

# Locality preserving projection on SPD matrix Lie group: algorithm and analysis

Yangyang LI<sup>1,2\*</sup> & Ruqian LU<sup>1</sup><sup>1</sup>*Academy of Mathematics and Systems Science Key Lab of Management, Decision and Information Systems, Chinese Academy of Sciences, Beijing 100190, China;*<sup>2</sup>*University of Chinese Academy of Sciences, Beijing 100049, China*

Received 15 March 2017/Revised 31 May 2017/Accepted 16 August 2017/Published online 27 April 2018

**Abstract** Symmetric positive definite (SPD) matrices used as feature descriptors in image recognition are usually high dimensional. Traditional manifold learning is only applicable for reducing the dimension of high-dimensional vector-form data. For high-dimensional SPD matrices, directly using manifold learning algorithms to reduce the dimension of matrix-form data is impossible. The SPD matrix must first be transformed into a long vector, and then the dimension of this vector must be reduced. However, this approach breaks the spatial structure of the SPD matrix space. To overcome this limitation, we propose a new dimension reduction algorithm on SPD matrix space to transform high-dimensional SPD matrices into low-dimensional SPD matrices. Our work is based on the fact that the set of all SPD matrices with the same size has a Lie group structure, and we aim to transform the manifold learning to the SPD matrix Lie group. We use the basic idea of the manifold learning algorithm called locality preserving projection (LPP) to construct the corresponding Laplacian matrix on the SPD matrix Lie group. Thus, we call our approach Lie-LPP to emphasize its Lie group character. We present a detailed algorithm analysis and show through experiments that Lie-LPP achieves effective results on human action recognition and human face recognition.

**Keywords** manifold learning, SPD matrix Lie group, locally preserving projection, Laplace operator, Log-Euclidean metric

**Citation** Li Y Y, Lu R Q. Locality preserving projection on SPD matrix Lie group: algorithm and analysis. *Sci China Inf Sci*, 2018, 61(9): 092104, <https://doi.org/10.1007/s11432-017-9233-4>

## 1 Introduction

Image recognition, including dynamic action recognition and static image recognition, is a popular research subject in the field of machine vision and pattern recognition [1–4]. This technique has a wide range of applications in many fields, such as intelligent video retrieval and perceived interaction. One key step of image recognition is to construct a high-quality image feature descriptor, which determines the accuracy rate of recognition. The original image feature descriptor is a pixel matrix, which is usually transformed into a high-dimensional row feature vector because image recognition on the pixel matrix space is difficult. Given the high dimension of the feature vector, manifold learning algorithms are applied to implement image recognition [1, 5–7]. However, the vector-form feature descriptor breaks the geometric structure of the pixel matrix space and is highly sensitive to various factors such as illumination intensity, background, and object location. To avoid these disadvantages, tensor space dimensionality reduction based on locality preserving projection (LPP) [8] was proposed in [9]; this approach is linear

\* Corresponding author (email: liyangyang12@mails.ucas.ac.cn)

and deals with the image pixel matrix directly. Porikli et al. [10] presented a new feature descriptor by computing a feature covariance matrix within any size region in an image, which preserves the local geometric structure of the pixel matrix (see Appendix A in the supplementary file). Covariance matrix, which is also called symmetric positive definite (SPD) matrix descriptor, has mainly been used in static image recognition [4, 10–12]. For human action recognition, Sanin et al. [13] proposed a new method based on spatial-temporal covariance descriptors. Tabia et al. [14] applied SPD matrices as descriptors of 3D skeleton location by constructing covariance matrices on 3D joint locations.

A set of SPD matrices with the same size forms a Riemannian manifold. This SPD Riemannian manifold has a group structure that forms an SPD matrix Lie group  $S_+^D$ . The group operation in [15] is shown as

$$S_1, S_2 \in S_+^D, \quad S_1 \odot S_2 \doteq \exp(\log(S_1) + \log(S_2)).$$

We proposed that the dimensions of SPD matrices calculated from image regions are especially high in general. As a result of the matrix-form data of the matrix Lie group, directly reducing the dimension of high-dimensional matrix Lie group is difficult without breaking the structure of the matrix form.

Harandi et al. [16] proposed to learn a kernel function to first map the SPD matrices into a higher-dimensional Euclidean space and then use LPP [8] to reduce their dimension. However, this method would distort the geometric and algebraic structure of the SPD manifold and would lose a considerable amount of important structure information. To overcome this limitation, Harandi et al. [17] suggested mapping the original SPD manifold to a Grassmann manifold and then solving the dimension reduction problem on the latter. However, this method has a high time cost. A similar study was conducted by Huang et al. [15], who proposed transforming SPD matrices to their corresponding tangent space and learning a linear dimensionality reduction map on that tangent space. However, this algorithm needs several parameters, which are sensitive factors that influence the algorithm. Overall, all three methods [15–17] require a linear space and rely on nonlinear dimensionality reduction mappings.

According to the definition of covariance matrix in [10, 18], each covariance matrix can be represented by the product of a set of feature vectors. Covariance matrix summarizes the linear bivariate relationships among a set of variables. Thus, we can solve the dimensionality reduction problem directly on the SPD matrix Lie group. We extend the idea of LPP [8] to the dimensionality reduction learning on the SPD matrix Lie group in a new approach called Lie-LPP.

The main contributions of our study can be summarized as follows:

- The LPP algorithm in [8] is extended to Lie-LPP and applied to the SPD matrix Lie group. A bilinear dimensionality reduction mapping on the SPD matrix Lie group is obtained, which preserves the intrinsic geometric and algebraic structure of the SPD matrix Lie group.
- To overcome the limitation of other methods regarding dimensionality reduction of the SPD matrix Lie group, our method solves the dimensionality reduction problem of the SPD matrix Lie group directly without mapping to other spaces and without needing numerous sensitive parameters, thereby resulting in a simple and straightforward approach.
- The graph Laplacian matrix is constructed on the SPD matrix Lie group to reflect the intrinsic geometric and algebraic structure of the original SPD matrix Lie group.
- A detailed algorithm analysis in theory is given to analyze the difference and relationship between Lie-LPP and LPP. The main conclusions are shown in Theorems 1 and 2 and in Proposition 1.

This paper is a further extension of the algorithm and analysis in our previous paper [19]. In [19], we simply presented a simple description of our algorithm. In this paper, we present the detailed algorithm analysis in theory, as well as full experiments on human action and face databases.

## 2 Background

### 2.1 Covariance matrix

Suppose a set of  $n$  feature vectors from an image region are expressed as the following matrix  $F$ , where

the dimensionality of each feature vector is  $d$ :

$$F = (f_1, f_2, \dots, f_n),$$

$f_i = (f_{i_1}, f_{i_2}, \dots, f_{i_d})$  is the  $i^{\text{th}}$  feature vector. The corresponding covariance matrix  $C$  with respect to these feature vectors is defined as [18]

$$C = \frac{1}{n} F F^T = \frac{1}{n} \sum_{i=1}^n f_i f_i^T. \quad (1)$$

In this definition, we assume the expectation of feature vectors is zero and each term  $f_i f_i^T$  in the summation is the outer product of the feature vector  $f_i$ . Obviously, covariance matrix is a semi-positive definite matrix. In this paper, we consider only the positive covariance matrix called the SPD matrix. When the feature vectors are adjacent, the corresponding covariance matrices are also adjacent. Thus, the covariance matrices preserve the local geometric structure of the corresponding feature vectors. The detailed proof is shown in Appendix A.

Covariance matrices (SPD matrices) have several advantages as the feature descriptors of images. First, covariance matrices can fuse all the features of images. Second, they provide a way of filtering noisy information such as illumination and the location of the object in the image. In addition, the size of the covariance matrix is dependent on the dimensionality of feature vectors other than the size of the image region. Thus, we can construct the covariance matrices with the same size from different regions.

## 2.2 Geometric structure of SPD matrix Lie group

In machine learning, we usually have to learn an effective metric for comparing data points. In particular, in the image recognition step, a metric is required to measure the distance between two different image feature descriptors. In this paper, we use the SPD matrices as feature descriptors of images. Thus, the corresponding Riemannian metric needs to be constructed on the SPD matrix Lie group to compute the intrinsic geodesic distances between any two SPD matrices.

The SPD matrix Lie group that we consider in this paper is represented by  $\mathcal{S}_+^D$ , where every point  $S_1 \in \mathcal{S}_+^D$  is a  $D \times D$  size matrix. The tangent space of  $\mathcal{S}_+^D$  at the identity is  $\text{Sym}(D)$ , a bilinear space of symmetric matrices. The learned lower-dimensional SPD matrix Lie group is represented by  $\mathcal{S}_+^d$ . The family of all scalar products on all tangent spaces of SPD matrix Lie group is known as the Riemannian metric. The geodesic distance  $d_G(S_1, S_2)$  between any two points  $S_1, S_2 \in \mathcal{S}_+^D$  can be computed under this Riemannian metric. In this paper, we choose the Log-Euclidean metric (LEM) from [20] as the Riemannian metric of SPD matrix Lie group. The detailed definition of LEM is shown in Definition 1.

**Definition 1** (Log-Euclidean metric [20]). The Riemannian metric at a point  $S_1 \in \mathcal{S}_+^D$  is a scalar product defined in the tangent space  $T_{S_1}(\mathcal{S}_+^D)$ ,

$$\langle T_1, T_2 \rangle = \langle D \log \cdot T_1, D \log \cdot T_2 \rangle, \quad (2)$$

where  $T_1, T_2 \in T_{S_1}(\mathcal{S}_+^D)$ ,  $D \log \cdot T_1$  indicates the directional derivative of the matrix logarithm along  $T_1$  [15].

LEM is a bi-invariant metric defined on the SPD matrix Lie group. The corresponding theoretical conclusion is shown in [20].

**Definition 2** (Bi-invariant metric [20]). Any bi-invariant metric on the Lie group of SPD matrices is also called a LEM because it corresponds to a Euclidean metric (EM) in the logarithm domain.

The logarithm domain is the tangent space of the SPD matrix Lie group.

**Corollary 1** (Flat Riemannian manifold [20]). Endowed with a bi-invariant metric, the space of SPD matrices is a flat Riemannian space; its sectional curvature is null everywhere.

Thus, under LEM,  $\mathcal{S}_+^D$  is a flat manifold and locally isometric to the tangent space  $\text{Sym}(D)$ . In a local neighborhood, the mapping between the SPD matrix Lie group and the corresponding tangent space is represented by the exponential map, and the inverse map is the logarithm shown in (3) and (4).

$$\exp_{S_1}(T_1) = \exp(\log(S_1) + D_{S_1} \log \cdot T_1), \quad (3)$$

$$\log_{S_1}(S_2) = D_{\log(S_1)} \exp \cdot (\log(S_2) - \log(S_1)), \quad (4)$$

where the exponential map is defined at point  $S_1 \in \mathcal{S}_+^D$ ,  $T_1 \in T_{S_1}\mathcal{S}_+^D$  is a tangent vector.  $\exp_{S_1}(T_1)$  is the exponential representation of  $T_1$  on  $\mathcal{S}_+^D$ , and the corresponding logarithmic representation of  $S_2$  at  $T_{S_1}\mathcal{S}_+^D$  is  $\log_{S_1}(S_2)$ .

The geodesic distance between  $S_1$ ,  $S_2$  under LEM is defined as follows [15]:

$$d_G(S_1, S_2) = \langle \log_{S_1}(S_2), \log_{S_1}(S_2) \rangle = \|\log(S_1) - \log(S_2)\|_F^2, \quad (5)$$

where  $\|\cdot\|_F$  represents the Frobenius norm of the matrix and  $\log_{S_1}(S_2)$  is the logarithm of  $S_2$  at  $S_1$ .

Under LEM, the SPD matrix Lie group is a complete manifold. Thus, every two points on the SPD matrix Lie group are linked by the shortest geodesic line.

### 3 Algorithm

In this section, we first analyze the LPP algorithm. Then, we give the construction process of graph Laplacian matrix on the SPD matrix Lie group. Finally, we describe the process of our proposed dimensionality reduction algorithm Lie-LPP.

#### 3.1 Locally preserving projection (LPP)

LPP aims to learn a linear dimensionality reduction map to reduce the dimension of high-dimensional vector-form data points, which can be seen as a linear approximation of LEP [21]. This linear dimensionality reduction map optimally preserves the local neighborhood geometric structure of a dataset by building a graph Laplacian matrix on the dataset. The graph Laplacian matrix is a discrete approximation of a differential Laplace operator that arises from the manifold. Let  $X = [x_1, x_2, \dots, x_N] \subset \mathbb{R}^D$  is the input data set distributed on a  $d$ -dimensional manifold  $\mathcal{M}$ , which is embedded into  $\mathbb{R}^D$ .  $a$  is the linear dimensionality reduction map. The learned lower-dimensional data set is  $Y = a^T X = [y_1, y_2, \dots, y_N] \subset \mathbb{R}^d$ .

The algorithm of LPP is stated as follows:

- **Constructing the adjacency graph.** Denote  $G$  a graph with  $N$  nodes. If  $x_i$  and  $x_j$  are “close”, then a connection exists between nodes  $i$  and  $j$ . The “closenes” between two nodes is measured by the  $K$ -nearest neighbor method.

- **Choosing the weights.** Here, the authors denote matrix  $W$  as the corresponding weight matrix, where  $W$  is a sparse symmetric matrix with size  $N \times N$ . The element  $W_{ij}$  is defined as follows:

$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{\tau}}, \quad \text{if nodes } i \text{ and } j \text{ are connected,}$$

$$W_{ij} = 0, \quad \text{if nodes } i \text{ and } j \text{ are not connected.}$$

- **Eigenmaps.** The following generalized eigenvector problem is solved to obtain the corresponding dimension reduction map  $a$ :

$$X L X^T a = \lambda X D X^T a, \quad (6)$$

where  $D$  is a diagonal matrix,  $D_{ii} = \sum_j W_{ji}$ .  $L = D - W$  is the graph Laplacian matrix.

Thus, the critical step of LPP is to construct the graph Laplacian matrix  $L$  on data points. The global lower dimensional representations are learned by solving the corresponding generalized eigenfunction.

#### 3.2 Laplace operator on SPD matrix Lie group

The Laplace operator is a significant operator defined on Riemannian manifold [22]. It measures the intrinsic structure of manifold such as the curvature of manifold, the similarities among different points on Riemannian manifold. The Laplacian matrix on a graph is a discrete analog of the Laplace operator that we are familiar with in functional analysis [23]. The critical step of the LPP algorithm is to construct a graph Laplacian matrix to represent the intrinsic local geometric structure of data points, which is

applied only on vector-form data points. For Lie-LPP, we aim to uncover the intrinsic structure of SPD matrices, which are essentially different from vector-form data points in a spatial structure. The Laplace operator is defined based on the Riemannian metric. For vector-form data points, the Laplacian matrix is constructed based on the EM in each local patch. For SPD matrices, we need to use LEM [20, 24] to construct the corresponding Laplacian matrix.

Several difficulties are encountered for the learning of the Laplacian matrix on SPD matrices. One critical difficulty is the discrete representations of the first- and second-order derivatives on SPD matrix space. Another difficulty is the need to find the representation of the Laplace operator on the SPD matrix Lie group, which is different from the structure on vector space. To construct a Laplacian matrix on SPD matrices, we need to solve these difficulties during the learning process. In addition, the core of Lie-LPP is to construct an accurate Laplacian matrix on the SPD matrix Lie group before dimensionality reduction. The construction process of a Laplacian matrix on SPD matrices and the descriptions for solving these difficulties are given in detail in a separate section.

To better understand the construction of the Laplacian matrix, we first present an intuitive example of a set of one dimensional nodes. Consider a graph  $G$  with  $n$  nodes. Every node  $i$  is adjacent to two nodes  $i - 1$  and  $i + 1$ . If we assign value  $v_i$  to node  $i$ , then the Laplacian is represented as  $(v_{i+1} - v_i) - (v_i - v_{i-1})$ . Thus,  $v_{i+1} - v_i$  is the discrete analog for a first-order derivative defined over the real number line.  $-2(v_i - \frac{1}{2}v_{i-1} - \frac{1}{2}v_{i+1})$  is the discrete approximation of the second-order derivative. For higher dimensions the ‘normalized’ graph Laplacian  $\Delta_{nm}$  about function  $f$  is defined as

$$(\Delta_{nm}f)(i) = f(i) - \frac{1}{\deg(i)} \sum_{j=1}^n W_{ij}f(j), \quad (7)$$

where the degree function  $\deg$  is defined as  $\deg(i) = \sum_{j=1}^n W_{ji}$ .  $W_{ji}$  is the heat kernel weight defined in the same way as in the second algorithmic procedure of LPP [8].

The abovementioned Laplacian matrix is defined on vector-form data points only. In the following, we construct the ‘normalized’ graph Laplacian matrix on the SPD matrix Lie group endowed with LEM. Suppose a parameterized SPD matrix Lie group  $\Sigma$  defined as  $\Sigma : \mathbb{R}^d \rightarrow \mathcal{S}_+^D$ , we call the vector  $\overrightarrow{\Sigma(x)\Sigma(x+u)}$  as the standard first-order derivative on  $\mathcal{S}_+^D$  [25],

$$\overrightarrow{\Sigma(x)\Sigma(x+u)} = \Sigma(x)^{\frac{1}{2}} (\log \Sigma(x+u) - \log \Sigma(x)) \Sigma(x)^{\frac{1}{2}}. \quad (8)$$

The Laplace-Beltrami operator  $\Delta$  about function  $\Sigma$  is defined as

$$\begin{aligned} \Delta \Sigma &= \sum_{i=1}^d \Delta_i \Sigma, \\ \Delta_i \Sigma &= \partial_i^2 \Sigma - 2(\partial_i \Sigma) \Sigma^{(-1)} (\partial_i \Sigma). \end{aligned} \quad (9)$$

To approximate the graph Laplacian matrix on SPD matrices, we first need to approximate the first- and second-order derivatives on SPD matrix Lie group. Under the approximation of the second-order derivative, the corresponding approximation of the Laplace-Beltrami operator on  $\mathcal{S}_+^D$  [25] is

$$\Delta_u \Sigma = \partial_u^2 \Sigma - 2(\partial_u \Sigma) \Sigma^{(-1)} (\partial_u \Sigma) = \overrightarrow{\Sigma(x)\Sigma(x+u)} + \overrightarrow{\Sigma(x)\Sigma(x-u)} + O(\|u\|^4). \quad (10)$$

To compute the complete Laplacian matrix of the SPD matrix Lie group of (9), we only have to compute the Laplace operator along  $d$  orthonormal directions.

For a discrete dataset, suppose  $\Sigma_1, \Sigma_2, \dots, \Sigma_N$  are a set of SPD matrices generated from  $\Sigma$ . The ‘normal’ graph Laplacian matrix on this data set is

$$(\Delta_{nm}\Sigma_i) = \Sigma_i^{\frac{1}{2}} \left( \log(\Sigma_i) - \sum_{j=1}^n \widetilde{W}_{ij} \log(\Sigma_j) \right) \Sigma_i^{\frac{1}{2}}, \quad (11)$$

where  $\widetilde{W}_{ij} = e^{-\frac{\|\log(S_i) - \log(S_j)\|_F^2}{t}}$ , if  $i$  and  $j$  are connected, else  $\widetilde{W}_{ij} = 0$ . The corresponding graph Laplacian matrix on a set of SPD matrices is  $\widetilde{L} = \widetilde{D} - \widetilde{W}$ , where  $\widetilde{W}$  is a symmetric matrix,  $\widetilde{W}_{ij}$  defined as above and  $\widetilde{D}$  is a diagonal matrix,  $\widetilde{D}_{ii} = \sum_j \widetilde{W}_{ji}$ .  $\widetilde{L}$  is a discrete representation of the Laplace operator on the SPD matrix Lie group.

### 3.3 Lie-LPP

On the basis of the definition of graph Laplacian matrix on  $\mathcal{S}_+^D$  and the algorithmic procedures of LPP, we show our Lie-LPP algorithm as follows.

#### 3.3.1 Lie-LPP algorithm

For the SPD matrix Lie group  $\mathcal{S}_+^D$ , the SPD matrix logarithms in the tangent space are also symmetric matrices. The bilinear mapping between tangent spaces is defined as follows:

$$f(\log(S_1)) = A^T \log(S_1) A. \tag{12}$$

The corresponding mapping between SPD matrix Lie groups is

$$g(S_1) = \exp \circ f(\log(S_1)) = \exp(A^T \log(S_1) A), \tag{13}$$

where  $S_1 \in \mathcal{S}_+^D$ ,  $A$  is a linear map matrix,  $f$  is the corresponding map defined on Lie algebras, and  $g$  is the derived map defined on SPD matrix Lie groups.

$g(S_1)$  is still an SPD matrix; this idea is easily proven. In this paper, we attempt to learn a transformation matrix  $A$  where  $A \in \mathcal{R}^{D \times d}$  is a full column rank matrix.  $D \times D$  is the size of  $\mathcal{S}_+^D$ , and  $d \times d$  is the mapped size of  $\mathcal{S}_+^d$ ,  $D \gg d$ . The linear map matrix  $A$  is proven to preserve the algebraic structure of  $\mathcal{S}_+^D$ . To obtain a discriminative SPD matrix Lie group  $\mathcal{S}_+^d$ ,  $A$  should also inherit and preserve the geometric structure of  $\mathcal{S}_+^D$ . According to the idea of LPP, the key step of the Lie-LPP algorithm is to construct the Laplacian matrix  $L$  on  $\mathcal{S}_+^D$ , which reflects the local geometrical structure of  $\mathcal{S}_+^D$ . Under LEM,  $\mathcal{S}_+^D$  is locally isometric to the tangent space of  $\mathcal{S}_+^D$ . Thus, the geodesic distance between two SPD matrices is equal to the Euclidean distance between the corresponding points on tangent space.

Suppose the input data points are  $S_1, S_2, \dots, S_N \in \mathcal{S}_+^D$ , and the output sample points are  $Y_1, Y_2, \dots, Y_N \in \mathcal{S}_+^d$ , where  $N$  is the number of sampled points.

The algorithm steps of Lie-LPP are as follows:

- The first step is to divide the input SPD matrices into a set of local patches. We use the  $K$ -nearest method to find the  $K$ -nearest neighborhoods  $U_i$  of every point  $S_i$ , where the distance metric between two points is defined as their geodesic distance on the SPD matrix Lie group  $\mathcal{S}_+^D$  using (5).
- The second step is to construct a weight matrix  $W$  on each local patch  $U_i$  to represent the local intrinsic geometric structure of  $U_i$ .

$$\begin{aligned} \widetilde{W}_{ij} &= e^{-\frac{\|\log(S_i) - \log(S_j)\|_F^2}{t}}, & \text{if } S_j \in U_i, \\ \widetilde{W}_{ij} &= 0, & \text{if } S_j \notin U_i. \end{aligned}$$

The definition of weight value  $\widetilde{W}_{ij}$  is based on the construction of a Laplacian matrix on input SPD matrices from (11).

- The third step is to compute the eigenvalues and the corresponding eigenvectors for the generalized eigenfunction problem:

$$S^T \widetilde{L} S A = \lambda S^T \widetilde{D} S A, \tag{14}$$

where  $S = [\log(S_1), \log(S_2), \dots, \log(S_N)]^T$  is a partitioned matrix.

### 3.3.2 Optimal embedding

The optimal dimension reduction map  $A$  is obtained by minimizing the following energy function:

$$\frac{1}{2} \sum_{i,j} d_G(Y_i, Y_j) \widetilde{W}_{ij}, \quad (15)$$

where  $d_G(Y_i, Y_j)$  is the geodesic distance between  $Y_i$  and  $Y_j$ , while  $\widetilde{W}_{ij}$  is the corresponding weight.

According to the definition of geodesic distance and LEM on the SPD matrix Lie group in (5), the energy function (15) can be transformed into the following equation:

$$\frac{1}{2} \sum_{i,j} \|\log Y_i - \log Y_j\|_F^2 \widetilde{W}_{ij}. \quad (16)$$

According to (12), the optimization function (16) is represented as follows:

$$\begin{aligned} \frac{1}{2} \sum_{i,j} \|\log Y_i - \log Y_j\|_F^2 \widetilde{W}_{ij} &= \frac{1}{2} \sum_{i,j} \|A^T \log S_i A - A^T \log S_j A\|_F^2 \widetilde{W}_{ij} \\ &= \text{tr}(P^T S^T (\widetilde{D} - \widetilde{W}) S P) \\ &= \text{tr}(P^T S^T \widetilde{L} S P), \end{aligned} \quad (17)$$

where  $\widetilde{W}$  and  $\widetilde{D}$  are two sparse  $N \times N$  block matrices,  $\widetilde{W} = [\widetilde{W}_{ij} I_{D \times D}]$ ,  $\widetilde{D} = \text{diag}(\widetilde{D}_{11} I_{D \times D}, \dots, \widetilde{D}_{NN} I_{D \times D})$ .  $P = AA^T \in R^{D \times D}$  and  $\widetilde{L} = \widetilde{D} - \widetilde{W}$  are all semi-SPD matrices. Thus,  $P^T S^T (\widetilde{D} - \widetilde{W}) S P$  is also a semi-SPD matrix, and its eigenvalues are all non-negative. We have  $\text{tr}(P^T S^T \widetilde{L} S P) \geq 0$ . To compute the minimum value of (17), we just need to compute the minimum eigenvalues of matrix  $P^T S^T \widetilde{L} S P$ . To avoid obtaining a singular solution, we impose a constraint as follows:

$$P^T S^T \widetilde{D} S P = I. \quad (18)$$

Then, the corresponding minimization problem turns to

$$\min \text{tr}(P^T S^T \widetilde{L} S P), \quad \text{s.t. } P^T S^T \widetilde{D} S P = I. \quad (19)$$

We use the Lagrange multiplier method to solve the minimization problem:

$$\begin{aligned} L(A, \lambda) &= \text{tr}(P^T S^T \widetilde{L} S P) - \lambda (\text{tr}(P^T S^T \widetilde{D} S P - I)) \\ &= \text{tr}(AA^T \widetilde{L} S AA^T - \lambda (AA^T S^T \widetilde{D} S AA^T - I)). \end{aligned} \quad (20)$$

As  $L$ 's derivative to  $A$ , we obtain the following:

$$\frac{\partial L(A, \lambda)}{\partial A} = 4\text{tr}(A^T S^T \widetilde{L} S AA^T - \lambda A^T S^T \widetilde{D} S AA^T), \quad (21)$$

where  $P = AA^T$  and  $S^T \widetilde{L} S$  are both semi-SPD matrices,  $\text{tr}(S^T \widetilde{L} S AA^T) = \text{tr}(AA^T S^T \widetilde{L} S)$ . Eq. (21) can be derived as

$$\frac{\partial L(A, \lambda)}{\partial A} = 4\text{tr}(A^T AA^T S^T \widetilde{L} S - \lambda A^T AA^T S^T \widetilde{D} S) = 4\text{tr}((S^T \widetilde{L} S A - \lambda S^T \widetilde{D} S A) A^T A), \quad (22)$$

$A$  is a  $D \times d$  full rank matrix; thus,  $A^T A$  is a  $d \times d$  SPD matrix. To obtain  $\frac{\partial L(A, \lambda)}{\partial A} = 0$ , we need to minimize the following generalized eigenfunction problem:

$$S^T \widetilde{L} S A = \lambda S^T \widetilde{D} S A. \quad (23)$$

We obtain the bottom smallest  $d$  eigenvalues  $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_d$ , and the corresponding eigenvectors  $a_1, a_2, \dots, a_d \in \mathbb{R}^{D \times 1}$ .  $A = [a_1, a_2, \dots, a_d] \in \mathbb{R}^{D \times d}$  is the learned linear dimensionality reduction map matrix. The corresponding dimension reduction map  $g$  between  $\mathcal{S}_+^D$  and  $\mathcal{S}_+^d$  is

$$Y_i = g(S_i) = \exp(A^T \log(S_i) A). \quad (24)$$

The lower-dimensional SPD matrix Lie group  $\mathcal{S}_+^d$  preserves the local geometric and algebraic structure of  $\mathcal{S}_+^D$ , which is maintained by the Laplacian matrix  $L$  on  $\mathcal{S}_+^D$ . Good similarity between two points corresponds to a large weight between them. In addition, through the construction of a graph Laplacian matrix on the SPD matrices, the reduction map is learned based on global data points. The Laplacian matrix can be viewed as an alignment matrix that aligns a set of local patch structures to obtain global lower-dimensional representations by solving a generalized eigenfunction. Unlike other methods [15–17], our method uncovers the intrinsic structure of SPD matrices by constructing this discrete Laplacian matrix without the help of other spaces.

## 4 Algorithm analysis

In this section, we mainly analyze the relationships between the proposed Lie-LPP and LPP [8] in theory. In Subsection 4.1, we present dimension reduction error comparisons between these two algorithms. In Subsection 4.2, we analyze the similarity relation between them.

### 4.1 Comparison with LPP

We analyze the reconstruction errors during dimension reduction of Lie-LPP and LPP from two aspects. First, we analyze the local weight matrix construction. Second, we analyze the global alignment matrix and the null space of the alignment matrix. The dimension reduction losses of the two algorithms are determined by the corresponding graph Laplacian matrices defined on data points. In this subsection, we mainly compare Lie-LPP and LPP in theory to analyze the improvements of our algorithm.

First, we show the relationship between two graph Laplacian matrices, which are defined on vector-form data points and SPD matrix-form data points.

The local weight matrix  $W$  of LPP is defined in the second step of the LPP algorithm:

$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{t}},$$

where  $x_i$  and  $x_j$  are in the same neighborhood.

The local weight matrix  $\widetilde{W}$  of Lie-LPP is defined in the second step of the Lie-LPP algorithm:

$$\widetilde{W}_{ij} = e^{-\frac{\|\log(S_i) - \log(S_j)\|_F^2}{t}},$$

where  $S_i$  and  $S_j$  are also in the same neighborhood.

The distance  $\|x_i - x_j\|$  between  $x_i$  and  $x_j$  is computed based on EM. However, the distance  $\|\log(S_i) - \log(S_j)\|_F$  between  $S_i$  and  $S_j$  is computed based on LEM. EM is not the real Riemannian metric of the embedded manifold  $\mathcal{M}$  defined in the LPP algorithm. We mentioned in Subsection 2.2 that LEM is the intrinsic Riemannian metric defined on the SPD matrix Lie group. Thus, the distance  $\|x_i - x_j\|$  under EM is not the intrinsic geodesic distance  $\text{dist}_{\mathcal{M}}(x_i, x_j)$  on  $\mathcal{M}$  and obviously  $\text{dist}_{\mathcal{M}}(x_i, x_j) \geq \|x_i - x_j\|$ . Under LEM, the intrinsic geometric structure of SPD matrix Lie group can be determined, and  $\|\log(S_i) - \log(S_j)\|_F$  is the real geodesic distance between  $S_i$  and  $S_j$ . To our knowledge,  $x_i$  and  $S_i$  are two different feature descriptors of the same image in computer vision. In Appendix A, we prove that the SPD matrix descriptors preserve the geometric structures of vector-form feature descriptors. Thus, we have

$$\|\log(S_i) - \log(S_j)\|_F^2 \geq \|x_i - x_j\|^2.$$

Under this analysis, we have

$$W_{ij} \geq \widetilde{W}_{ij}. \quad (25)$$

The corresponding graph Laplacian matrices defined on  $\{x_i\}$  and  $\{S_i\}, i = 1, 2, \dots, N$  are represented as  $L$  and  $\widetilde{L}$ , respectively. On the basis of the above analysis, we present our first comparison conclusion in Theorem 1.



**Theorem 1.** If the datasets  $\{x_i\}$  and  $\{S_i\}$ , ( $i = 1, 2, \dots, N$ ) are two different feature descriptors of the same  $N$  images, then we have  $L \succeq \tilde{L}$ , that is

$$\lambda_i(L) \geq \lambda_i(\tilde{L}),$$

for all  $i = 1, 2, \dots, N$ .

*Proof.* According to the definition of weight matrices  $\tilde{W}$  and  $W$ , we obtain (25). Then  $D - \tilde{D}$  is a positive diagonal matrix. We know that  $\tilde{L} = \tilde{D} - \tilde{W}$  and  $L = D - W$ ; thus,  $L - \tilde{L} = D - \tilde{D} - (W - \tilde{W})$  is also a Laplacian matrix.  $L - \tilde{L}$  is a semi-positive definite symmetric matrix. We have

$$\lambda_i(L - \tilde{L}) \geq 0, \quad L \succeq \tilde{L},$$

for all  $i = 1, 2, \dots, N$ .

For two symmetric matrices  $L, \tilde{L}$ , if  $L - \tilde{L} \succeq 0$ , then we write  $L \succeq \tilde{L}$ . If  $L \succeq \tilde{L}$ , then

$$\lambda_i(L) \geq \lambda_i(\tilde{L}),$$

for every  $i$ .

For a special situation, if the original sub-manifold  $\mathcal{M}$  in LPP is highly curved and the Riemannian curvature is not zero everywhere, then the eigenvalues of  $L$  are strictly greater than those of  $\tilde{L}$ , that is,  $\lambda_i(L) > \lambda_i(\tilde{L})$  for every  $i$  under this situation. Therefore, we have proven this theorem.

After analyzing the relationship between  $L$  and  $\tilde{L}$  in Theorem 1, we need to analyze the dimension reduction errors of Lie-LPP and LPP. In the dimension reduction step, the two algorithms need to minimize the following generalized eigenvalue function:

$$E = \frac{1}{2} \sum_{i,j} (y_i - y_j)^2 W_{ij} = Y^T LY,$$

where  $\{y_1, y_2, \dots, y_N\}$  are lower-dimensional representations of  $\{x_i\}$  or  $\{S_i\}$ , for  $i = 1, 2, \dots, N$ .

The dimension reduction errors of Lie-LPP and LPP are measured by the smallest eigenvalues of graph Laplacian matrices. Suppose the dimension reduction error under LPP is represented as  $E$  and the error under Lie-LPP is represented as  $\tilde{E}$ . On the basis of Theorem 1, we present our second comparison conclusion in Theorem 2.

**Theorem 2.** The dimension reduction error  $\tilde{E}$  under Lie-LPP is less than the dimension reduction error  $E$  under LPP, that is, we have

$$\|\tilde{E}\|_F \leq \|E\|_F.$$

*Proof.* According to the algorithm procedures of the Laplacian eigenmap, the dimension reduction errors  $E$  of LPP and Lie-LPP are mainly determined by the smallest eigenvalues of the graph Laplacian matrix. Thus, the norm of general reconstruction error  $E$  is measured as

$$\|E\|_F = \sum_{i=1}^d \lambda_i,$$

where  $d$  is the intrinsic dimension of lower-dimensional representations.

From Theorem 1, we can deduce that for the same image database, the graph Laplacian matrix  $\tilde{L}$  constructed on the SPD matrices is lower than that on the vector-form descriptor. Thus, we have  $\lambda_i(L) \geq \lambda_i(\tilde{L})$ , for all  $i = 1, 2, \dots, N$ . Then, on the basis of the definition of the dimension reduction error norm of Lie-LPP and LPP, we have

$$\|\tilde{E}\|_F \leq \|E\|_F.$$

Under the same special situation in Theorem 1, if the embedded manifold  $\mathcal{M}$  in LPP is highly curved, then the reconstruction error  $E$  is strictly greater than  $\tilde{E}$ .

$$\|\tilde{E}\|_F < \|E\|_F.$$

The key reason for this situation is that in LPP, the authors used EM to determine the local intrinsic geometric structure of  $\mathcal{M}$ , which is not the real local Riemannian metric of  $\mathcal{M}$ .

### 4.2 Connection to LPP

We also present a theoretical analysis of the similarity relation between Lie-LPP and LPP aside from reconstruction error comparisons between these two algorithms. Through the analysis, we can see that under the following special situation, Lie-LPP is equivalent to LPP by defining a new weight matrix. Suppose the vector form descriptor of an object is represented as  $x_i$  a row vector. The corresponding SPD matrix descriptor of this object is shown as  $S_i = x_i^T x_i$ . SPD matrix Lie group is a flat Riemannian manifold, where the local neighborhood of the Lie group is locally isometric to the corresponding tangent space. Thus, the local tangent space can be approximately represented by the local neighborhood of the SPD matrix Lie group. Under this special situation and theoretical analysis, Lie-LPP can be transformed into LPP with a special weight matrix. The theoretical analysis is stated in Proposition 1.

**Proposition 1.** The vector form descriptor of an object be  $x_i$  and the corresponding SPD matrix descriptor be  $x_i^T x_i$ . Under this special situation, Lie-LPP can be transformed into LPP by defining a new weight matrix.

*Proof.* First, we give the following representation of the generalized eigenvalue function of Lie-LPP:

$$\begin{aligned} S^T \tilde{L} S A &= \lambda S^T \tilde{D} S A, \\ S^T \tilde{W} S A &= (1 - \lambda) S^T \tilde{D} S A, \end{aligned}$$

where  $\tilde{L} = \tilde{D} - \tilde{W}$  is the corresponding graph Laplacian matrix defined on a set of SPD matrices  $S$  represented as  $S^T = [x_1^T x_1, x_2^T x_2, \dots, x_N^T x_N]$ , and  $S^T$  is the transpose matrix of  $S$ .

By rewriting  $S^T$  as a matrix representation, we obtain the following representation:

$$S^T = [x_1^T, x_2^T, \dots, x_N^T] \begin{pmatrix} x_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N \end{pmatrix}.$$

Under this representation, we obtain

$$S^T \tilde{W} S A = [x_1^T, x_2^T, \dots, x_N^T] \begin{pmatrix} x_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N \end{pmatrix} \tilde{W} \begin{pmatrix} x_1^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N^T \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} A.$$

Define

$$W_V = \begin{pmatrix} x_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N \end{pmatrix} \tilde{W} \begin{pmatrix} x_1^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N^T \end{pmatrix}, \quad D_V = \begin{pmatrix} x_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N \end{pmatrix} \tilde{D} \begin{pmatrix} x_1^T & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_N^T \end{pmatrix}.$$

Under this new weight matrix  $W_V$ , we rewrite  $S^T \tilde{W} S A$  as follows:

$$S^T \tilde{W} S A = [x_1^T, x_2^T, \dots, x_N^T] W_V \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} A.$$

Suppose

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix}, \quad S^T \tilde{W} S A = X^T W_V X A.$$

Thus, under this new weight matrix  $W_V$ , Lie-LPP is equivalent to LPP. The above analysis shows that the Lie-LPP algorithm can be transformed into LPP by defining a new weight matrix  $W_V$  if the feature descriptors of vector form and SPD matrix form are  $\{x_i\}$  and  $\{x_i^T x_i\}$ ,  $i = 1, 2, \dots, N$ , respectively.

## 5 Experiments

In this section, we first report the results we obtained after running Lie-LPP on two human action databases, namely, Motion Capture HDM05 [26] and CMU Motion Graph. We compare our algorithm with traditional manifold learning algorithms through two experiments. In Subsection 5.2, we test our algorithms on a static face database, e.g., extended Yale Face Database B (YFB DB). We compare LEML algorithm [15] and SPD-ML algorithm [17] with our algorithm, after which we present the experimental comparison between Lie-LPP and LPP [8].

### 5.1 Human action recognition

In this subsection, we test Lie-LPP on two human action databases. Each action segment trajectory can be seen as a curve traversing a manifold. Action recognition involves classifying the different action curves. In the recognition step, we use the nearest neighborhood framework to classify human action sequences. After feature extraction, the embedded covariance feature descriptors form an SPD matrix, which belongs to a low-dimensional SPD matrix Lie group.

#### 5.1.1 Motion Capture HDM05 database

The HDM05 database [26] contains more than 70 motion classes in 10 to 50 realizations executed by various actors. The actions are performed by five subjects whose names are ‘bd’, ‘bk’, ‘dg’, ‘mm’, and ‘tr’. For each subject, we choose the following 14 actions: ‘clap above head’, ‘deposit floor’, ‘elbow to knee’, ‘grab high’, ‘hop on both legs’, ‘jog’, ‘kick forward’, ‘lie down on the floor’, ‘rotate both arms backward’, ‘sit down on chair’, ‘sneak’, ‘stand up’, and ‘throw basketball’. We choose three motion fragments per subject per action. Thus, the total number of motion fragments in this experiment is 210. The dataset provides the 3D locations of 31 joints over time acquired at the speed of 120 frames per second. In our experiment, we describe each action observed over  $T$  frames [3] by its joint covariance descriptor, which is an SPD matrix of size  $93 \times 93$ .  $T$  is a parameter. In this experiment, we choose  $T = 20$ .

If we directly perform the classification in the original  $93 \times 93$  dimensional SPD matrix Lie group, the experimental time cost would be as high as up to 21133.9 s. In practice, each joint action is controlled by only a few features, which can be much less than 93. Therefore, dimension reduction is necessary before recognition. We perform eight groups of experiments. Each experiment is divided into two parts. The first part involves using the Lie-LPP algorithm to reduce the dimension of SPD matrices, while the second part involves using the leave-one-out method to compute the recognition rate. We measure the similarity between two SPD matrices under two Riemannian metrics, namely, EM and LEM, for comparison. The final comparison results are shown in Table 1. The purpose of this experiment is to analyze the time complexity of Lie-LPP. Thus, we present comparisons of our method under different Riemannian metrics and different reduced dimensions. Under the same reduced dimensions  $d \times d$ , the accuracy rates under LEM are higher than those under EM overall, but the time cost under LEM is higher than that under EM because of the especially high time cost of the log operation on the SPD matrix. Under the same Riemannian metric, the time cost increases along with the growth of the reduced dimensions. If we do not reduce the dimensionality of SPD matrices before recognition, then the recognition accuracies under both Riemannian metrics are 0.5857 and 0.6000, respectively. However, the recognition rates under the low-dimensional SPD matrix Lie group with LEM are 0.7619, 0.7857, and 0.7905 for dimensions reduced  $10 \times 10$ ,  $15 \times 15$ , and  $20 \times 20$ , respectively (Table 1); these rates are much higher than they would be without dimension reduction. In conclusion, reducing the dimension of the original SPD matrices is necessary before recognition. Our method reduces the time cost of the experiment and improves the

**Table 1** Time comparison, accuracy rates of HDM05 database under different reduced dimensions and different Riemannian metrics, i.e., EM and LEM

Method	Dimension $d \times d$	Time (s)	Accuracy rate
Lie-LPP-EM	$10 \times 10$	16.1961	0.5905
Lie-LPP-EM	$15 \times 15$	21.0645	0.6095
Lie-LPP-EM	$20 \times 20$	30.2843	0.6095
Lie-LPP-LEM	$10 \times 10$	275.156	0.7619
Lie-LPP-LEM	$15 \times 15$	822.289	0.7857
Lie-LPP-LEM	$20 \times 20$	1003.13	0.7905
Lie-LPP-EM	$93 \times 93$	438.777	0.5857
Lie-LPP-LEM	$93 \times 93$	21133.9	0.6000

**Table 2** Classification performance of CMU Motion Graph database together with the comparison results for Lie-LPP and traditional manifold learning algorithms

Method	CMU	Dimension $d$
<b>Lie-LPP</b>	<b>0.9750</b>	$3 \times 3$
LPP [8]	0.8500	9
LEP [21]	0.9000	9
PCA [27]	0.2500	9

recognition rate. In addition, on the basis of the LEM, the recognition rates are relatively higher. Thus, the intrinsic structure of SPD matrices is more accurately determined based on LEM.

### 5.1.2 CMU Motion Graph database

We consider four different action classes, namely, walking, running, jumping, and swing. Each class contains 10 sequences or a total of 40 sequences for our experiment. A total of 31 joints are marked in the skeletal joint sequences. Only the root joint is represented by a 6D vector; the other 30 joints are represented by 3D rotation angle vectors. Each action frame is represented by a 96D vector. The SPD matrix, which is a  $96 \times 96$  matrix, is constructed by computing the covariance of  $T$  frames subwindows, where we choose  $T = 20$ . To guarantee connectivity between subwindows, we take a  $\frac{T}{2}$  frames that overlap between adjacent subwindows, as mentioned in [3].

In this experiment, we use leave-one-out method to compute the recognition rate. Every time, we choose one sequence of each class as the test set and the remaining sequences as the training set. The comparison results in Table 2 show the recognition accuracies for different action classes, where the recognition accuracy of our method is 0.975, which is higher than that of the other three methods. Notably, the SPD matrix descriptor under joint locations in our method obtains a surprising result.

## 5.2 Human face recognition

In [19], we tested Lie-LPP on human face databases, namely, YFB DB and YTC (YouTube celebrity) DB. In this subsection, we further test our algorithm on another static human face database, namely, YFB DB. Only two other algorithms, namely, LEML [15] and SPD-ML [17], are similar to our method in that they reduce the dimensionality of SPD matrix manifold. In this subsection, we report the comparison results for our method with LEML, SPD-ML, the manifold learning algorithm LPP [8], and linear dimensionality reduction PCA [27].

### 5.2.1 Extended Yale Face Database B

YFB DB [28] contains 2414 single-light-source images of 38 individuals each seen under approximately 64 near-frontal images and under different illuminations per individual. For every subject in a particular pose, an image with ambient illumination was captured. The face region in each image is resized to  $32 \times 32$ . We use the raw intensity feature to construct the corresponding SPD matrix, in which we follow [15] to

**Table 3** Classification performance of YFB DB together with the comparison results for Lie-LPP and traditional manifold learning algorithms PCA, LPP, LEML, SPD-ML-Stain, and SPD-ML-Airm<sup>a)</sup>

YFB DB	YFD-trn20/tst40	YFD-trn30/tst30	YFD-trn40/tst20	YFD-trn50/tst10
PCA [27]	46.4 ± 1.8	50.2 ± 2.2	61.8 ± 1.7	63.7 ± 1.7
LPP [8]	48.2 ± 1.5	57.2 ± 2.4	62.8 ± 1.8	69.4 ± 1.4
LEML [15]	46.5 ± 1.6	53.6 ± 2.1	57.3 ± 1.7	72.9 ± 1.6
SPD-Stain [17]	44.6 ± 1.9	52.4 ± 2.1	56.7 ± 1.3	69.4 ± 1.7
SPD-Airm [17]	45.2 ± 1.8	53.1 ± 1.7	58.3 ± 1.6	68.5 ± 1.9
Lie-LPP	<b>50.8 ± 2.4</b>	<b>66.3 ± 1.2</b>	<b>70.2 ± 1.7</b>	<b>73.9 ± 2.1</b>

a) “trn” is the abbreviation of training; “tst” is the abbreviation of testing.

construct the SPD matrix for each image. For this dataset, because the image size is  $32 \times 32$ , the size of the corresponding SPD matrix is  $1024 \times 1024$ .

### 5.2.2 Recognition

In the recognition step, SPD-ML [17] used the affine-invariant metric and the Stein divergence metric to measure the similarities among SPD matrices. LEML [15] used LEM, under which the SPD matrix Lie group is locally isometric to Lie algebra. LPP and PCA used EM to measure the similarities among vector-form data points. For Lie-LPP, we also use LEM to compute the geodesic distances between two SPD matrices. Unlike the other two algorithms, our algorithm aims to construct a Laplace-Beltrami operator on the SPD matrix Lie group and then learn a more discriminable Lie group, which preserves the geometric and algebraic structure of the original one.

For YFB DB, we choose 60 images per subject and run the experiment four times by using each algorithm. In each experiment, we randomly choose  $p$  ( $p = 20, 30, 40, 50$ ) images per subject as the training dataset and the remaining  $60 - p$  images as the test dataset, respectively. For PCA and LPP, we choose the dimension of low dimensional space with  $d(d+1)/2$  after dimension reduction, where  $d = 10$ . For Lie-LPP, LEM, and SPD-ML, we choose the low size of the SPD matrix, that is,  $d \times d$ . Under these choices, the low dimension of PCA and LPP is equal to the low dimension of Lie-LPP. The recognition accuracy results are reported in Table 3. We choose the same classification methods with LEML and SPD-ML in the recognition step. Table 3 shows that the recognition results of Lie-LPP outperform PCA and LPP in any case. Comparisons among Lie-LPP, LEM, and SPD-ML show that Lie-LPP is slightly higher than LEML only when we choose  $p = 50$  images per subject. The accuracy recognition rates of Lie-LPP especially outperform the accuracies of LEML and SPD-ML when we choose  $p = 20, 30, 40$ . These results show that the effect of dimensionality reduction for SPD matrices by Lie-LPP is better than the effects of LEML and SPD-ML. In addition, LEML and SPD-ML need several parameters when performing their algorithms. Our algorithm only needs to analyze the Laplacian operator of the SPD matrix Lie group and solves the dimensionality reduction problem directly on the SPD matrix Lie group.

## 6 Conclusion and future directions

In summary, the following are the main conclusions of this paper.

- (1) We construct a Laplace-Beltrami operator on the SPD matrix Lie group and give the corresponding discrete Laplacian matrix.
- (2) We extend the manifold learning algorithm LPP to Lie-LPP and used it on the SPD Matrix Lie group. We have shown that Lie-LPP can be successfully applied to human action recognition and human face recognition.
- (3) Our analysis of the geometric and algebraic structure of the SPD matrix Lie group shows that the SPD matrix Lie group is a complete and compact manifold. The sectional curvature on every point is zero; thus, the SPD matrix Lie group is a flat manifold and locally isometric to Lie algebra.
- (4) However, this is not a simple application of the idea in paper [8]. We developed a new algorithm called Lie-LPP, which is a substantial extension of the LPP algorithm.

(5) We analyze the relationships between Lie-LPP and LPP in theory and obtain three theoretical conclusions.

(6) We conducted practical experiments, the results of which show that Lie-LPP outperforms the existing ones significantly.

In the future, we will further improve this algorithm. We will attempt to add a time dimension to enhance human action recognition, introduce manifold learning algorithms to other types of Lie groups, and introduce new manifold learning algorithms on higher-dimensional tensor space.

**Acknowledgements** This work was supported by National Key Research and Development Program of China (Grant No. 2016YFB1000902), National Natural Science Foundation of China (Grant Nos. 61232015, 61472412, 61621003), Beijing Science and Technology Project on Machine Learning-based Stomatology, and Tsinghua-Tencent-AMSS-Joint Project on WWW Knowledge Structure and its Application.

**Supporting information** Appendix A. The supporting information is available online at [info.scichina.com](http://info.scichina.com) and [link.springer.com](http://link.springer.com). The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

## References

- 1 Ma A J, Yuen P C, Zou W W W, et al. Supervised spatio-temporal neighborhood topology learning for action recognition. *IEEE Trans Circ Syst Video Technol*, 2013, 23: 1447–1460
- 2 Hussein M E, Torki M, Gawayyed M A, et al. Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations. In: *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, Beijing, 2013. 2466–2472
- 3 Vemulapalli R, Arrate F, Chellappa R. Human action recognition by representing 3D skeletons as points in a Lie group. In: *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, 2014
- 4 Tuzel O, Porikli F, Meer P. Region covariance: a fast descriptor for detection and classification. In: *Proceedings of the 7th European Conference on Computer Vision: Part II*, Graz, 2006. 589–600
- 5 Wang L, Suter D. Learning and matching of dynamic shape manifolds for human action recognition. *IEEE Trans Image Process*, 2007, 16: 1646–1661
- 6 Roweis S, Saul L. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000, 290: 2323–2326
- 7 Tenenbaum J B, Silva V, Langford J. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000, 290: 2319–2323
- 8 He X F, Niyogi P. Locality preserving projections. In: *Proceedings of the 16th Annual Conference on Neural Information Processing Systems*, Chicago, 2003
- 9 He X F, Cai D, Niyogi P. Tensor subspace analysis. In: *Proceedings of the 18th Annual Conference on Neural Information Processing Systems*, British Columbia, 2005. 499–506
- 10 Porikli F, Tuzel O. Fast construction of covariance matrices for arbitrary size image windows. In: *Proceedings of the International Conference on Image Processing*, Atlanta, 2006. 1581–1584
- 11 Tuzel O, Porikli F, Meer P. Pedestrian detection via classification on Riemannian manifolds. *IEEE Trans PAMI*, 2008, 30: 1713–1727
- 12 Porikli F, Tuzel O, Meer P. Covariance tracking using model update based on Lie algebra. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, 2006
- 13 Sanin A, Sanderson C, Harandi M T, et al. Spatio-temporal covariance descriptors for action and gesture recognition. In: *Proceedings of the 2013 IEEE Workshop on Applications of Computer Vision*, Washington, 2013. 103–110
- 14 Tabia H, Laga H, Picard D, et al. Covariance descriptors for 3D shape matching and retrieval. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, 2014
- 15 Huang Z W, Wang R P, Shan S G, et al. Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification. In: *Proceedings of the 32nd International Conference on Machine Learning*, Lille, 2015. 720–729
- 16 Harandi M, Sanderson C, Wiliem A, et al. Kernel analysis over Riemannian manifolds for visual recognition of actions, pedestrians and textures. In: *Proceedings of the IEEE Workshop on the Applications of Computer Vision*, Breckenridge, 2012
- 17 Harandi M, Salzmann M, Hartley R. From manifold to manifold: geometry-aware dimensionality reduction for SPD matrices. In: *Proceedings of the 15th European Conference on Computer Vision*, Zurich, 2014. 17–32
- 18 Kwatra V, Han M. Fast covariance computation and dimensionality reduction for sub-window features in image. In: *Proceedings of the 11th European Conference on Computer Vision: Part II*, Heraklion, 2010. 156–169
- 19 Li Y Y. Locally preserving projection on symmetric positive definite matrix Lie group. In: *Proceedings of the International Conference on Image Processing*, Beijing, 2017
- 20 Arsigny V, Fillard P, Pennec X, et al. Geometric means in a novel vector space structure on symmetric positive-definite

- matrices. *SIAM J Matrix Anal Appl*, 2007, 29: 328–347
- 21 Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In: *Proceedings of the International Conference Advances in Neural Information Processing Systems*, Vancouver, 2001. 585–591
  - 22 Willmore T. *Riemannian Geometry*. Oxford: Oxford University Press, 1997
  - 23 Belkin M, Niyogi P. Towards a theoretical foundation for Laplacian-based manifold methods. *J Comput Syst Sci*, 2005, 74: 1289–1308
  - 24 Li X, Hu W M, Zhang Z F, et al. Visual tracking via incremental Log-Euclidean Riemannian subspace learning. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, 2008
  - 25 Pennec X, Fillard P, Ayache N. A Riemannian framework for tensor computing. *Int J Comput Vision*, 2006, 66: 41–66
  - 26 Muller M, Roder T, Clausen M, et al. Documentation: Mocap Database HDM05. *Computer Graphics Technical Reports CG-2007-2*. 2007
  - 27 Smith L. A tutorial on principal components analysis. 2002. arXiv:1404.1100
  - 28 Yale University. Face database. <http://cvc.yale.edu/projects/yalefaces/yalefaces/html>