director of the Beijing Laboratory of Intelligent Information Technology. His current research interests include computer vision, media computing, and intelligent systems.

Peihua Li received the B.S. and M.S. degrees from the Harbin Engineering University, China, in 1993 and 1996, respectively, and the Ph.D. degree in computer science and technology from the Harbin Institute of Technology, China, in 2002. In 2003, he visited Microsoft Research Asia. From 2003 to 2004, he was a Post-Doctoral Fellow at INRIA, IRISA, Rennes, France. From 2004 to 2013, he was with the School of Computer Science and Technology, Heilongjiang University. He is currently with the School of Information and Communication Engineering, Dalian University of Technology. His research interests include computer vision, pattern recognition, and machine learning. He received the Best Ph.D. Dissertation Award from the Harbin Institute of Technology, in 2004, and honorary nomination for the National Excellent Doctoral Dissertation Award in China in 2005.

Jian Zhang (SM'04) received the B.Sc. degree from East China Normal University, Shanghai, China, in 1982, the M.Sc. degree in computer science from Flinders University, Adelaide, Australia, in 1994, and the Ph.D. degree in electrical engineering from the University of New South Wales (UNSW), Sydney, Australia, in 1999.

He was with the Visual Information Processing Laboratory, Motorola Laboratories, Sydney, from 1997 to 2003, as a Senior Research Engineer, and later became a Principal Research Engineer and a Foundation Manager with the Visual Communications Research Team. From 2004 to 2011, he was a Principal Researcher and a Project Leader with National ICT Australia, Sydney, and a Conjoint Associate Professor with the School of Computer Science and Engineering, UNSW. He is currently an Associate Professor with the Advanced Analytics Institute, School of Software, Faculty of Engineering and Information Technology, University of Technology at Sydney, Sydney, and a Visiting Researcher with the Neville Roach Laboratory, National ICT Australia, Kensington, Australia. He is the author or co-author of more than 100 paper publications, book chapters, and six issued patents filed in the U.S. and China. His current research interests include social multimedia signal processing, image and video processing, machine learning, pattern recognition, human–computer interaction, and intelligent healthcare systems.

Dr. Zhang was the General Chair of the IEEE International Conference on Multimedia and Expo in 2012. He is also an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY and the *EURASIP Journal on Image and Video Processing*.



Junsong Yuan (M'08) received the Ph.D. degree from Northwestern University, USA, and the M.Eng. degree from the National University of Singapore. He graduated from the Special Class for the Gifted Young of the Huazhong University of Science and Technology, China. He is currently an Assistant Professor and a Program Director of Video Analytics with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. He is the author or co-author of more than100 paper publications, book chapters, and six issued patents filed in the U.S. and China. His research interests include computer vision, video analytics, large-scale visual search and mining, and human–computer interaction.

He has served as the Area Chair of the IEEE Winter Conference on Computer Vision, the IEEE Conference on Multimedia Expo (ICME) in 2014, and the Asian Conference on Computer Vision (ACCV) in 2014, the Organizing Chair of ACCV in 2014, and the Co-Chair Workshops at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 12, 13) and the IEEE Conference on Computer Vision. He currently serves as an Associate Editor of *The Visual Computer* journal (TVC) and the *Journal of Multimedia*. He received the Nanyang Assistant Professorship and Tan Chin Tuan Exchange Fellowship from Nanyang Technological University, an Outstanding EECS Ph.D. Thesis Award from Northwestern University, the Best Doctoral Spotlight Award from CVPR 09, and the National Outstanding Student from the Ministry of Education, China. He has recently given tutorials at the IEEE ICIP13, FG13, ICME12, SIGGRAPH VRCAI12, and PCM12.