Dimensionality Reduction on SPD Manifolds: The Emergence of Geometry-Aware Methods

Mehrtash Harandi, *Member, IEEE*, Mathieu Salzmann, *Member, IEEE*, and Richard Hartley, *Fellow, IEEE*

Abstract—Representing images and videos with Symmetric Positive Definite (SPD) matrices, and considering the Riemannian geometry of the resulting space, has been shown to yield high discriminative power in many visual recognition tasks. Unfortunately, computation on the Riemannian manifold of SPD matrices –especially of high-dimensional ones– comes at a high cost that limits the applicability of existing techniques. In this paper, we introduce algorithms able to handle high-dimensional SPD matrices by constructing a lower-dimensional SPD manifold. To this end, we propose to model the mapping from the high-dimensional SPD manifold to the low-dimensional one with an orthonormal projection. This lets us formulate dimensionality reduction as the problem of finding a projection that yields a low-dimensional manifold either with maximum discriminative power in the supervised scenario, or with maximum variance of the data in the unsupervised one. We show that learning can be expressed as an optimization problem on a Grassmann manifold and discuss fast solutions for special cases. Our evaluation on several classification tasks evidences that our approach leads to a significant accuracy gain over state-of-the-art methods.

Index Terms—Riemannian manifolds, Riemannian geometry, symmetric positive definite matrices, Grassmann manifolds, dimensionality reduction, visual recognition

1 Introduction

Dimensionality Reduction (DR) is imperative in various disciplines of computer science, including machine learning and computer vision. Conventional methods, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), are specifically designed to work with real-valued vectors coming from a flat Euclidean space. In modern computer vision, however, the data and the mathematical models often naturally lie on Riemannian manifolds (e.g., subspaces form Grassmannian, 2D shapes lie on Kendall shape spaces [1]). There has therefore been a growing need and interest to go beyond the extensively studied Euclidean spaces and analyze non-linear and curved Riemannian manifolds. In this context, a natural question arises: *How can popular DR techniques be extended to curved Riemannian spaces?* A principled answer to this question will open the door to exploiting higher-dimensional, more discriminative features, and thus to improved accuracies in a wide range of applications involving classification and clustering.

This paper tackles the problem of dimensionality reduction on the space of Symmetric Positive Definite (SPD) matrices, i.e., the SPD manifold. In computer vision, SPD matrices have been successfully employed for a variety of tasks, such as analyzing medical imaging [2], segmenting images [3] and recognizing textures [4], [5], pedestrians [6], [7], [8], faces [9], [10], [11], and actions [12], [13].

The set of SPD matrices is clearly not a vector space as it is not closed under addition and scalar product (e.g., multiplying a positive definite matrix with a negative scalar makes it negative definite). As such, analyzing SPD matrices through the geometry of Euclidean spaces, such as using the Frobenius inner product as a means to measure similarity, is not only unnatural, but also inadequate. This inadequacy has recently been demonstrated in computer vision by a large body of work, e.g., [2], [6], [8]. One striking example is the *swelling effect* that occurs in diffusion tensor imaging (DTI), where a matrix represents the covariance of the local Brownian motion of water molecules [2]—when considering Euclidean geometry to interpolate between two diffusion tensors, the determinant of the intermediate matrices may become strictly larger than the determinants of both original matrices, which, from a physics point of view, is unacceptable.

A popular and geometric way to analyze SPD matrices is through the Riemannian structure induced by the Affine Invariant Riemannian Metric (AIRM) [2], which is usually referred to as SPD manifold. The geodesic distance induced by the AIRM is related to the distance induced by the Fisher-Rao metric on the manifold of multivariate Gaussian distributions with fixed means (see for example [14]). It enjoys several properties, such as invariance to affine transformations, which are of particular interest in computer vision.

While the Riemannian structure induced by the AIRM has been shown to overcome the limitations of Euclidean geometry to a great extent, the computational cost of the resulting

- *M. Harandi and R. Hartley are with the College of Engineering and Computer Science, Australian National University, Canberra and Data61-CSIRO, Canberra Research Laboratory, Canberra, ACT 2601, Australia. E-mail: mehrtash.harandi@data61.csiro.au, richard.hartley@anu.edu.au.*
- *M. Salzmann is with the CVLab, EPFL, Lausanne CH-1015, Switzerland. E-mail: mathieu.salzmann@epfl.ch.*

Manuscript received 18 May 2016; revised 11 Jan. 2017; accepted 12 Jan. 2017. Date of publication 17 Jan. 2017; date of current version 12 Dec. 2017. Recommended for acceptance by K. Weinberger.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2017.2655048

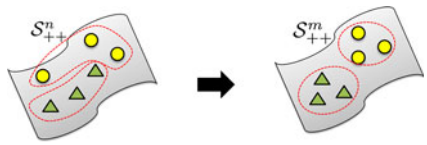

Fig. 1. *Dimensionality reduction on SPD manifolds:* Given data on a high-dimensional SPD manifold, where each sample represents an $n \times n$ SPD matrix, we learn a mapping to a lower-dimensional SPD manifold. We consider both the supervised scenario, illustrated here, where the resulting $m \times m$ SPD matrices are clustered according to class labels, and the unsupervised one, where the resulting matrices have maximum variance.

techniques increases drastically with the dimension of the manifold (i.e., the size of the SPD matrices). As a consequence, with the exception of a few works that handle medium-sized [3], [10] or high-dimensional [15], previous studies have opted for low-dimensional SPD matrices (e.g., region covariance descriptors obtained from low-dimensional features). Clearly, and as evidenced by the recent feature-learning trends in computer vision, low-dimensional features are bound to be less powerful and discriminative. In other words, to match or even outperform state-of-the-art recognition systems on complex tasks, manifold-based methods will need to exploit high-dimensional SPD matrices. In fact, this is what was achieved in [15] by using deep learning features to compute covariance descriptors for image recognition. The effectiveness of this method, however, comes at a high computational cost. In this paper, we propose to address this drawback by introducing supervised and unsupervised DR techniques dedicated to SPD manifolds, as illustrated by Fig. 1.

More specifically, in the supervised scenario, we introduce an approach that constructs a lower-dimensional and more discriminative SPD manifold from a high-dimensional one. To this end, we encode the notion of discriminative power by pulling together the training samples from the same class while pushing apart those from different classes. We study three variants of this approach, where the distance is defined by either the AIRM, the Stein divergence [16], or the Jeffrey divergence [17]. The latter two divergences are motivated by the fact that they share invariance properties with the AIRM while being faster to compute.

In the unsupervised scenario, we draw inspiration from the Maximum Variance Unfolding (MVU) algorithm [18]. That is, we introduce a method that maps a high-dimensional SPD manifold to a low-dimensional one, where the training matrices are furthest apart from their mean. As in the supervised case, we study three variants, that rely on the AIRM, the Stein divergence and the Jeffrey divergence, respectively [19].

We demonstrate the benefits of our approach on several classification and clustering tasks where the data can be represented with high-dimensional SPD matrices. In particular, our method outperforms state-of-the-art techniques on image-based material categorization and face recognition, and action recognition from 3D motion capture sequences. A Matlab implementation of our algorithms is available from the first author's webpage .

2 Background Theory

This section provides a review of the Riemannian geometry of SPD manifolds, as well as of Bregman divergences and their properties. We conclude the section by discussing optimization on Grassmann manifolds, which will be useful in our DR algorithms.

Notation. Throughout the paper, bold capital letters denote matrices (e.g., $\boldsymbol{X}$) and bold lower-case letters denote column vectors (e.g., $\boldsymbol{x}$). $\mathbf{I}_n$ is the $n \times n$ identity matrix. $\mathrm{G}L(n)$ denotes the general linear group, i.e., the group of real invertible $n \times n$ matrices. $\mathrm{Sym}(n)$ is the space of real $n \times n$ symmetric matrices. $\mathcal{S}_{++}^{n}$ and $\mathcal{G}(p,n)$ are the SPD and Grassmannian manifolds, respectively, and will be formally defined later. $\mathrm{Diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is a diagonal matrix with the real values $\lambda_1, \lambda_2, \ldots, \lambda_n$ as diagonal elements. The principal matrix logarithm $\log(\cdot) : \mathcal{S}_{++}^{n} \rightarrow \mathrm{Sym}(n)$ is defined as

$$\log(\boldsymbol{X}) = \sum_{r=1}^{\infty} \frac{(-1)^{r-1}}{r} (\boldsymbol{X} - \mathbf{I}_n)^r = \boldsymbol{U}\mathrm{Diag}(\log(\lambda_i))\boldsymbol{U}^T, \quad (1)$$

with $\boldsymbol{X} = \boldsymbol{U}\mathrm{Diag}(\lambda_i)\boldsymbol{U}^T$. Similarly, the matrix exponential $\exp(\cdot) : \mathrm{Sym}(n) \rightarrow \mathcal{S}_{++}^{n}$ is defined as

$$\exp(\boldsymbol{X}) = \sum_{r=0}^{\infty} \frac{1}{r!} \boldsymbol{X}^r = \boldsymbol{U}\mathrm{Diag}(\exp(\lambda_i))\boldsymbol{U}^T, \quad (2)$$

with $\boldsymbol{X} = \boldsymbol{U}\mathrm{Diag}(\lambda_i)\boldsymbol{U}^T$.

2.1 The Geometry of SPD Manifolds

An $n \times n$ real SPD matrix $\boldsymbol{X}$ has the property that $\boldsymbol{v}^T\boldsymbol{X}\boldsymbol{v} > 0$ for all non-zero $\boldsymbol{v} \in \mathbb{R}^n$. The space of $n \times n$ SPD matrices, denoted by $\mathcal{S}_{++}^{n}$, forms the interior of a convex cone in the $n(n+1)/2$-dimensional Euclidean space. $\mathcal{S}_{++}^{n}$ is mostly studied when endowed with the Affine Invariant Riemannian Metric (AIRM) [2], defined as

$$\begin{aligned} \langle \boldsymbol{V}, \boldsymbol{W} \rangle_{\boldsymbol{P}} &\triangleq \langle \boldsymbol{P}^{-1/2}\boldsymbol{V}\boldsymbol{P}^{-1/2}, \boldsymbol{P}^{-1/2}\boldsymbol{W}\boldsymbol{P}^{-1/2} \rangle \\ &= \mathrm{Tr}\big(\boldsymbol{P}^{-1}\boldsymbol{V}\boldsymbol{P}^{-1}\boldsymbol{W}\big) , \end{aligned} \quad (3)$$

for $\boldsymbol{P} \in \mathcal{S}_{++}^{n}$ and $\boldsymbol{V}, \boldsymbol{W} \in T_{\boldsymbol{P}}\mathcal{S}_{++}^{n}$, where $T_{\boldsymbol{P}}\mathcal{S}_{++}^{n}$ denotes the *tangent space*[1] of $\mathcal{S}_{++}^{n}$ at $\boldsymbol{P}$. This metric induces the following geodesic distance between points $\boldsymbol{X}$ and $\boldsymbol{Y}$:

$$\delta_R(\boldsymbol{X}, \boldsymbol{Y}) = \left\| \log\left(\boldsymbol{X}^{-1/2}\boldsymbol{Y}\boldsymbol{X}^{-1/2}\right) \right\|_F . \quad (4)$$

The AIRM has several useful properties such as invariance to affine transformations (as the name implies), i.e., $\delta_R(\boldsymbol{X}, \boldsymbol{Y}) = \delta_R(\boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T, \boldsymbol{A}\boldsymbol{Y}\boldsymbol{A}^T)$, for $\boldsymbol{A} \in \mathrm{G}L(n)$. For in-depth discussions of the AIRM, we refer the interested reader to [2] and [20].

2.2 Bregman Divergences

We now introduce two divergences derived from the Bregman matrix divergence, namely the Jeffrey and Stein divergences. Below, we discuss their properties and establish some connections with the AIRM, which motivated our choice of these divergences in our DR formulations.

Definition 1. *Let $\zeta : \mathcal{S}_{++}^{n} \rightarrow \mathbb{R}$ be a strictly convex and differentiable function defined on the symmetric positive cone $\mathcal{S}_{++}^{n}$. The Bregman matrix divergence $d_\zeta : \mathcal{S}_{++}^{n} \times \mathcal{S}_{++}^{n} \rightarrow [0, \infty)$ is defined as*

$$d_\zeta(\boldsymbol{X}, \boldsymbol{Y}) = \zeta(\boldsymbol{X}) - \zeta(\boldsymbol{Y}) - \langle \nabla_{\boldsymbol{Y}}(\zeta), \boldsymbol{X} - \boldsymbol{Y} \rangle, \quad (5)$$

1. For a general manifold $\mathcal{M}$, the tangent space $T_{\boldsymbol{P}}\mathcal{M}$ is the set of all tangent vectors to $\mathcal{M}$ at $\boldsymbol{P}$ and admits the structure of a vector space.

where $\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \mathrm{Tr}(\boldsymbol{X}^T\boldsymbol{Y})$ *is the Frobenius inner product, and* $\nabla_{\boldsymbol{Y}}(\zeta)$ *represents the gradient of* ζ *at* $\boldsymbol{Y}$.

The Bregman divergence is asymmetric, non-negative, and definite (i.e., $d_\zeta(\boldsymbol{X}, \boldsymbol{Y}) = 0$, iff $\boldsymbol{X} = \boldsymbol{Y}$). While the Bregman divergence enjoys a variety of useful properties [21], its asymmetric behavior is often a hindrance. In this paper, we therefore study two types of symmetrized Bregman divergences, namely the *Stein* and the *Jeffrey* divergences.

Definition 2. *The Stein, or* S*, divergence (also known as Jensen-Bregman LogDet divergence [19]) is obtained from the Bregman divergence of Eq. (5) by using* $\zeta(\boldsymbol{X}) = -\log\det(\boldsymbol{X})$ *as seed function and by Jensen-Shannon symmetrization. This yields*

$$\begin{aligned}\delta_S^2(\boldsymbol{X}, \boldsymbol{Y}) &\triangleq \frac{1}{2} d_\zeta\left(\boldsymbol{X}, \frac{\boldsymbol{X}+\boldsymbol{Y}}{2}\right) + \frac{1}{2} d_\zeta\left(\boldsymbol{Y}, \frac{\boldsymbol{X}+\boldsymbol{Y}}{2}\right) \\ &= \log\det\left(\frac{\boldsymbol{X}+\boldsymbol{Y}}{2}\right) - \frac{1}{2}\log\det(\boldsymbol{X}\boldsymbol{Y})\ .\end{aligned} \tag{6}$$

Definition 3. *The Jeffrey, or* J*, divergence (also known as symmetric KL divergence) is obtained from the Bregman divergence of Eq. (5) by using* $\zeta(\boldsymbol{X}) = -\log\det(\boldsymbol{X})$ *as seed function and by direct symmetrization. This yields*

$$\begin{aligned}\delta_J(\boldsymbol{X}, \boldsymbol{Y}) &\triangleq \frac{1}{2} d_\zeta(\boldsymbol{X}, \boldsymbol{Y}) + \frac{1}{2} d_\zeta(\boldsymbol{Y}, \boldsymbol{X}) \\ &= \frac{1}{2}\mathrm{Tr}(\boldsymbol{X}^{-1}\boldsymbol{Y}) - \frac{1}{2}\log\det(\boldsymbol{X}^{-1}\boldsymbol{Y}) \\ &\quad + \frac{1}{2}\mathrm{Tr}(\boldsymbol{Y}^{-1}\boldsymbol{X}) - \frac{1}{2}\log\det(\boldsymbol{Y}^{-1}\boldsymbol{X}) - n \\ &= \frac{1}{2}\mathrm{Tr}(\boldsymbol{X}^{-1}\boldsymbol{Y}) + \frac{1}{2}\mathrm{Tr}(\boldsymbol{Y}^{-1}\boldsymbol{X}) - n.\end{aligned} \tag{7}$$

The S and J divergences have a variety of properties akin to those of the AIRM. An especially attractive property for the computer vision community is the invariance of the J and S divergences to affine transforms. Similar to the AIRM, for $\boldsymbol{A} \in \mathrm{G}L(n)$, we have

$$\begin{aligned}\delta_S^2(\boldsymbol{X}, \boldsymbol{Y}) &= \delta_S^2(\boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T, \boldsymbol{A}\boldsymbol{Y}\boldsymbol{A}^T), \\ \delta_J(\boldsymbol{X}, \boldsymbol{Y}) &= \delta_J(\boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^T, \boldsymbol{A}\boldsymbol{Y}\boldsymbol{A}^T).\end{aligned}$$

Furthermore, the S divergence enjoys the property of inducing valid positive definite Gaussian-like kernels, which does not hold for AIRM [8]. More specifically, given a set $\mathcal{X}$, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if $\sum_{i,j=1}^{p} a_i a_j k(\boldsymbol{X}_i, \boldsymbol{X}_j) \geq 0$ for all $p \in \mathbb{N}$, $\boldsymbol{X}_i \in \mathcal{X}$ and $a_i \in \mathbb{R}$. In Euclidean spaces, the Radial Basis Function (RBF) is the most common face when it comes to kernel functions. Unfortunately, the RBF kernel with the AIRM is not a valid positive definite kernel on the SPD manifold [8]. However, as shown in[16], the kernel

$$k_S(\boldsymbol{X}, \boldsymbol{Y}) = \exp\{-\beta\delta_S^2(\boldsymbol{X}, \boldsymbol{Y})\}, \tag{8}$$

is positive definite for

$$\beta \in \left\{\frac{1}{2}, \frac{2}{2}, \ldots, \frac{n-1}{2}\right\} \cup \left(\frac{1}{2}(n-1), \infty\right). \tag{9}$$

Having a positive definite kernel on the SPD manifold enables us to readily employ various kernel techniques such as kernel SVM, kernel k-means and kernel PCA on SPD data.

The kernel $k_J(\cdot,\cdot) = \exp\{-\beta\delta_J(\boldsymbol{X}, \boldsymbol{Y})\}$ has previously been considered to be positive definite (e.g., Eqs. (5) and (6) in [22]). However, we observed this not to be the case in general, as can be verified by building a kernel matrix from the following three samples:

$$\boldsymbol{X}_1 = \begin{bmatrix} 72 & 1 \\ 1 & 88 \end{bmatrix},\ \boldsymbol{X}_2 = \begin{bmatrix} 123 & -10 \\ -10 & 66 \end{bmatrix},\ \boldsymbol{X}_3 = \begin{bmatrix} 51 & 5 \\ 5 & 109 \end{bmatrix}.$$

The resulting matrix $[\boldsymbol{K}]_{i,j} = k_J(\boldsymbol{X}_i, \boldsymbol{X}_j)$ has a negative eigenvalue for $\beta = 1/4$.

2.3 Optimization Framework

In this work, we will be facing optimization problems with unitary constraints. That is, we will seek to solve problems of the form

$$\begin{aligned}&\min_{\boldsymbol{W}}\ f(\boldsymbol{W}) \\ &\text{s.t. } \boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_m,\end{aligned} \tag{10}$$

where $f(\boldsymbol{W})$ is a cost function and $\boldsymbol{W} \in \mathbb{R}^{n\times m}$. In many cases and especially in dimensionality reduction, it is customary to design the cost function $f(\boldsymbol{W})$ such that the whole problem can be cast as an eigenvalue problem (e.g., [23], [24], [25], [26], [27]). However, the complexity of our cost functions prohibits us from doing so. Instead, we propose to make use of manifold-based optimization techniques.

Recent advances in optimization methods formulate problems with unitary constraints as optimization problems on Stiefel or Grassmann manifolds [28], [29]. The geometrically correct setting for the minimization problem in (10) is, in general, on a Stiefel manifold. However, if the cost function $f(\boldsymbol{W})$ is independent from the choice of basis spanned by $\boldsymbol{W}$, that is if $f(\boldsymbol{W}) = f(\boldsymbol{W}\boldsymbol{R})$ for $\boldsymbol{R} \in \mathcal{O}(m)$, with $\mathcal{O}(m)$ denoting the group of $m \times m$ orthogonal matrices, then the problem is a Grassmannian problem. As will be shown later, in our case, the affine invariance property of the AIRM, the Stein divergence and the Jeffrey divergence, makes the problem Grassmannian. Below, we briefly review the Newton-type optimization employed here.

A Grassmann manifold $\mathcal{G}(m, n)$ is the space of m-dimensional linear subspaces of $\mathbb{R}^n$ for $0 < m < n$ [29]. In our work, we make use of a Riemannian Conjugate Gradient (RCG) method on the Grassmannian. RCG methods rely on the notion of gradient on the manifold. On an abstract Riemannian manifold $\mathcal{M}$, the gradient of a function $f(\cdot)$ at a point $x \in \mathcal{M}$, denoted by $\mathrm{grad} f(x)$, is the element of $T_x\mathcal{M}$ satisfying $\langle \mathrm{grad} f(x), \zeta\rangle_x = Df(x)[\zeta],\ \forall \zeta \in T_x\mathcal{M}$. Here, $\langle\cdot,\cdot\rangle_x$ is the Riemannian metric at x and $Df(x)[\zeta]$ denotes the directional derivative of f at x along direction ζ. On $\mathcal{G}(m, n)$, the gradient of a function is expressed as

$$\mathrm{grad} f(\boldsymbol{W}) = (\mathbf{I}_n - \boldsymbol{W}\boldsymbol{W}^T)\nabla_{\boldsymbol{W}}(f), \tag{11}$$

where $\nabla_{\boldsymbol{W}}(f)$ is the (usual) $n \times m$ Jacobian matrix at $\boldsymbol{W}$.

RCG methods then compute the new descent direction by combining the gradient at the current and the previous

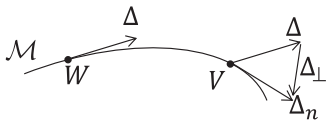

Fig. 2. Parallel transport of a tangent vector Δ from a point W to another point V on the manifold.

solutions. This requires transporting the previous gradient to the current point on the manifold. Unlike Euclidean spaces, on a manifold one cannot transport a tangent vector Δ from one point to another point by simple translation. To get a better intuition, take the case where the manifold is a sphere, and consider two tangent spaces, one located at the pole and one at a point on the equator. Obviously the tangent vectors at the pole do not belong to the tangent space at the equator. Therefore, simple vector translation is not sufficient. As illustrated in Fig. 2, transporting Δ from $\boldsymbol{W}$ to $\boldsymbol{V}$ on the manifold $\mathcal{M}$ requires subtracting the normal component $\Delta_\perp$ at $\boldsymbol{V}$ for the resulting vector to be a tangent vector. Such a transfer of tangent vector is called *parallel transport*. On the Grassmann manifold, parallel transport, and the other operations required for an RCG method, have efficient numerical forms, which makes them well-suited to perform optimization on the manifold.

RCG on a Grassmann manifold can be summarized by the following steps:

(i) Compute $\mathrm{grad} f(\boldsymbol{W})$ at the current solution.
(ii) Determine the search direction $\boldsymbol{H}$ by parallel transporting the previous search direction and combining it with $\mathrm{grad} f(\boldsymbol{W})$.
(iii) Perform a line search along the geodesic at $\boldsymbol{W}$ in the direction $\boldsymbol{H}$. On the Grassmann manifold, the geodesics going from point $\boldsymbol{X}$ in direction Δ can be represented by the geodesic equation [28]

$$\boldsymbol{X}(t) = [\,\boldsymbol{X}\boldsymbol{V} \quad \boldsymbol{U}\,] \begin{bmatrix} \cos(\Sigma t) \\ \sin(\Sigma t) \end{bmatrix} \boldsymbol{V}^T, \tag{12}$$

where t is the parameter indicating the location along the geodesic, and $\boldsymbol{U}\Sigma\boldsymbol{V}^T$ is the compact singular value decomposition of Δ.

These steps are repeated until convergence to a local minimum, or until a maximum number of iterations is reached.

Note that optimization on matrix manifolds (e.g., Stiefel, Grassmann) is at the core of several recent DR schemes [30], [31], [32]. This is in part due to the availability of the *manopt* package [33], which makes optimizing over various Riemannian manifolds straightforward. As a matter of fact, in our experiments, we used the implementation of the RCG method on Grassmannian provided by *manopt* to obtain $\boldsymbol{W}$. Note that *manopt* also provides other methods, such as trust-region solvers. A full evaluation of these solvers, however, goes beyond the scope of this paper.

With the mathematical tools discussed in this section, we can now turn to developing our DR algorithms for SPD matrices. We start the following section by introducing our approach to tackling supervised DR on SPD manifolds and then discuss the unsupervised scenario.

3 Proposed Methods

In this section, we describe our approach to learning an embedding of high-dimensional SPD matrices to a more discriminative, low-dimensional SPD manifold. In doing so, we propose to learn the parameters $\boldsymbol{W} \in \mathbb{R}^{n\times m}$, $m < n$, of a generic mapping $f_{\boldsymbol{W}} : \mathcal{S}^n_{++} \rightarrow \mathcal{S}^m_{++}$, which we define as

$$f_{\boldsymbol{W}}(\boldsymbol{X}) = \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}. \tag{13}$$

Clearly, for a full rank matrix $\boldsymbol{W}$, if $\mathcal{S}^n_{++} \ni \boldsymbol{X} \succ 0$ then $\mathcal{S}^m_{++} \ni \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W} \succ 0$. Given a set of training SPD matrices $\mathcal{X} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_p\}$, where each matrix $\boldsymbol{X}_i \in \mathcal{S}^n_{++}$, our goal is to find the transformation $\boldsymbol{W}$ such that the resulting low-dimensional SPD manifold preserves some interesting structure of the original data. In the remainder of this section, we discuss two different such structures: one coming from the availability of class labels, and one derived from unsupervised data.

3.1 Supervised Dimensionality Reduction

Let us first assume that each point $\boldsymbol{X}_i \in \mathcal{S}^n_{++}$ belongs to one of C possible classes and denote its class label by y_i. In this scenario, we propose to encode the structure of the data via an affinity function $a : \mathcal{S}^n_{++} \times \mathcal{S}^n_{++} \rightarrow \mathbb{R}$. That is $a(\boldsymbol{X}, \boldsymbol{Y})$ measures some notion of affinity between matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, and may be negative. In particular, we make use of the class labels to build $a(\cdot,\cdot)$[2] and define an affinity function that encodes the notions of intra-class and inter-class distances. In short, our goal is to find a mapping that minimizes the intra-class distances while simultaneously maximizing the inter-class distances (i.e., a discriminative mapping).

More specifically, and inspired by [34], we make use of notions of within-class similarity $g_w : \mathcal{S}^n_{++} \times \mathcal{S}^n_{++} \rightarrow \mathbb{R}_+$ and between-class similarity $g_b : \mathcal{S}^n_{++} \times \mathcal{S}^n_{++} \rightarrow \mathbb{R}_+$ to compute the affinity between two SPD matrices. In particular, we define $g_w(\cdot,\cdot)$ and $g_b(\cdot,\cdot)$ to be binary functions given by

$$g_w(\boldsymbol{X}_i, \boldsymbol{X}_j) = \begin{cases} 1, & \text{if } \boldsymbol{X}_i \in N_w(\boldsymbol{X}_j) \text{ or } \boldsymbol{X}_j \in N_w(\boldsymbol{X}_i) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

$$g_b(\boldsymbol{X}_i, \boldsymbol{X}_j) = \begin{cases} 1, & \text{if } \boldsymbol{X}_i \in N_b(\boldsymbol{X}_j) \text{ or } \boldsymbol{X}_j \in N_b(\boldsymbol{X}_i) \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

where $N_w(\boldsymbol{X}_i)$ is the set of ν_w nearest neighbors of $\boldsymbol{X}_i$ that share the same label as y_i, and $N_b(\boldsymbol{X}_i)$ contains the ν_b nearest neighbors of $\boldsymbol{X}_i$ having different labels. Note that nearest neighbors are computed according to the AIRM, the Stein divergence, or the Jeffrey divergence. The affinity function $a(\cdot,\cdot)$ is then defined as

$$a(\boldsymbol{X}_i, \boldsymbol{X}_j) = g_w(\boldsymbol{X}_i, \boldsymbol{X}_j) - g_b(\boldsymbol{X}_i, \boldsymbol{X}_j)\,, \tag{16}$$

which resembles the Maximum Margin Criterion (MMC) of [24].

Having $a(\boldsymbol{X}, \boldsymbol{Y})$ at our disposal, we propose to search for an embedding such that the affinity between pairs of

2. Note that the framework developed in this section could also apply to the unsupervised and semi-supervised settings by changing the definition of the affinity function accordingly.

TABLE 1
Definition of the Jacobian for the Stein Divergence, Jeffrey Divergence and AIRM, Respectively

Method	Jacobian
S-**divergence**	$\nabla_{\boldsymbol{W}}\Big(\delta_S^2(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})\Big) = -\boldsymbol{X}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})^{-1} - \boldsymbol{Y}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1} + (\boldsymbol{X}+\boldsymbol{Y})\boldsymbol{W}\Big(\boldsymbol{W}^T\frac{\boldsymbol{X}+\boldsymbol{Y}}{2}\boldsymbol{W}\Big)^{-1}$. (21)
J-**divergence**	$\nabla_{\boldsymbol{W}}\Big(\delta_J(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})\Big) = \boldsymbol{X}\boldsymbol{W}\Big((\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1} - (\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})^{-1}\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})^{-1}\Big) + \boldsymbol{Y}\boldsymbol{W}\Big((\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})^{-1} - (\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}\Big)$. (22)
AIRM	$\nabla_{\boldsymbol{W}}\Big(\delta_R^2(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})\Big) = 4\Big(\boldsymbol{X}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})^{-1} - \boldsymbol{Y}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}\Big) \times \log\Big(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}\Big)$. (23)

The Jacobian is used to perform Newton-type optimization on the Grassmannian.

SPD matrices is reflected by a measure of similarity on the low-dimensional SPD manifold. In particular, we make use of the AIRM, the Stein divergence, or the Jeffrey divergence to encode similarity between SPD matrices. This lets us write a loss function of the form

$$L(\boldsymbol{W}) = \sum_{\substack{i,j=1\\ j\neq i}}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j)\delta\big(\boldsymbol{W}^T\boldsymbol{X}_i\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{X}_j\boldsymbol{W}\big), \tag{17}$$

where p is the number of training samples and δ is δ_R^2, δ_S^2 or δ_J. To perform dimensionality reduction, our idea is to minimize $L(\boldsymbol{W})$ by enforcing a unitary constraint on $\boldsymbol{W}$. That is,

$$\begin{aligned} &\min_{\boldsymbol{W}\in\mathbb{R}^{n\times m}} \; L(\boldsymbol{W}) \\ &\text{s.t. } \boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_m, \end{aligned} \tag{18}$$

As discussed in Section 2.3, (18) is an optimization problem on a Grassmann manifold, and can thus be solved by Newton-type methods on the Grassmannian $\mathcal{G}(m,n)$. To this end, we need to compute the Jacobian of $\delta(\cdot,\cdot)$ with respect to $\boldsymbol{W}$. For the S divergence, this Jacobian matrix, denoted by $\nabla_{\boldsymbol{W}}(\cdot)$ hereafter, can be obtained by noting that (see Eq. 53 in [35])

$$\nabla_{\boldsymbol{W}}\log\det\big(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}\big) = 2\boldsymbol{X}\boldsymbol{W}\big(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}\big)^{-1}. \tag{19}$$

This allows us to express the Jacobian of the Stein divergence as Eq. (21) in Table 1. For the J divergence, the Jacobian can be obtained by noting that (see Eq. 126 in [35])

$$\begin{aligned} \nabla_{\boldsymbol{W}}\Big(\mathrm{Tr}\Big(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}\Big)\Big) &= 2\boldsymbol{X}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1} \\ &- 2\boldsymbol{Y}\boldsymbol{W}(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})(\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W})^{-1}, \end{aligned} \tag{20}$$

which leads to Eq. (22) in Table 1.

Putting all the details together, our supervised DR method for SPD matrices is summarized in Algorithm 1, where $\tau(\boldsymbol{H},\boldsymbol{W}_0,\boldsymbol{W}_1)$ denotes the parallel transport of tangent vector $\boldsymbol{H}$ from $\boldsymbol{W}_0$ to $\boldsymbol{W}_1$ (see Section 2.3 for details).

Remark 1. To avoid degeneracies and ensure that the resulting embedding forms a valid SPD manifold, i.e., $\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W} \succ 0,\ \forall \boldsymbol{X}\in\mathcal{S}_{++}^n$, we need $\boldsymbol{W}$ to have full rank. Here, we enforce this requirement by imposing the unitary constraints $\boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_m$. Note that, with the affine invariance property, this entails no loss of generality. Indeed, any full rank matrix $\tilde{\boldsymbol{W}}$ can be expressed as $\boldsymbol{W}\boldsymbol{M}$, with $\boldsymbol{W}$ an orthonormal matrix and $\boldsymbol{M}\in\mathrm{GL}(m)$. The affine invariance property of the metric therefore guarantees that

$$L(\tilde{\boldsymbol{W}}) = L(\boldsymbol{W}\boldsymbol{M}) = L(\boldsymbol{W}).$$

Algorithm 1. Supervised SPD DR

Input:
A set of SPD matrices $\{\boldsymbol{X}_i\}_{i=1}^p,\ \boldsymbol{X}_i \in \mathcal{S}_{++}^n$.
The corresponding labels $\{y_i\}_{i=1}^p,\ y_i\in\{1,2,\ldots,C\}$.
The dimensionality m of the induced manifold.
Output:
The mapping $\boldsymbol{W}\in\mathcal{G}(m,n)$
Generate $a(\boldsymbol{X}_i,\boldsymbol{X}_j)$ using (16)
$\boldsymbol{W}_{old} \leftarrow \mathbf{I}_{n\times m}$ (i.e., the truncated identity matrix)
$\boldsymbol{W} \leftarrow \boldsymbol{W}_{old}$
$\boldsymbol{H}_{old} \leftarrow \mathbf{0}$
repeat
  $\boldsymbol{H} \leftarrow -\mathrm{grad}_{\boldsymbol{W}}L(\boldsymbol{W}) + \eta\tau(\boldsymbol{H}_{old},\boldsymbol{W}_{old},\boldsymbol{W})$
  Line search along the geodesic starting from $\boldsymbol{W}$ in the direction $\boldsymbol{H}$ to find $\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\mathrm{argmin}}\ L(\boldsymbol{W})$
  $\boldsymbol{H}_{old} \leftarrow \boldsymbol{H}$
  $\boldsymbol{W}_{old} \leftarrow \boldsymbol{W}$
  $\boldsymbol{W} \leftarrow \boldsymbol{W}^*$
until convergence

3.2 Unsupervised Dimensionality Reduction

We now turn to the scenario where we do *not* have access to the labels of the training samples. In other words, our training data only consists of a set of SPD matrices $\{\boldsymbol{X}_i\}_{i=1}^p,\ \boldsymbol{X}_i\in\mathcal{S}_{++}^n$. To tackle this unsupervised DR scenario, we draw inspiration from algorithms such as PCA and MVU [18]. These algorithms search for a low-dimensional latent space where the points have maximum variance, i.e., collectively maximize the distance to their mean.

Here, we follow the same intuition, but with the goal of mapping high-dimensional SPD matrices to lower-dimensional ones. To this end, we express unsupervised DR on SPD manifolds as the optimization problem

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}\in\mathbb{R}^{n\times m}}{\arg\max} \sum_{i=1}^{p} \delta\big(\boldsymbol{W}^T\boldsymbol{X}_i\boldsymbol{W}, \boldsymbol{W}^T\boldsymbol{M}\boldsymbol{W}\big) \quad \text{s.t. } \boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_m, \tag{24}$$

with $\boldsymbol{M}$ being the mean of $\{\boldsymbol{X}_i\}_{i=1}^p$ with respect to the metric δ. As in the supervised case, and as discussed in more details in Section 2.3, (24) corresponds to an optimization problem in the Grassmann manifold. We therefore again opt for a Newton-type method on the Grassmannian to (approximately) solve it. Note that the gradient of the objective function has essentially the same form as in the supervised case, and can thus be easily obtained from Eqs. (21), (22) and (23) for the Stein divergence, J-divergence and AIRM, respectively.

As mentioned above, (24) depends on the mean of the training samples. Since these samples lie on a manifold, special care must be taken to compute their means. In particular, we make use of the Fréchet formulation to obtain $\boldsymbol{M}$. This can be expressed as

$$\boldsymbol{M}^* \triangleq \underset{\boldsymbol{M}\in\mathcal{S}_{++}^n}{\arg\min} \sum_{i=1}^{p} \delta^2(\boldsymbol{X}_i, \boldsymbol{M}). \tag{25}$$

For the AIRM, this is equivalent to computing the Riemannian (Karcher) mean by exploiting the exponential and logarithm maps [2]. For the Stein metric, we make use of the iterative Convex Concave Procedure (CCCP) introduced in [19]. For the Jeffrey divergence, we show below that, unlike the AIRM and the Stein divergence, the Fréchet mean can be computed analytically.

Theorem 1. *The Fréchet mean of a set of points $\{\boldsymbol{X}_i\}_{i=1}^p, \boldsymbol{X}_i \in \mathcal{S}_{++}^n$, based on the Jeffrey metric, i.e., the minimizer of (25) for $\delta^2(\cdot,\cdot) = \delta_J$, is given by*

$$\boldsymbol{M}^* = \mathbf{L}^{-1/2}\big(\mathbf{L}^{1/2}\boldsymbol{\Gamma}\mathbf{L}^{1/2}\big)^{1/2}\mathbf{L}^{-1/2} \tag{26}$$

with $\mathbf{L} = \sum_{i=1}^p \boldsymbol{X}_i^{-1}$ and $\boldsymbol{\Gamma} = \sum_{i=1}^p \boldsymbol{X}_i$.

Proof. To prove this theorem, let us first recall that, for $\boldsymbol{A} \succ 0$ and $\boldsymbol{B} \succeq 0$, a quadratic equation of the form $\boldsymbol{X}\boldsymbol{A}\boldsymbol{X} = \boldsymbol{B}$, called a *Riccati* equation, has only one positive definite solution of the form [20]

$$\boldsymbol{X} = \boldsymbol{A}^{-1/2}\big(\boldsymbol{A}^{1/2}\boldsymbol{B}\boldsymbol{A}^{1/2}\big)^{1/2}\boldsymbol{A}^{-1/2}. \tag{27}$$

According to (25), and by making use of the J divergence, the Fréchet mean must satisfy

$$\frac{\partial \sum_{i=1}^p \delta_J(\boldsymbol{X}_i, \boldsymbol{M})}{\partial \boldsymbol{M}} = 0. \tag{28}$$

Given that

$$\frac{\partial \mathrm{Tr}(\boldsymbol{X}\boldsymbol{M}^{-1})}{\partial \boldsymbol{M}} = \boldsymbol{M}^{-1}\boldsymbol{X}\boldsymbol{M}^{-1},$$

we have

$$\frac{\partial \sum_{i=1}^p \delta_J(\boldsymbol{X}_i, \boldsymbol{M})}{\partial \boldsymbol{M}} = \sum_{i=1}^{p} \boldsymbol{X}_i^{-1} - \sum_{i=1}^{p} \boldsymbol{M}^{-1}\boldsymbol{X}_i\boldsymbol{M}^{-1} = 0$$
$$\Leftrightarrow \boldsymbol{M}\sum_{i=1}^{p} \boldsymbol{X}_i^{-1}\boldsymbol{M} = \sum_{i=1}^{p} \boldsymbol{X}_i,$$

which is a *Riccati* equation with a unique and closed form solution. A slightly different proof is also provided in [17]. □

Remark 2. There is a subtle difference between PCA in Euclidean space and the solution developed here. More specifically, unlike PCA in Euclidean space $\boldsymbol{W}^T\boldsymbol{M}\boldsymbol{W}$ does not necessarily represent the mean of the transformed data in $\mathcal{S}_{++}^m$. That is,

$$\boldsymbol{W}^T\boldsymbol{M}\boldsymbol{W} \neq \underset{\boldsymbol{F}\in\mathcal{S}_{++}^m}{\arg\min} \sum_{i=1}^{p} \delta^2(\boldsymbol{W}^T\boldsymbol{X}_i\boldsymbol{W}, \boldsymbol{F})$$

in general.

3.3 DR with the Log-Euclidean Metric

In the previous parts, we have developed DR methods on $\mathcal{S}_{++}^n$ based on the AIRM and two Bregman divergences. Another widely used metric to compare SPD matrices is the log-Euclidean metric defined as

$$\delta_{lE}(\boldsymbol{X}, \boldsymbol{Y}) = \|\log(\boldsymbol{X}) - \log(\boldsymbol{Y})\|_F, \tag{29}$$

where $\log(\cdot)$ denotes the matrix principal logarithm. This metric is indeed a true Riemannian metric (for a zero-curvature manifold) and can be understood as a metric over the flat space that identifies the Lie algebra of an SPD manifold. Below, we develop a supervised DR method on SPD manifolds similar to the one in Section 3.2, but using the log-Euclidean metric. The adaptation to the unsupervised scenario introduced in Section 3.2 can easily be derived in a similar manner.

With the log-Euclidean metric, (18) can be rewritten as

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n\times m}} \sum_{i,j=1}^{p} a(\boldsymbol{X}_i, \boldsymbol{X}_j)\|\log(\boldsymbol{W}^T\boldsymbol{X}_i\boldsymbol{W}) - \log(\boldsymbol{W}^T\boldsymbol{X}_j\boldsymbol{W})\|_F^2, \quad \text{s.t. } \boldsymbol{W}^T\boldsymbol{W} = \mathbf{I}_m. \tag{30}$$

A difficulty in tackling (30) arises from the fact that an analytic form for the gradient of $\|\log(\boldsymbol{W}^T\boldsymbol{X}_i\boldsymbol{W}) - \log(\boldsymbol{W}^T\boldsymbol{X}_j\boldsymbol{W})\|_F^2$ with respect to $\boldsymbol{W}$ is not known [36]. To overcome this limitation, we introduce an approximation of $\log(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})$. This approximation relies on the following lemma.

Lemma 1. *The term $\log(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})$ can be approximated as $\boldsymbol{W}^T\log(\boldsymbol{X})\boldsymbol{W}$.*

Proof. Note that the Taylor expansion of $\log(\mathbf{I}_n - \boldsymbol{A})$ is given by [37]

$$\log(\mathbf{I}_n - \boldsymbol{A}) = -\boldsymbol{A} - \frac{1}{2}\boldsymbol{A}^2 - \frac{1}{3}\boldsymbol{A}^3 - \cdots. \tag{31}$$

Therefore, we can write

$$\begin{aligned}\log(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}) &= \log(\mathbf{I}_n - (\mathbf{I}_n - \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})) \\ &\approx -(\mathbf{I}_n - \boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W})) = -\boldsymbol{W}^T(\mathbf{I}_n - \boldsymbol{X})\boldsymbol{W} \\ &\approx \boldsymbol{W}^T\log(\boldsymbol{X})\boldsymbol{W},\end{aligned}$$

where both the second and third lines make use of the first order Taylor approximation from Eq. (31). □

From the lemma above, we can cast (30) into the optimization problem

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n\times m}} \sum_{i,j=1}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j)\left\|\boldsymbol{W}^T\log(\boldsymbol{X}_i)\boldsymbol{W}-\boldsymbol{W}^T\log(\boldsymbol{X}_j)\boldsymbol{W}\right\|_F^2, \quad \text{s.t. } \boldsymbol{W}^T\boldsymbol{W}=\mathbf{I}_m. \tag{32}$$

The objective function of (32) can then be written as

$$\sum_{i,j=1}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j)\left\|\boldsymbol{W}^T\log(\boldsymbol{X}_i)\boldsymbol{W}-\boldsymbol{W}^T\log(\boldsymbol{X}_j)\boldsymbol{W}\right\|_F^2 = \mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{F}(\boldsymbol{W})\boldsymbol{W}),$$

with

$$\boldsymbol{F}(\boldsymbol{W}) = \sum_{i,j=1}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j)(\log(\boldsymbol{X}_i)-\log(\boldsymbol{X}_j))\boldsymbol{W}\boldsymbol{W}^T \times(\log(\boldsymbol{X}_i)-\log(\boldsymbol{X}_j)), \tag{33}$$

which yields the optimization problem

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n\times m}} \mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{F}(\boldsymbol{W})\boldsymbol{W}), \quad \text{s.t. } \boldsymbol{W}^T\boldsymbol{W}=\mathbf{I}_m. \tag{34}$$

Note that $\mathrm{Tr}\big(\boldsymbol{W}^T\boldsymbol{F}(\boldsymbol{W})\boldsymbol{W}\big)$ is invariant to the action of the orthogonal group, i.e., replacing $\boldsymbol{W}$ with $\boldsymbol{W}\boldsymbol{R}$ for $\boldsymbol{R}\in\mathrm{O}(m)$ does not change the value of the trace. As such, in principle, (34) is a problem on $\mathcal{G}(m,n)$ and can be optimized in a similar manner as discussed before. In particular, to perform Newton-type methods on the Grassmannian, the required gradient is given by

$$\nabla_{\boldsymbol{W}}\mathrm{Tr}(\boldsymbol{W}^T\boldsymbol{F}(\boldsymbol{W})\boldsymbol{W}) = 4\sum_{i,j=1}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j) \times(\log(\boldsymbol{X}_i)-\log(\boldsymbol{X}_j))\boldsymbol{W}\boldsymbol{W}^T(\log(\boldsymbol{X}_i)-\log(\boldsymbol{X}_j))\boldsymbol{W}.$$

While optimization on the Grassmannian can indeed be employed to solve (34), here, we propose a faster alternative which relies on eigen-decomposition. To this end, we follow an iterative two-stage procedure. First, we fix $\boldsymbol{F}(\boldsymbol{W})$ (i.e., assume that it does not depend on $\boldsymbol{W}$), and compute the solution of the resulting approximation of (34), which can be achieved by computing the m smallest eigenvectors of $\boldsymbol{F}(\boldsymbol{W})$ [27]. Given the new $\boldsymbol{W}$, we update $\boldsymbol{F}(\boldsymbol{W})$, and iterate. The pseudo-code of this procedure is given in Algorithm 2.

Algorithm 2. Iterative Eigen-Decomposition Solver for Log-Euclidean-Based Supervised DR:

Input:
A set of SPD matrices $\{\boldsymbol{X}_i\}_{i=1}^p,\ \boldsymbol{X}_i\in\mathcal{S}_{++}^n$
The corresponding labels $\{y_i\}_{i=1}^p,\ y_i\in\{1,2,\ldots,C\}$
The dimensionality m of the induced manifold
Output:
The mapping $\boldsymbol{W}\in\mathcal{G}(m,n)$
Generate $a(\boldsymbol{X}_i,\boldsymbol{X}_j)$ using Eq. (16)
$\boldsymbol{W}\leftarrow\mathbf{I}_{n\times m}$ (i.e., the truncated identity matrix)
repeat
  Compute $\boldsymbol{F}(\boldsymbol{W})$ using Eq. (33)
  $\boldsymbol{W}\leftarrow m$ smallest eigenvectors of $\boldsymbol{F}(\boldsymbol{W})$.
until convergence

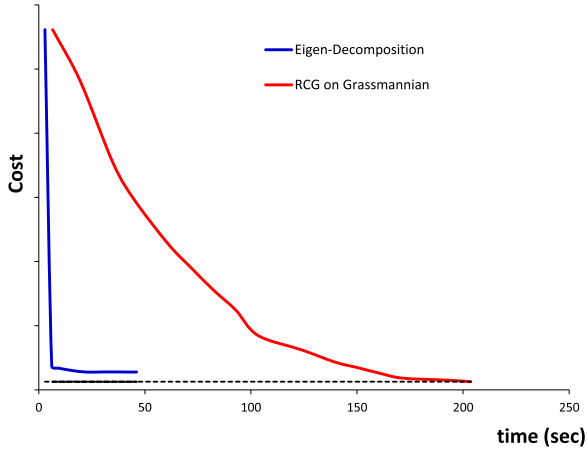

Fig. 3. Convergence behavior of the proposed eigen-decomposition-based solver compared to that of conjugate gradient optimization on the Grassmann manifold.

Fig. 3 compares the speed and convergence behavior of our iterative eigen-decomposition-based solution against the RCG method on the Grassmannian. This figure was computed using the MOCAP dataset (see Section 6.1.2 for details). First, note that the RCG method yields a lower error. This is due to the fact that, in the eigen-decomposition-based method, we approximate the solution by fixing $\boldsymbol{F}(\boldsymbol{W})$ in each iteration. Nevertheless, the eigen-decomposition-based solver converges much faster than RCG. In fact, our experiments show that the eigen-decomposition-based solver is typically 10 times faster than RCG on the Grassmannian, which, we believe, justifies its usage. This suggests a hybrid strategy where one initializes the RCG method with our eigen-decomposition-based solution.

Remark 3. Since, in (32), the log operation maps the matrices to Euclidean space, one might wonder if we should not apply a traditional vector-based DR approach to the resulting space. Note, however, that our goal is to go from $\mathcal{S}_{++}^n$ to $\mathcal{S}_{++}^m$. Therefore, optimizing a projection between the corresponding Euclidean spaces would translate to an optimization problem on $\mathcal{G}(\frac{m\times(m+1)}{2},\frac{n\times(n+1)}{2})$. By contrast, our symmetric formulation results in an optimization problem on $\mathcal{G}(m,n)$, which is clearly less expensive.

Remark 4. From a purely geometrical point of view, we believe that the solutions developed using the AIRM, the Stein divergence and the Jeffrey divergence are more attractive. In particular, these solutions model the nonlinear geometry of the SPD manifold, while the log-Euclidean metric flattens it. Furthermore, in contrast with the log-Euclidean metric, the AIRM, Stein and Jeffrey divergences are invariant to affine transformations. We acknowledge, however, that the log-Euclidean metric has been shown to be a useful substitute to the AIRM in several applications (e.g., [3], [10], [38]).

Remark 5. Since the Frobenius norm also belongs to the family of Bregman divergences (with $\zeta(\boldsymbol{X})=\|\boldsymbol{X}\|_F^2$), one could derive supervised/unsupervised DR formulations using $\|\cdot\|_F^2$ as similarity measure. For example, in the supervised case, this would translate to solving

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n\times m}} \sum_{i,j=1}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j)\left\|\boldsymbol{W}^T\boldsymbol{X}_i\boldsymbol{W}-\boldsymbol{W}^T\boldsymbol{X}_j\boldsymbol{W}\right\|_F^2, \quad \text{s.t. } \boldsymbol{W}^T\boldsymbol{W}=\mathbf{I}_m . \tag{35}$$

This can easily be rewritten in the form of (34), but where now $\boldsymbol{F}(\boldsymbol{W})$ can be expressed as

$$\boldsymbol{F}(\boldsymbol{W})=\sum_{i,j=1}^{p} a(\boldsymbol{X}_i,\boldsymbol{X}_j)(\boldsymbol{X}_i-\boldsymbol{X}_j)\boldsymbol{W}\boldsymbol{W}^T(\boldsymbol{X}_i-\boldsymbol{X}_j).$$

Therefore, our previous eigen-decomposition-based solution directly applies here.

4 Further Discussions

In this section, we discuss several points regarding our DR framework. In particular, we look into the computational complexity of the developed methods and then discuss the use of SPD matrices as image/video descriptors [4].

4.1 Computational Complexity

To obtain the solution of (18) or (24), an iterative Riemannian optimization technique is employed. The complexity of each iteration of the Riemannian optimization technique depends on the computational cost of three major steps, namely, evaluating the objective, computing the gradient and performing RCG on $\mathcal{G}(m,n)$. Below, we discuss the complexity of each of the aforementioned major steps separately.

4.1.1 Complexity of the Objective Function

Computing the quadratic form $\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W}$ for $\boldsymbol{X}\in\mathcal{S}_{++}^n, \boldsymbol{W}\in\mathcal{G}(m,n)$ requires n^2m+nm^2 flops. We also note that computing $\delta_R^2(\boldsymbol{A},\boldsymbol{B})$, $\delta_S^2(\boldsymbol{A},\boldsymbol{B})$ and $\delta_J(\boldsymbol{A},\boldsymbol{B})$ for $\boldsymbol{A},\boldsymbol{B}\in\mathcal{S}_{++}^m$ needs $4m^3$, m^3 and $8/3m^3$ flops, respectively (see [19]). As such,

- Evaluating the loss in (18) takes $n^2mp+nm^2p+4m^3p^2$, $n^2mp+nm^2p+m^3p^2$ and $n^2mp+nm^2p+8/3m^3p^2$ flops for the AIRM, the Stein divergence and the Jeffrey divergence, respectively.
- Evaluating the loss in (24) needs $n^2mp+nm^2p+4m^3p$, $n^2mp+nm^2p+m^3p$ and $n^2mp+nm^2p+8/3m^3p$ flops for the AIRM, the Stein divergence and the Jeffrey divergence, respectively.

4.1.2 Complexity of the Gradient

The complexity of inverting a matrix of size $m\times m$ is $O(m^{2.376})$ using the Coppersmith-Winograd algorithm. The complexity of computing the logarithm of an $m\times m$ matrix is $O(2m^3)$, since an eigen-decomposition followed by a matrix multiplication is required. As such, computing $\nabla_{\boldsymbol{W}}(\delta_S^2(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W}))$ according to Eq. (21) takes $2n^2m+5nm^2+3m^{2.376}$. As for the J-divergence, computing $\nabla_{\boldsymbol{W}}(\delta_J(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W}))$ according to Eq. (22) takes $2n^2m+4nm^2+2m^{2.376}+2m^3$. Finally, computing $\nabla_{\boldsymbol{W}}(\delta_R^2(\boldsymbol{W}^T\boldsymbol{X}\boldsymbol{W},\boldsymbol{W}^T\boldsymbol{Y}\boldsymbol{W}))$ according to Eq. (23) takes $2n^2m+5nm^2+2m^{2.376}+3m^3$.

To give the reader an idea about the computational complexity of this step, we report the overall time required to compute the gradients according to Table 1 over 1,000 trials. Table 2 presents the measured running-times for various dimensionalities on an eight-core machine using Matlab2016. Note that both divergences are 5-10× faster than the AIRM. For smaller values of m, the Jeffrey divergence is marginally faster than the Stein divergence. Increasing m makes the influence of the term $2m^3$ in the Jeffrey divergence noticeable, leading to a higher computation load compared to the Stein divergence.

TABLE 2
Overall Running-Times for Computing the Gradients According to Table 1 over 1,000 Trials

n	m	Stein	Jeffrey	AIRM
256	64	0.1s	0.1s	0.7s
	128	0.3s	0.3s	2.9s
512	64	0.2s	1.5s	6.8s
	128	0.5s	0.4s	3.0s
	256	1.2s	1.5s	14.1s
1024	128	0.8s	0.6s	3.1s
	256	2.5s	2.2s	14.5s
	512	8.5s	11.9s	77.0s

4.1.3 Complexity of the Optimization

The complexity of RCG on the Grassmannian depends mainly on two operations, namely, computing the Riemannian gradient from the Jacobians and mapping a tangent vector back to the manifold using a retraction. As mentioned in Section 2.3, to compute the Riemannian gradient on the Grassmannian, one needs to project the Jacobian $\nabla_{\boldsymbol{W}}(\cdot)$ according to Eq. (11). This operation requires $nm(n+m)$ flops. For a tangent vector $\xi\in T_{\boldsymbol{W}}$, we use the retraction (see Chapter 4 of [29])

$$\Upsilon_{\boldsymbol{W}}(\xi)=\mathrm{qf}(\boldsymbol{W}+\xi). \tag{36}$$

Here, $\mathrm{qf}(\boldsymbol{A})$ is the $\boldsymbol{Q}$ factor of the QR decomposition [39]. More specifically, let $\mathbb{R}^{n\times m}\ni\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{R}$ with $\boldsymbol{Q}^T\boldsymbol{Q}=\mathbf{I}_m$ and $\boldsymbol{R}$ being an upper triangular $m\times m$ matrix with nonzero diagonal elements. Then $\boldsymbol{Q}=\mathrm{qf}(\boldsymbol{A})$. The complexity of the QR decomposition using the Householder algorithm is $O\big(2m^2(n-m/3)\big)$ [39].

4.2 Region Covariance Descriptors

When it comes to the SPD matrices used in our experiments, we exploited Region Covariance Matrices (RCMs) [4] as image descriptors. Here, we discuss some interesting properties of our algorithm when applied to these specific SPD matrices.

There are several reasons why RCMs are attractive to represent images and videos. First, RCMs provide a natural way to fuse various feature types. Second, they help reducing the impact of noisy samples in a region via their inherent averaging operation. Third, RCMs are independent of the size of the region, and can therefore easily be utilized to compare regions of different sizes. Finally, RCMs can be efficiently computed using integral images [6], [12].

Let I be a $W\times H$ image, and $\mathbb{O}=\{\boldsymbol{o}_i\}_{i=1}^r$, $\boldsymbol{o}_i\in\mathbb{R}^n$, be a set of r observations extracted from I, e.g., $\boldsymbol{o}_i$ concatenates

intensity values, gradients along the horizontal and vertical directions, filter responses,... for image pixel i. Let $\mu = \frac{1}{r}\sum_{i=1}^{r} \boldsymbol{o}_i$ be the mean value of the observations. Then, image I can be represented by the $n \times n$ RCM

$$\boldsymbol{C}_I = \frac{1}{r-1}\sum_{i=1}^{r}(\boldsymbol{o}_i - \mu)(\boldsymbol{o}_i - \mu)^T = \boldsymbol{O}\boldsymbol{J}\boldsymbol{J}^T\boldsymbol{O}^T, \qquad (37)$$

where $\boldsymbol{J} = r^{-3/2}(r\mathbf{I}_r - 1_{r\times r})$. To have a valid RCM, $r \geq n$, otherwise $\boldsymbol{C}_I$ would have zero eigenvalues, which would make δ_S^2, δ_J and δ_R^2 indefinite.

After learning the projection $\boldsymbol{W}$, the low-dimensional representation of image I is given by $\boldsymbol{W}^T\boldsymbol{O}\boldsymbol{J}\boldsymbol{J}^T\boldsymbol{O}^T\boldsymbol{W}$. This reveals two interesting properties of our learning scheme. **1)** The resulting representation can also be thought of as an RCM with $\boldsymbol{W}^T\boldsymbol{O}$ as a set of low-dimensional observations. Hence, in our framework, we can create a valid $\mathcal{S}_{++}^m$ manifold with only m observations instead of at least n in the original input space. This is not the case for other algorithms, which require having matrices on $\mathcal{S}_{++}^n$ as input. **2)** Applying $\boldsymbol{W}$ directly to the set of observations reduces the computation time of creating the final RCM on $\mathcal{S}_{++}^m$. This is due to the fact that the computational complexity of computing an RCM is quadratic in the dimensionality of the features.

Nowadays, deep networks (e.g., Convolutional Neural Network (CNN)) are the methods of choice to classify images. Nevertheless, RCMs are still considered rich and powerful image descriptors in various applications including texture classification and inference from skeletal data. Interestingly, several recent studies benefit from a combination of RCMs and deep architectures to deliver state-of-the-art solutions [15], [40], [41]. In particular, the recent work of Wang et al. [15] shows that high-dimensional RCMs built from CNN features easily outperform many deep Euclidean representations. This, however, results in high-dimensional RCMs, and thus further justifies our approach to reducing their dimensionality.

5 Related Work

In this section, we review the methods that have exploited notions of Riemannian geometry for DR. In contrast with our approach that goes from one high-dimensional SPD manifold to a lower-dimensional manifold, most of the literature has focused on going from a manifold to Euclidean space.

In this context, a popular approach consists of flattening the manifold via its tangent space. The best-known example of such an approach is Principal Geodesic Analysis (PGA) [42], [43]. PGA and its variants, such as [44], [45], [46], have been successfully employed for various applications, such as analyzing vertebrae outlines [47] and motion capture data [44]. PGA can be understood as a generalization of PCA to Riemannian manifolds. To this end, the widely-used formulation proposed in [43] identifies the tangent space whose corresponding subspace maximizes the variability of the data on the manifold. PGA, however, is equivalent to flattening the Riemannian manifold by taking its tangent space at the Karcher, or Fréchet, mean of the data. As such, it does not fully exploit the structure of the manifold. Furthermore, PGA, as PCA, cannot exploit the availability of class labels, and may therefore be suboptimal for classification.

Another recent popular trend consists of embedding the manifold to an RKHS to perform DR. In particular, [8] relied on kernel PCA and [10] on kernel Partial Least Squares (kPLS) and kernel Linear Discriminant Analysis (LDA) to achieve this goal. Embedding the manifold to an RKHS inherently requires a p.d. kernel. While significant progress has been made in identifying p.d. kernels on Riemannian manifolds [8], [48], [49], our knowledge is still limited in this regard. For example and in the case of SPD manifolds, the kernel employed in [10] is a linear kernel on the identity tangent space of $\mathcal{S}_{++}^n$. In [8], the best performing kernel corresponds to the Gaussian kernel defined on the identity tangent space of $\mathcal{S}_{++}^n$. Therefore, in a very strict sense, the true structure of the manifold is not used by either of these works. As a matter of fact, Feragen et al. recently showed that Gaussian kernels obtained using the geodesic distance on curved Riemannian manifolds, such as the SPD manifold, are not valid p.d. kernels [49].

Different from the methods that flatten the manifold, via either a tangent space, or an RKHS, [50] directly employs notions of Riemannian geometry to perform nonlinear DR. In particular, [50] extends several nonlinear DR techniques, such as Locally Linear Embedding (LLE), Hessian LLE and Laplacian Eigenmaps, to their Riemannian counterparts. Take for example the case of LLE [23]. Given a set of vectors $\{\boldsymbol{x}_i\}_{i=1}^p, \boldsymbol{x}_i \in \mathbb{R}^D$, the LLE algorithm determines a weight matrix $\boldsymbol{W} \in \mathbb{R}^{p\times p}$ to minimize a notion of reconstruction error on $\{\boldsymbol{x}_i\}_{i=1}^p$. Once the weight matrix $\boldsymbol{W}$ is determined, the algorithm embeds $\{\boldsymbol{x}_i\}_{i=1}^p$ into a lower dimensional space $\mathbb{R}^d, d < D$, where the neighboring properties of $\{\boldsymbol{x}_i\}_{i=1}^p$ are preserved. The neighboring properties are encoded by $\boldsymbol{W}$ and the embedding takes the form of an eigen-decomposition in the end. As shown in [50], the construction of $\boldsymbol{W}$ can be generalized to the case of an arbitrary Riemannian manifold $\mathcal{M}$ by using the logarithm map. Hence, for a given set of points on $\mathcal{M}$, an embedding from $\mathcal{M} \rightarrow \mathbb{R}^d$ can be obtained once $\boldsymbol{W}$ is appropriately constructed. In [50], the authors showed that the embedded data was more discriminative than the original one using several clustering problems on $\mathcal{M}$. In principle, the Riemannian extension of LLE (and of the other nonlinear DR algorithms discussed in [50]) can also be applied to classification problems. However, they are limited to the transductive setting since they do not define any parametric mapping to the low-dimensional space.

Recently, a few techniques have studied the case of mapping between two manifolds of different dimensions [31], [32], [36], [51], [52]. In [51], a mapping between covariance matrices of different dimensions was learnt, but by ignoring the Riemannian geometry of the SPD manifold. More recently, and probably inspired by our preliminary study [53], the bilinear form was employed to perform DR on the SPD manifold and on the Grassmannian by exploiting notions of Riemannian geometry [31], [32], [36]. In particular, [31] introduced the idea of learning a log-Euclidean metric, which is related to our log-Euclidean-based supervised DR approach. This work formulates DR as the problem of finding a positive semi-definite matrix $\boldsymbol{Q} \in \mathrm{Sym}(n)$ that maximizes the discriminative power of pairs of samples according to

$$\delta_{ij}(\boldsymbol{Q}) = \mathrm{Tr}(\boldsymbol{Q}(\log(\boldsymbol{X}_i) - \log(\boldsymbol{X}_j))((\boldsymbol{X}_i) - \log(\boldsymbol{X}_j))).$$

In particular, $\boldsymbol{Q}$ was forced to have rank m, and thus identifies a low-dimensional latent space. Obtaining $\boldsymbol{Q}$ was then formulated as a *log-det* problem [31]. A similar idea was employed to perform dimensionality reduction on the Grassmannian by noting that a point on the Grassmannian can be represented by a symmetric and idempotent matrix [32]. In [36], Yger and Sugiyama proposed to learn a metric on SPD manifolds, where the goal is to find $\boldsymbol{Q} \in \mathcal{S}_{++}^n$ such that the points transformed by $f(\boldsymbol{X}) = \boldsymbol{Q}\boldsymbol{X}\boldsymbol{Q}$ are more discriminative. In contrast to our work, in [36], a mapping from $\mathcal{S}_{++}^n \rightarrow \mathcal{S}_{++}^n$ was learned. Note also that our work covers various metrics and both supervised and unsupervised scenarios, while the algorithm in [36] was devised for supervised learning and only considers the log-Euclidean metric.

Finally, concepts of Riemannian geometry have also been exploited in the context of DR in Euclidean space. For instance, Lin and Zha [54] exploited the idea that the input (Euclidean) data lies on a low-dimensional Riemannian manifold. Recently, Cunningham and Ghahramani [30] revisited linear DR techniques and analyzed them using the geometry of Stiefel manifolds.

6 Empirical Evaluation

Below, we evaluate our different SPD-based DR methods on several problems. We first consider the supervised scenario and present results on image and video classification tasks. We then turn to evaluating our unsupervised techniques for clustering on SPD manifolds. In all our experiments, the dimensionality of the low-dimensional SPD manifold was determined by cross-validation.

6.1 Image/Video Classification

The supervised SPD-DR algorithm introduced in Section 3.1 allows us to obtain a low-dimensional, more discriminative SPD manifold from a high-dimensional one. Many different classifiers can then be used to categorize the data on this new manifold. In our experiments, we make use of two such classifiers. First, we employ a simple nearest neighbor classifier based on the manifold metric (AIRM, S or J divergence). This simple classifier clearly evidences the benefits of mapping the original Riemannian structure to a lower-dimensional one. Second, we make use of the Riemannian sparse coding algorithm of [5]. This algorithm exploits the notion of sparse coding to represent a query SPD matrix using a codebook of SPD matrices. In all our experiments, we formed the codebook purely from the training data, i.e., no dictionary learning was employed. Note that this algorithm relies on a kernel derived from either the S divergence or the log-Euclidean metric. We refer to the different algorithms evaluated in our experiments as:

NN-AIRM: AIRM-based Nearest Neighbor classifier on the high-dimensional SPD manifold.
NN-S: S divergence-based Nearest Neighbor classifier on the high-dimensional SPD manifold.
NN-J: J divergence-based Nearest Neighbor classifier on the high-dimensional SPD manifold.
NN-lE: log-Euclidean-based Nearest Neighbor classifier on the high-dimensional SPD manifold.
NN-AIRM-DR: AIRM-based Nearest Neighbor classifier on the low-dimensional SPD manifold obtained with our approach.

Fig. 4. Samples from the UIUC material dataset [56].

NN-S-DR: S divergence-based Nearest Neighbor classifier on the low-dimensional SPD manifold obtained with our approach.
NN-J-DR: J divergence-based Nearest Neighbor classifier on the low-dimensional SPD manifold obtained with our approach.
NN-lE-DR: log-Euclidean-based Nearest Neighbor classifier on the low-dimensional SPD manifold obtained with our approach.
kSC-S: kernel sparse coding [55] using the S divergence on the high-dimensional SPD manifold.
kSC-lE: kernel sparse coding using the log-Euclidean metric on the high-dimensional SPD manifold.
kSC-S-DR: kernel sparse coding using the S divergence on the low-dimensional SPD manifold obtained with our approach.
kSC-lE-DR: kernel sparse coding using the log-Euclidean metric on the low-dimensional SPD manifold obtained with our approach.

In addition to these methods, we also provide the results of the PLS-based Covariance Discriminant Learning (CDL) technique of [10] and of the state-of-the-art baselines of each specific dataset.

In practice, to define the affinity function (see Section 3.1), we set ν_w to the minimum number of points in each class and, to balance the influence of $g_w(\cdot,\cdot)$ and $g_b(\cdot,\cdot)$, choose $\nu_b \leq \nu_w$, with the specific value found by cross-validation.

6.1.1 Material Categorization

For the task of material categorization, we used the UIUC dataset [56]. The UIUC material dataset contains 18 subcategories of materials taken in the wild from four general categories (see Fig. 4): *bark*, *fabric*, *construction materials*, and *outer coat of animals*. Each subcategory has 12 images taken at various scales. Following standard practice, half of the images from each subcategory was randomly chosen as training data, and the rest was used for testing. We report the average accuracy over 10 different random partitions.

Small RCMs, such as those used for texture recognition in [4], are hopeless here due to the complexity of the task. Recently, SIFT features [57] have been shown to be robust and discriminative for material classification [56]. Therefore, we constructed RCMs of size 155×155 using 128 dimensional SIFT features (from grayscale images) and 27 dimensional color descriptors. To this end, we resized all the images to 400×400 and computed dense SIFT descriptors on a regular grid with 4 pixels spacing. The color descriptors were obtained by simply stacking colors from 3×3 patches centered at the grid points. Each grid point therefore yields one 155-dimensional observation $\boldsymbol{o}_i$ in Eq. (37). The parameters for this experiments were set to $\nu_w = 6$ (minimum number of samples in a class), $m = 20$ and $\nu_b = 3$ obtained by 5-fold cross-validation.

TABLE 3
Recognition Accuracies for the UIUC Material Dataset [56]

Method	Recognition Accuracy
CDL [10]	$52.3\% \pm 4.3$
NN-AIRM	$35.6\% \pm 2.6$
NN-AIRM-DR	$58.3\% \pm 2.3$
NN-S	$35.8\% \pm 2.6$
NN-S-DR	$58.1\% \pm 2.8$
kSC-S	$52.8\% \pm 2.1$
kSC-S-DR	$\mathbf{66.6\% \pm 3.1}$
NN-J	$30.9\% \pm 2.4$
NN-J-DR	$53.4\% \pm 2.9$
NN-lE	$36.7\% \pm 2.8$
NN-lE-DR	$51.2\% \pm 3.0$
kSC-lE	$57.7\% \pm 4.2$
kSC-lE-DR	$63.9\% \pm 4.3$

Table 3 compares the performance of the studied algorithms. The performance of the state-of-the-art method on this dataset was reported to be 43.5 percent [56]. The results show that appropriate manifold-based methods (i.e., kSC-S and CDL) with the original 155×155 RCMs already outperform this state-of-the-art, while nearest neighbor (e.g., NN-AIRM, NN-S) on the same manifold yields worse performance. However, after applying our learning algorithm, NN not only outperforms the state-of-the-art significantly, but also outperforms both CDL and kSC, except for the log-Euclidean solution. For example, kSC using the S divergence is boosted by nearly 14 percent by dimensionality reduction. The maximum accuracy of 66.6 percent is obtained by kernel sparse coding on the learned SPD manifold (kSC-S-DR).

6.1.2 Action Recognition from Motion Capture Data

As a second experiment, we tackled the problem of human action recognition from motion capture sequences using the HDM05 database [58]. This database contains the following 14 actions: 'clap above head', 'deposit floor', 'elbow to knee', 'grab high', 'hop both legs', 'jog', 'kick forward', 'lie down floor', 'rotate both arms backward', 'sit down chair', 'sneak', 'squat', 'stand up lie' and 'throw basketball' (see Fig. 5 for an example). The dataset provides the 3D locations of 31 joints over time acquired at the speed of 120 frames per second. We describe an action of a K joints skeleton observed over τ frames by its joint covariance descriptor [60], which is an SPD matrix of size $3K \times 3K$. More specifically, let $x_i(t)$, $y_i(t)$ and $z_i(t)$ be the x, y, and z coordinates of the ith joint at frame t. Let $\boldsymbol{s}(t)$ be the vector of all joint locations at time t, i.e., $\boldsymbol{s}(t) = \big(x_1(t), \cdots, x_K(t), y_1(t), \ldots, y_K(t), z_1(t), \ldots, z_K(t)\big)^T$, which has $3K$ elements. The SPD matrix describing an action occurring over τ frames is then taken as the covariance of the vectors $\boldsymbol{s}(t),\ 1 \leq t \leq \tau$.

In our experiments, we used 2 subjects for training (i.e., 'bd' and 'mm') and the remaining 3 subjects for testing (i.e., 'bk', 'dg' and 'tr').[3] This resulted in 118 training and 188 test sequences for this experiment. The parameters of our method were set to $\nu_w = 5$ (minimum number of samples in one class), $m = 65$ and $\nu_b = 5$ by cross-validation.

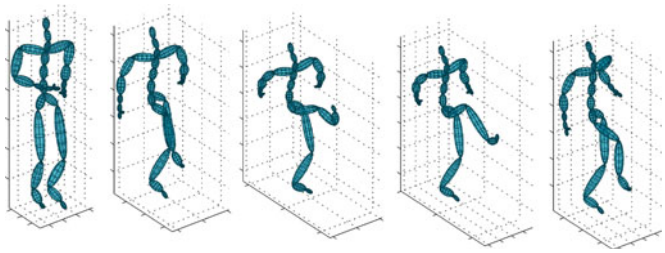

Fig. 5. Kicking action from the HDM05 motion capture sequences database [58].

We report the performance of the different methods on this dataset in Table 4. Again we can see that the accuracies of NN and kSC are significantly improved by our learning algorithm, and that the kSC-S-DR approach achieves the best accuracy of 81.9 percent.

6.1.3 Face Recognition

We then used the YTC dataset [59] for the task of image-set-based face recognition. The YTC dataset contains 1,910 video clips of 47 subjects. See Fig. 6 for samples from YTC. We used face regions extracted from the videos and resized them to 64×64. From each frame in a video, we then extracted 4 histograms of Local Binary Patterns (LBP) [61], each obtained from a 32×32 sub-region of the frame. By concatenating the LBP histograms, frame i in a video is described by $\boldsymbol{o}_i$, a 232-dimensional vector. A video is then described by one SPD matrix of the form

$$C = \begin{bmatrix} OO^T + \mu\mu^T & \mu \\ \mu^T & 1 \end{bmatrix}, \tag{38}$$

where $\mathbb{R}^{232\times\tau} \ni \boldsymbol{O} = [\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_\tau]$ is a matrix storing the descriptors of all τ frames of a video and $\mu = \frac{1}{\tau}\sum_{i=1}^{\tau} \boldsymbol{o}_i$. Following standard practice [62], 3 videos from each person were randomly chosen as training/gallery data, and the query set contained 6 randomly chosen videos from each subject. The process of random selection was repeated 5 times.

TABLE 4
Recognition Accuracies for the HDM05-MOCAP Dataset [58]

Method	Recognition Accuracy
CDL [10]	79.8%
NN-AIRM	62.8%
NN-AIRM-DR	67.6%
NN-S	61.7%
NN-S-DR	68.6%
kSC-S	76.1%
kSC-S-DR	**81.9%**
NN-J	69.2%
NN-J-DR	71.8%
NN-lE	69.7%
NN-lE-DR	71.3%
kSC-lE	75.5%
kSC-lE-DR	78.7%

3. Note that this differs from the setup in [60], where 3 subjects were used for training and 2 for testing. However, with the setup of [60] where an accuracy of 95.41 percent was reported, all our algorithms gave about 99 percent accuracy, which made it impossible to compare them.

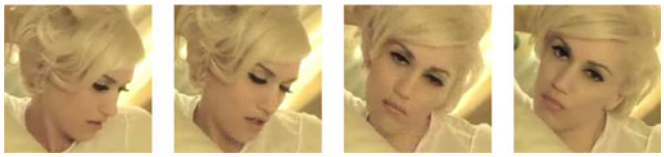

Fig. 6. Samples from YouTube celebrity [59].

Fig. 7. Sample images from the UMD Keck body-gesture dataset [66].

In Table 5, we compare the performance of all the studied algorithm. To the best of our knowledge, the highest reported accuracy using holistic descriptors (i.e., one descriptor per video) is 78.2 percent [62]. Both kSC methods after dimensionality reduction outperform this result, with kSC-S-DR achieving the maximum performance of 80.1 percent. Note also that our DR scheme significantly boosts the performance of NN using the log-Euclidean and the Stein metrics (e.g., from 45.4 percent to 72.8 percent in the case of the Stein divergence).

6.2 Video Clustering

The unsupervised algorithm introduced in Section 3.2 allows us to obtain a low-dimensional SPD manifold from a high-dimensional one by maximizing a notion of data variance. We now evaluate the performance of this unsupervised DR approach on the task of video clustering. To this end, we report both the clustering accuracy and the Normalized Mutual Information (NMI) [63], which measures the amount of statistical information shared by random variables representing the cluster distribution and the underlying class distribution of the data points. Let P_C be the random variable denoting the cluster assignments of the points and P_K the random variable denoting the underlying class labels on the points. Then, the NMI is defined as

$$NMI = 2\frac{I(P_C; P_K)}{H(P_C) + H(P_K)}, \tag{39}$$

where $I(P_X; P_Y) = H(P_X) - H(P_X|P_Y)$ is the mutual information between the random variables P_X and P_Y, $H(P_X)$ is the Shannon entropy of P_X, and $H(P_X|P_Y)$ is the conditional entropy of P_X given P_Y. The normalization by the average entropy of P_C and P_K makes the NMI range between 0 and 1. To measure the clustering accuracy, we followed the metric described in [64]. More specifically, for a query sample $\boldsymbol{X}_i$, let r_i and s_i be the obtained cluster label and the ground-truth label, respectively. The accuracy (AC) is then defined as

$$\mathrm{AC} = \frac{1}{n}\sum_{i=1}^{n} g(s_i, \mathrm{map}(r_i)),$$

where n is the total number of queries, $g(x, y)$ is equal to one if $x = y$ and zero otherwise, and $\mathrm{map}(r_i)$ is the permutation mapping function that maps each cluster label r_i to the equivalent label from the ground truth. The best mapping can be found by using the Kuhn-Munkres algorithm [65].

TABLE 5
Recognition Accuracies for the YTC Dataset [59]

Method	Recognition Accuracy
CDL [10]	70.9%
NN-AIRM	64.7%
NN-AIRM-DR	75.7%
NN-S	45.4%
NN-S-DR	72.8%
kSC-S	78.0%
kSC-S-DR	**80.1%**
NN-J	62.0%
NN-J-DR	68.9%
NN-lE	39.8%
NN-lE-DR	55.0%
kSC-lE	73.5%
kSC-lE-DR	78.8%

For the task of clustering, we used the static setting of the UMD Keck body-gesture data set [66], which consists of 126 videos of 14 naval body gestures. Samples are shown in Fig. 7. We described each video in a similar manner as in the YTC experiment, albeit with a couple of differences. More specifically, we used Histograms of Gradients (HoG) [67] instead of LBP histograms to describe each frame. Furthermore, each frame was resized to 32×32, and we concatenated HoG features extracted from 16×16 non-overlapped regions to form the frame descriptor. Using the idea of Eq. (38) to aggregate the frame descriptors, we obtained an SPD matrices of size 125×125 to describe each video.

For our evaluation, we employed the k-means algorithm on the manifold using the AIRM and the Jeffrey and Stein divergences. We also made use of the k-means algorithm on the identity tangent space for the log-Euclidean metric. In addition to k-means on the manifold, we also utilized the kernel k-means algorithm using the Jeffrey, Stein and log-Euclidean kernels. We refer to the algorithms evaluated in our experiments as:

KM-AIRM: k-means based on the AIRM on the high-dimensional SPD manifold.
KM-S: k-means based on the S divergence on the high-dimensional SPD manifold.
KM-J: k-means based on the J divergence on the high-dimensional SPD manifold.
KM-lE: k-means based on the log-Euclidean metric on the high-dimensional SPD manifold.
KM-AIRM-DR: k-means based on the AIRM on the low-dimensional SPD manifold obtained with our approach.
KM-S-DR: k-means based on the S divergence on the low-dimensional SPD manifold obtained with our approach.
KM-J-DR: k-means based on the J divergence on the low-dimensional SPD manifold obtained with our approach.
KM-lE-DR: k-means based on the log-Euclidean metric on the low-dimensional SPD manifold obtained with our approach.
kKM-S: kernel k-means based on the S divergence on the high-dimensional SPD manifold.
kKM-lE: kernel k-means based on the log-Euclidean metric on the high-dimensional SPD manifold.

TABLE 6
Recognition Accuracies and Normalized Mutual Information Scores (Mean and Standard Deviations) for the Keck Dataset [66]

Method	AC	NMI
AIRM	$56.2\% \pm 1.2$	$73.0\% \pm 1.8$
AIRM-DR	$64.7\% \pm 2.0$	$79.0\% \pm 1.7$
KM-S	$53.9\% \pm 1.4$	$71.0\% \pm 1.1$
KM-S-DR	$61.4\% \pm 1.2$	$78.7\% \pm 0.9$
kKM-S	$64.1\% \pm 1.8$	$77.5\% \pm 0.1$
kKM-S-DR	$71.2\% \pm 1.4$	$83.7\% \pm 0.4$
KM-J	$53.8\% \pm 1.7$	$71.2\% \pm 1.0$
KM-J-DR	$55.3\% \pm 2.1$	$72.8\% \pm 0.9$
KM-lE	$62.7\% \pm 0.9$	$79.2\% \pm 0.3$
KM-lE-DR	$75.3\% \pm 1.5$	$88.3\% \pm 0.2$
kKM-lE	$71.3\% \pm 1.7$	$83.5\% \pm 0.2$
kKM-lE-DR	$\mathbf{83.2\% \pm 0.2}$	$\mathbf{91.8\% \pm 0.2}$

kKM-S-DR: kernel k-means based on the S divergence on the low-dimensional SPD manifold obtained with our approach.

kKM-lE-DR: kernel k-means based on the log-Euclidean metric on the low-dimensional SPD manifold obtained with our approach.

Table 6 reports the accuracy and NMI values for all the studied methods. It is interesting to see that the log-Euclidean metric achieves better accuracy on this dataset. We also note that the AIRM outperforms the solutions based on the Bregman divergences. However, with kernel k-means, the Stein-based algorithm surpasses the AIRM-based one.

6.3 Parameter Sensitivity

In all our experiments, the parameters of our approach were set in a principled manner (i.e., ν_w is the minimum number of samples in one class, and ν_b by cross-validation). In this section, we nonetheless study the influence of these parameters on the overall performance. To this end, we employed the UIUC material dataset and report the accuracy of our NN-AIRM-DR method when varying one parameter and fixing the other to the values reported in Section 6.1.1. In particular, Fig. 8 depicts the recognition accuracy for various values of ν_b in the interval $[1, 12]$. Note that for $\nu_b = 1$, which is equivalent to mainly considering the intra-class discrimination, the performance drops. For $\nu_b = 12$, which makes the inter-class discrimination dominant, the performance drops even further. The maximum performance of 58.6 percent is achieved for $\nu_b = 4$, which again shows that balance between the intra-class and inter-class terms is important.

We also analyzed the sensitivity of the algorithm to ν_w. To this end, we fixed the value of $\nu_b = 5$ and varied the value of ν_w from one to five. We observed that the accuracy monotonically increased for this test. We emphasize that tuning the values of ν_b and ν_w is a common task in a large group of learning techniques (e.g., distance learning [34] and dimensionality reduction [25]) and should not be perceived as a bottleneck for the development done in this work.

7 Conclusions and Future Work

We have introduced an approach to mapping a high-dimensional SPD manifold into a lower-dimensional one. In particular, we have derived both a supervised and an unsupervised formulation. In both cases, we have studied different metrics to encode the similarity between SPD matrices, namely, the AIRM, the Stein divergence, the Jeffrey divergence and the log-Euclidean metric. Our experiments have shown that reducing the dimensionality consistently improved accuracy over directly using the high-dimensional SPD matrices. We believe that this work, extending our preliminary study [53] that already generated follow-ups [31], [32], [36], is an important step towards developing DR algorithms dedicated to Riemannian manifolds, and in particular in the context of going from a high-dimensional manifold to a lower-dimensional one. In the future, we therefore intend to extend this framework to other types of Riemannian manifolds.

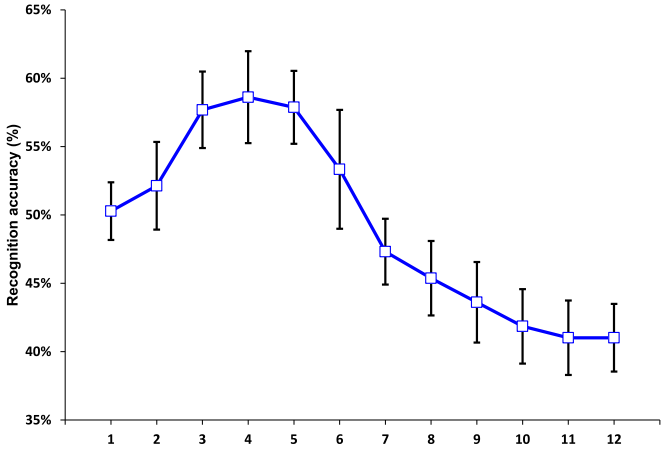

Fig. 8. Accuracy on the UIUC material dataset for varying values of ν_b.

Acknowledgments

The first author wishes to thank Dr. Florian Yger for fruitful discussions on this topic.

References

[1] D. G. Kendall, "Shape manifolds, procrustean metrics, and complex projective spaces," *Bulletin London Math. Soc.*, vol. 16, no. 2, pp. 81–121, 1984.

[2] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, 2006.

[3] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Free-form region description with second-order pooling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 6, pp. 1177–1189, Jun. 2015.

[4] O. Tuzel, F. Porikli, and P. Meer, "Region covariance: A fast descriptor for detection and classification," in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 589–600.

[5] M. T. Harandi, R. Hartley, B. C. Lovell, and C. Sanderson, "Sparse coding on symmetric positive definite manifolds using Bregman divergences," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 27, no. 6, pp. 1294–1303, Jun. 2016.

[6] 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.

[7] D. Tosato, M. Spera, M. Cristani, and V. Murino, "Characterizing humans on Riemannian manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1972–1984, Aug. 2013.

[8] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel methods on Riemannian manifolds with Gaussian RBF kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 12, pp. 2464–2477, Dec. 2015.

[9] 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.

[10] R. Wang, H. Guo, L. S. Davis, and Q. Dai, "Covariance discriminative learning: A natural and efficient approach to image set classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2496–2503.
[11] 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.
[12] A. Sanin, C. Sanderson, M. Harandi, and B. Lovell, "Spatio-temporal covariance descriptors for action and gesture recognition," in *Proc. IEEE Workshop Appl. Comput. Vis.*, 2013, pp. 103–110.
[13] K. Guo, P. Ishwar, and J. Konrad, "Action recognition from video using feature covariance matrices," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2479–2494, Jun. 2013.
[14] C. Atkinson and A. F. Mitchell, "Rao's distance measure," *Sankhyā: Indian J. Stat., Series A*, vol. 43, pp. 345–365, 1981.
[15] Q. Wang, P. Li, W. Zuo, and L. Zhang, "RAID-G: Robust estimation of approximate infinite dimensional Gaussian with application to material recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4433–4441.
[16] S. Sra, "A new metric on the manifold of kernel matrices with application to matrix geometric means," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 144–152.
[17] Z. Wang and B. C. Vemuri, "An affine invariant tensor dissimilarity measure and its applications to tensor-valued image segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2004, pp. I–228.
[18] K. Q. Weinberger and L. K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *Int. J. Comput. Vis.*, vol. 70, no. 1, pp. 77–90, 2006.
[19] 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.
[20] R. Bhatia, *Positive Definite Matrices*. Princeton, NJ, USA: Princeton Univ. Press, 2007.
[21] B. Kulis, M. A. Sustik, and I. S. Dhillon, "Low-rank kernel learning with Bregman matrix divergences," *J. Mach. Learning Res.*, vol. 10, pp. 341–376, Feb. 2009.
[22] P. J. Moreno, P. P. Ho, and N. Vasconcelos, "A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 1385–1392.
[23] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifolds," *J. Mach. Learning Res.*, vol. 4, pp. 119–155, 2003.
[24] H. Li, T. Jiang, and K. Zhang, "Efficient and robust feature extraction by maximum margin criterion," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 157–165, Jan. 2006.
[25] S. Yan, D. Xu, B. Zhang, H.-J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, Jan. 2007.
[26] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Trans. Neural Netw.*, vol. 20, no. 4, pp. 729–735, Apr. 2009.
[27] E. Kokiopoulou, J. Chen, and Y. Saad, "Trace optimization and eigenproblems in dimension reduction methods," *Numerical Linear Algebra Appl.*, vol. 18, no. 3, pp. 565–602, 2011.
[28] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.
[29] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2008.
[30] J. P. Cunningham and Z. Ghahramani, "Linear dimensionality reduction: Survey, insights, and generalizations," *J. Mach. Learning Res.*, vol. 16, pp. 2859–2900, 2015.
[31] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen, "Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification," in *Proc. Int. Conf. Mach. Learning*, 2015, pp. 720–729.
[32] Z. Huang, R. Wang, S. Shan, and X. Chen, "Projection metric learning on Grassmann manifold with application to video based face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 140–149.
[33] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," *J. Mach. Learning Res.*, vol. 15, pp. 1455–1459, 2014. [Online]. Available: http://www.manopt.org
[34] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1473–1480.
[35] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," Nov 2012, version 20121115. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?3274
[36] F. Yger and M. Sugiyama, "Supervised log-Euclidean metric learning for symmetric positive definite matrices," *CoRR*, 2015. [Online]. Available: http://arxiv.org/abs/1502.03505
[37] S. H. Cheng, N. J. Higham, C. S. Kenney, and A. J. Laub, "Approximating the logarithm of a matrix to specified accuracy," *SIAM J. Matrix Anal. Appl.*, vol. 22, no. 4, pp. 1112–1125, 2001.
[38] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Log-Euclidean metrics for fast and simple calculus on diffusion tensors," *Magnetic Resonance Med.*, vol. 56, no. 2, pp. 411–421, 2006.
[39] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2013.
[40] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2965–2973.
[41] Z. Huang and L. Van Gool, "A Riemannian network for spd matrix learning," in *Prc. Assoc. Advancement Artif. Intell.*, 2017.
[42] P. T. Fletcher, C. Lu, and S. Joshi, "Statistics of shape via principal geodesic analysis on Lie groups," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. I–95.
[43] P. T. Fletcher, C. Lu, S. M. Pizer, and S. Joshi, "Principal geodesic analysis for the study of nonlinear statistics of shape," *IEEE Trans. Med. Imaging*, vol. 23, no. 8, pp. 995–1005, Aug. 2004.
[44] S. Said, N. Courty, N. Le Bihan, and S. J. Sangwine, "Exact principal geodesic analysis for data on SO(3)," in *Proc. Eur. Signal Proc. Conf.*, 2007, pp. 1700–1705.
[45] S. Huckemann, T. Hotz, and A. Munk, "Intrinsic shape analysis: Geodesic principal component analysis for Riemannian manifolds modulo Lie group actions. discussion paper with rejoinder," *Statistica Sinica*, vol. 20, pp. 1–100, 2010.
[46] S. Sommer, F. Lauze, S. Hauberg, and M. Nielsen, "Manifold valued statistics, exact principal geodesic analysis and the effect of linear approximations," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 43–56.
[47] S. Sommer, et al., "Bicycle chain shape models," in *Proc. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2009, pp. 157–163.
[48] 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.*, 2013, pp. 1601–1608.
[49] A. Feragen, F. Lauze, and S. Hauberg, "Geodesic exponential kernels: When curvature and linearity conflict," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3032–3042.
[50] A. Goh and R. Vidal, "Clustering and dimensionality reduction on Riemannian manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2008, pp. 1–7.
[51] H. Wang, A. Banerjee, and D. Boley, "Common component analysis for multiple covariance matrices," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 956–964.
[52] I. Horev, F. Yger, and M. Sugiyama, "Geometry-aware principal component analysis for symmetric positive definite matrices," in *Proc. Asian Conf. Mach. Learning*, 2015, pp. 1–16.
[53] M. T. Harandi, M. Salzmann, and R. Hartley, "From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 17–32.
[54] T. Lin and H. Zha, "Riemannian manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 796–809, May 2008.
[55] M. Harandi and M. Salzmann, "Riemannian coding and dictionary learning: Kernels to the rescue," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3926–3935.
[56] Z. Liao, J. Rock, Y. Wang, and D. Forsyth, "Non-parametric filtering for geometric detail extraction and material representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 963–970.
[57] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[58] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber, "Documentation: Mocap database HDM05," Universität Bonn, Bonn, Germany, Tech. Rep. CG-2007-2, 2007.
[59] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8, doi: 10.1109/CVPR.2008.4587572.

[60] M. E. Hussein, M. Torki, M. A. Gowayyed, and M. El-Saban, "Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations," in *Proc. Int. Joint Conf. Artif. Intell.*, 2013, pp. 2466–2472.
[61] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.
[62] J. Lu, G. Wang, and P. Moulin, "Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 329–336.
[63] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Proc. AAAI Workshop Artif. Intell. Web Search*, 2000, pp. 58–64.
[64] D. Cai, X. He, and J. Han, "Document clustering using locality preserving indexing," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 12, pp. 1624–1637, Dec. 2005.
[65] L. Lovász and M. D. Plummer, *Matching Theory*. Providence, RI, USA: American Math. Soc., 2009, vol. 367.
[66] Z. Lin, Z. Jiang, and L. Davis, "Recognizing actions by shape-motion prototype trees," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Sep. 2009, pp. 444–451.
[67] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 886–893.

Mehrtash Harandi is a senior research scientist at Computer Vision Research Group (CVRG), Data61 and an adjunct lecturer with Australian National University, Canberra, Australia. His main research interests are theoretical and computational methods in computer vision and machine learning with a focus on Riemannian geometry. He is member of the IEEE.

Mathieu Salzmann received the PhD degree from EPFL, in Jan. 2009, under the supervision of Prof. Pascal Fua. He is a senior researcher at EPFL-CVLab. Previously, he was a senior researcher and Research Leader in NICTA's computer vision research group. Prior to this, from Sep. 2010 to Jan. 2012, he was a research assistant professor at TTI-Chicago, and, from Feb. 2009 to Aug. 2010, a postdoctoral fellow at ICSI and EECS with UC Berkeley under the supervision of Prof. Trevor Darrell. His research interests include visual recognition, 3D reconstruction and Riemannian geometry for computer vision. He is member of the IEEE.

Richard Hartley is a member of the computer vision group in the Research School of Engineering, at ANU, where he has been since January, 2001. He is also a member of the computer vision research group in NICTA. He worked with the GE Research and Development Center from 1985 to 2001, working first in VLSI design, and later in computer vision. He became involved with Image Understanding and Scene Reconstruction working with GE's Simulation and Control Systems Division. He is an author (with A. Zisserman) of the book *Multiple View Geometry in Computer Vision*. He is fellow of the IEEE.