

# Group Action Induced Distances for Averaging and Clustering Linear Dynamical Systems with Applications to the Analysis of Dynamic Scenes

Bijan Afsari<sup>1</sup>   Rizwan Chaudhry<sup>1</sup>   Avinash Ravichandran<sup>2</sup>   René Vidal<sup>1</sup>

<sup>1</sup> Center for Imaging Science, Johns Hopkins University

<sup>2</sup> Vision Lab, University of California, Los Angeles

## Abstract

*We introduce a framework for defining a distance on the (non-Euclidean) space of Linear Dynamical Systems (LDSs). The proposed distance is induced by the action of the group of orthogonal matrices on the space of state-space realizations of LDSs. This distance can be efficiently computed for large-scale problems, hence it is suitable for applications in the analysis of dynamic visual scenes and other high dimensional time series. Based on this distance we devise a simple LDS averaging algorithm, which can be used for classification and clustering of time-series data. We test the validity as well as the performance of our group-action based distance on synthetic as well as real data and provide comparison with state-of-the-art methods.*

## 1. Introduction

Analysis of dynamic scenes is an important area of computer vision. Some examples of dynamic scenes are videos of non-rigid objects such as water, fire, flags fluttering in the air, *etc.* (collectively called dynamic textures), videos of humans performing different actions and videos of lip articulations. A number of state-of-the-art techniques use Linear Dynamical Systems (LDSs) to model the temporal evolution of visual features in dynamic scenes. Furthermore, in template tracking applications, the most common motion models are based on either LDSs or linearized versions of more general, nonlinear dynamical systems. Some of the most important applications of LDS in computer vision have been in synthesis [11, 20], segmentation [6, 18], registration [19] and recognition [11, 4, 5, 17, 3, 8, 21] of human gaits, dynamic textures, face motions and lip articulations. Most of these applications, especially recognition, require comparison of test data with training data using the notion of a distance between LDSs. A 1-NN classifier, for example, computes the distance of the test LDS to all the training LDSs and assigns the class label of the nearest point in the training set. As the size of the training data increases,

computing distances to all training data becomes computationally expensive. If however, a representative point for a certain class of LDSs were computable, the recognition problem would be drastically simplified to finding distances only to these representative points and assigning the label of the closest class representative.

One such representative point is the *mean* of the training samples belonging to a certain class. The space of LDSs, however, is not a Euclidean space and has a complicated manifold structure as we will discuss in detail in this paper. Computing means on this space is therefore not straightforward. In fact, to the best of our knowledge, there is not a single approach that correctly models the manifold structure of multiple-input and multiple-output LDSs and computes statistics on this space. Recent attempts by Turaga *et al.* [21] and Chaudhry *et al.* [7] (roughly) embed the space of LDSs in an arbitrarily large Grassmann manifold and perform averaging in that space as it has a rather computationally friendly Riemannian structure. However, as we will describe in this paper, these approaches have some theoretical and computational drawbacks. Alternatively, approaches such as [17] approximate the mean of the LDSs by computing a sample point that is closest to the mean under certain cord distances.

In this paper, we will formalize the notion of computing means on a set of LDSs. We will perform a theoretical analysis of the methods in [21, 7] and discuss the inherent limitations that have not been addressed by these authors. We will then propose some changes to fix these limitations. Our main contribution is the definition of a new method for computing distances between LDSs based on the notion of aligning LDSs. This allows us to propose a novel iterative approach for computing the mean of a set of LDSs. Our approach is computationally efficient because it requires neither creating arbitrarily large matrices, as in the approaches of [21, 7], nor computing an all-pair training data distance matrix as in [17].

The rest of the paper is structured as follows. In §2, we will review some of the preliminaries for LDS modeling and summarize the state-of-the-art algorithms for comput-

ing means of LDSs. We will then describe some of the technical issues with these approaches, propose corrections, and motivate the need for a more formal analysis of the space of LDSs and a proper definition of a distance for LDSs. In §3, we will propose our method for computing a distance and average of LDS. We will test our approach on synthetic and real data in §4 and give concluding remarks in §5.

## 2. Prior Work: Limitations and Corrections

### 2.1. Linear Dynamical Systems

Stochastic Linear Dynamical Systems are a class of dynamical systems that satisfy the following equations:

$$\begin{aligned} \mathbf{x}_t &= A\mathbf{x}_{t-1} + B\mathbf{v}_t \\ \mathbf{y}_t &= \boldsymbol{\mu} + C\mathbf{x}_t + \mathbf{w}_t, \end{aligned} \quad \forall t = 0, 1, 2, \dots \quad (1)$$

Here  $\mathbf{y}_t \in \mathbb{R}^p$  is the  $p$ -dimensional stochastic output of the LDS at time  $t$ . For example, in the case of dynamic textures,  $\mathbf{y}_t$  would represent the stacked vector of image intensities in a patch or frame. In the case of human activity analysis,  $\mathbf{y}_t$  would be an orientation histogram or joint angles, *etc.*  $\mathbf{x}_t \in \mathbb{R}^n$  is the hidden state of the LDS at time  $t$ . The dimension,  $n$ , of  $\mathbf{x}_t$  is also referred to as the order of the LDS. The output is a linear transformation of the state under the *observation* matrix,  $C \in \mathbb{R}^{p \times n}$ .  $\mathbf{x}_0$  is the initial state of the system and the current state,  $\mathbf{x}_t$ , is linearly related to the previous state,  $\mathbf{x}_{t-1}$ , by the *dynamics* matrix,  $A \in \mathbb{R}^{n \times n}$ .  $\boldsymbol{\mu} \in \mathbb{R}^p$  is the mean output of the dynamical system.  $\mathbf{v}_t$  is the process noise and  $\mathbf{w}_t$  is the output noise. It is generally assumed that both the process and output noise are white Gaussian and independent. Specifically,  $\mathbf{v}_t \sim \mathcal{N}(0, I_{m \times m})$  and  $\mathbf{w}_t \sim \mathcal{N}(0, \Omega)$ . Here  $I_{n \times n}$  is the  $n$ -dimensional identity matrix. Usually, if the output dimension,  $p$ , is very large, it is assumed that  $\Omega = \sigma^2 I_{p \times p}$ .

**Equivalent representations and the action of  $GL(n)$ .** We call the tuple  $R_F = (\mathbf{x}_0, \boldsymbol{\mu}, A, B, C, \Omega)$  a full representation or realization of the LDS described by Eq. (1), and we call  $R = (A, B, C)$  a (dynamics) representation. A system has an equivalent class of representations. The output of the system described by Eq. (1) does not change (in the *deterministic* sense, see below) under state change of basis, that is, if one replaces the full representation  $R_F$  by

$$P \cdot R_F = (P^{-1}\mathbf{x}_0, \boldsymbol{\mu}, P^{-1}AP, P^{-1}B, CP, \Omega), \quad (2)$$

for every matrix  $P \in GL(n)$ , where  $GL(n)$  is the group of non-singular  $n \times n$  matrices. More formally,  $GL(n)$  acts on the set of all representations of LDSs of  $m$  inputs, order  $n$  and  $p$  outputs as above and each system has a family of equivalent representations related by this action. This simple fact makes naive comparison of two full representations  $R_1$  and  $R_2$ , *e.g.*, by measuring the norm of  $R_1 - R_2$  inadequate. In §3 we give a systematic way of comparing two representations that takes into account the group action.

**Deterministic vs. stochastic equivalence.** In addition to the above deterministic invariance, notice that (under our assumptions about (1)) if  $B$  is transformed to  $B\Theta$  where  $\Theta$  is any orthogonal  $m \times m$  matrix, then the output will not change in the *stochastic* sense. We call this *stochastic equivalence*. Although our framework can be extended to full representations  $(\mathbf{x}_0, \boldsymbol{\mu}, A, B, C, \Omega)$  as well as stochastic equivalence, in this paper we only focus on comparing representations  $(A, B, C)$  under the above (deterministic) state change of basis equivalence. In fact, at the end we just specialize to comparing the dynamic-observation pairs  $(A, C)$ . This will not put our method in disadvantage against existing methods (*e.g.*, [11, 5, 17, 3, 8, 9]), as all these methods achieve acceptable performance by merely comparing the partial representations  $(A, C)$ . Ignoring the stochastic equivalence, however, could lead to a larger error than just comparing the pairs  $(A, C)$  under deterministic equivalence (because *e.g.*, in the stochastic setting  $(A, -B, C)$  and  $(A, B, C)$  are the same while in the deterministic setting they are quite “afar”). Therefore, comparing the triples  $(A, B, C)$  by taking into account the stochastic equivalence is the correct approach.

**Stable and observable LDS.** We are mainly interested in asymptotically stable LDSs, *i.e.*, where  $\|A\|_2 < 1$  ( $\|A\|_2$  is the 2-norm (or spectral norm) of  $A$ ). We call  $O_{n,p,k}(A, C) = [C^\top, (CA)^\top, (CA^2)^\top, \dots, (CA^{k-1})^\top]^\top$  the observability matrix of length  $k$ . An LDS  $M$  is called *observable* if for a representation  $(A, B, C)$  of  $M$  the observability matrix  $O_{n,p,n}(A, C)$  has rank  $n$ .

**System identification.** Finding the parameters of model (1) based on observed data is a well-studied problem known as *system identification*. In computer vision applications such as dynamic texture modeling, a sub-optimal but efficient method proposed by [11] is preferred over optimal but computationally demanding alternatives. An important feature of this method is that the matrix  $C$  has orthonormal columns, *i.e.*,  $C$  belongs to the (compact) *Stiefel* manifold  $ST(n, p)$  defined as

$$ST(n, p) = \{C \in \mathbb{R}^{p \times n} | C^\top C = I_{n \times n}\}. \quad (3)$$

### 2.2. Existing Distances in the Space of LDSs

Throughout this paper, by the space of LDSs we implicitly mean the set of all (or a subset of all) LDSs of *fixed* number of inputs  $m$ , number of outputs  $p$ , and order  $n$ . That is, we are only interested in comparing systems of the same dimensions and order. Defining a notion of a distance between two LDSs has a rather long history in the control systems community [14]. Most of the approaches (*e.g.*, the Riemannian framework in [14]) result in computationally intensive methods (especially for large  $p$ ). Recently, more computationally efficient methodologies have been proposed (with limited versatility though). For

example, Martin [16] defined a distance between single-input single-output (SISO) systems (*i.e.*,  $m = p = 1$ ) described by transfer functions. The Martin distance enjoys convolution-invariance, which is very important from a system-theoretic point of view. In [10], the computation of the Martin distance is addressed for SISO systems described in state-space form. For AR models, the Martin distance is computed in terms of subspace angles between the infinite observability subspaces (*i.e.*,  $O_{1,1,\infty}(A, C)$ ). In computer vision and machine learning communities, other approaches for comparing LDS have been proposed, such as Binet-Cauchy kernels [22] and KL-divergence [4]. However these approaches do not seem to be theoretically or computationally conducive to a notion of average over the space of LDSs.

### 2.3. Distances Based on Observability Matrix and Grassmann Embedding

An alternative approach to calculating distances between LDS is to embed a subset of the parameters in an ambient space and calculate the distance associated with the ambient space. One such example is the *Grassmann embedding*. We assume that the observability matrices,  $O_{n,p,k}(A, C)$  are of rank  $n$  for some fixed  $k \geq 2$  which is true for a general observable LDS at  $k = n$ . Furthermore, in our applications of interest we have  $p \geq n$  and  $C \in \text{ST}(n, p)$ , hence,  $O_{n,p,k}(A, C)$  is full rank at  $k = 1$ . Although for different  $P$  the observability matrices generated by  $P \cdot (A, C)$  are different, the range of the observability matrix,  $\mathcal{R}(O_{n,p,k}(A, C))$ , is the same for all  $P$ . Therefore, we can identify  $\mathcal{R}(O_{n,p,k}(A, C))$  with a point in  $\text{Gr}(n, kp)$ , where  $\text{Gr}(r, q)$  is the Grassmann manifold of  $r$ -dimensional subspaces of  $\mathbb{R}^q$ . Note that if  $\mathcal{R}(O_{n,p,k}(A_1, C_1)) = \mathcal{R}(O_{n,p,k}(A_2, C_2))$ , then there is  $P \in \text{GL}(n)$  such that  $A_1 = P^{-1}A_2P$  and  $C_1 = C_2P$ . This means that a distance in  $\text{Gr}(r, q)$  induces a distance on the (partial) LDSs represented by pairs  $(A, C)$ . The Grassmannian  $\text{Gr}(r, q)$  can be equipped with a standard Riemannian structure, which results in the standard Riemannian distance denoted by  $d_{\text{Gr}}(\cdot, \cdot)$  [12]. Therefore, using the above embedding we can now calculate the *Grassmann embedding* or *observability subspace partial distance* [10, 21, 9]. We call it a partial distance since it only compares pairs  $(A, C)$ .

### 2.4. Averaging in the Space of LDSs

The first step in computing the mean of a set of LDSs is to have a proper notion of distance between LDSs. Given the difficulty of defining and computing such a distance, state-of-the-art methods for averaging LDSs either embed the space of (partial) LDSs in a Riemannian manifold with computationally feasible properties, or use approximate methods to find an LDS from the samples that is the closest to the “true mean.” We briefly analyze these methods below.

**Averaging based on the Grassmann embedding.** The first approach to averaging LDSs presented in [21, 7] uses the Grassmann embedding distance described in §2.3. To be more concrete, let us define the observability *submanifold* in  $\text{Gr}(n, kp)$  as

$$\mathcal{OS}(n, p, k) = \{\mathcal{R}(O_{n,p,k}(A, C)) \mid (A, C) \text{ is observable}\}. \quad (4)$$

Given a set of partial representations  $\{(A_i, C_i)\}_{i=1}^N$ , for every  $i$  we find,  $U_i$ , an orthogonal basis for  $\mathcal{R}(O_{n,p,k}(A_i, C_i))$  ( $U_i$  is identified with an element of  $\mathcal{OS}(n, p, k) \subset \text{Gr}(n, kp)$ ). The methods proposed in [21, 7], then use the Riemannian geometry of the ambient space  $\text{Gr}(n, kp)$  to find  $\bar{U} \in \text{Gr}(n, kp)$ , the Riemannian average or mean of  $\{U_i\}_{i=1}^N$ . Although this approach has already been applied with acceptable performance for classification problems [21, 7], this method has several drawbacks. First, it is not clear that  $\bar{U}$  coincides with the range of an observability matrix  $\mathcal{R}(O_{n,p,k}(\bar{A}, \bar{C}))$  *i.e.*,  $\bar{U}$  might fall out of  $\mathcal{OS}(n, p, k)$ . Therefore, we propose to perform an extra step of projecting<sup>1</sup> onto the submanifold of observability subspaces (*i.e.*,  $\mathcal{OS}(n, p, k)$ ). We call this method *projected Grassmann averaging*. This method is analogous to finding the so-called *extrinsic mean*, except that in the case of the standard extrinsic mean the ambient space is a Euclidean space [2]. Our experiments in §4 suggest that although the “falling out” might not cause huge errors, it still introduces some form of bias. The second drawback with the Grassmann embedding is that in many applications, typically we have  $p \approx 10,000$  and  $n \approx k \approx 5 - 20$ , which results in huge matrices  $U_i$ . In many circumstances, storing and manipulating such matrices is very time-consuming and the computations could even become numerically inaccurate. Another issue with this approach is that  $B$  or the initial conditions cannot be included in the distance computation. Work by [10] extends the Martin distance to (SISO) ARMA models, which account for  $B$ . However, the computation requires solving algebraic Riccati equations with very large matrices, which becomes too expensive for computer vision applications.

**Approximate averaging.** An approximate method for computing a sample (partial) LDS closest to the mean of the samples was proposed by Ravichandran *et al.* [17]. In this method, the all-pair Martin distance matrix,  $D_M = [d_M(M_i, M_j)]$  is computed from all the samples. Multi-Dimensional Scaling (MDS) is then performed to project the LDSs  $\{M_i\}_{i=1}^N$  to a low-dimensional Euclidean space to get  $\{z_i\}_{i=1}^N$ . The mean,  $\bar{z}$  is then computed in the low-dimensional Euclidean space. The LDS  $M_j$  where  $j = \arg \min_k \|z_k - \bar{z}\|^2, k = 1, \dots, N$  is then chosen as the

<sup>1</sup>A (suboptimal) way to do this projection is to use  $\bar{C} = \bar{U}(1 : p, :)$ , *i.e.*, the first  $p$  rows of  $\bar{U}$ . Then  $\bar{A}$  can be computed using least-squares as  $\bar{A} = \arg \min_A \|\bar{U}(1 : (k-1)p, :)A - \bar{U}(p+1 : kp, :)\|_F^2$ .

mean. Even though this method is computationally faster than the Grassmann embedding, it does not scale with the number of points, because computing the all-pair distance matrix for a very large number of points is not computationally tractable.

### 3. Group Action Induced Distances, Averaging and Clustering in the Space of LDSs

In this section, we propose a new distance for comparing dynamical models and show how this distance can be used for averaging and clustering a set of LDSs.

#### 3.1. Group Action Induced Distances for LDSs

We denote by  $\mathcal{L}_{m,n,p}$  the set of all representations of asymptotically stable observable dynamical systems of  $m$  inputs, order  $n$ , and  $p$  outputs, that is

$$\mathcal{L}_{m,n,p} = \{(A, B, C) \in \mathbb{R}^{n \times n} \times \mathbb{R}^{n \times m} \times \mathbb{R}^{p \times n} \mid \|A\|_2 < 1 \text{ and } (A, B, C) \text{ is observable}\}. \quad (5)$$

In analogy with *shape analysis theory*, we also call the space of all state space parameterizations  $\mathcal{L}_{m,n,p}$  the *pre-LDS* space.  $GL(n)$  acts on  $\mathcal{L}_{m,n,p}$  via the change of coordinates

$$P \cdot R \equiv P \cdot (A, B, C) = (P^{-1}AP, P^{-1}B, CP), \quad (6)$$

where  $P \in GL(n)$ . The space of observable asymptotically stable LDSs of  $m$  inputs, order  $n$ , and  $p$  outputs is defined as the quotient space of  $\mathcal{L}_{m,n,p}$  with respect to the above action of  $GL(n)$ :

$$\mathcal{M}_{m,n,p} = \mathcal{L}_{m,n,p}/GL(n). \quad (7)$$

We choose to equip  $\mathcal{L}_{m,n,p}$  with the following distance based on the matrix Frobenius norm  $\|\cdot\|_F$ :

$$d_{\mathcal{L}_{m,n,p}}^2(R_1, R_2) = \|C_1 - C_2\|_F^2 + \lambda_A \|A_1 - A_2\|_F^2 + \lambda_B \|B_1 - B_2\|_F^2, \quad (8)$$

where  $R_i = (A_i, B_i, C_i)$  for  $i = 1, 2$  and  $\lambda_A, \lambda_B > 0$  are weights. Strictly positive weights are required to make  $d_{\mathcal{L}_{m,n,p}}$  a distance. It is easy to see that  $\mathcal{L}_{m,n,p}$  has a structure of a smooth manifold. It is known that  $\mathcal{M}_{m,n,p}$  is a smooth manifold (the observability of elements of  $\mathcal{M}_{m,n,p}$  is crucial for this [14]). Inspired by shape analysis theory [23, Chapter 12], we could attempt to define a distance  $d_{\mathcal{M}_{m,n,p}}(M_1, M_2)$  on  $\mathcal{M}_{m,n,p}$  via

$$d_{\mathcal{M}_{m,n,p}}^2(M_1, M_2) = \inf_{P_1, P_2 \in GL(n)} d_{\mathcal{L}_{m,n,p}}^2(P_1 \cdot R_1, P_2 \cdot R_2), \quad (9)$$

where  $R_i = (A_i, B_i, C_i)$  is any representation of  $M_i \in \mathcal{M}_{m,n,p}$  for  $i = 1, 2$ . Unfortunately,  $d_{\mathcal{M}_{m,n,p}}(\cdot, \cdot)$  might

not obey the triangle inequality since  $d_{\mathcal{L}_{m,n,p}}$  is not a  $GL(n)$ -invariant distance [23, Chapter 12]. For such technical and computational difficulties (stemming from  $GL(n)$  being non-compact), we leave this approach to further research. However, if we restrict the representations to a smaller subset of  $\mathcal{L}_{m,n,p}$ , then a rather simple computational framework emerges.

#### 3.1.1 Align distance for LDSs with orthogonal $C$

Let  $\mathcal{OL}_{m,n,p} = \{(A, B, C) \in \mathcal{L}_{m,n,p} \mid C^\top C = I_{n \times n}\}$ , where  $p \geq n$  (for dynamic scene applications in fact  $p \gg n$ ). Any system that has a representation with full-rank  $C$  has a family of equivalent representations in  $\mathcal{OL}_{m,n,p}$ . An important fact is that if  $(A, B, C)$  and  $P \cdot (A, B, C)$  are two equivalent representations in  $\mathcal{OL}_{m,n,p}$ , then  $P$  must be orthogonal. Therefore, if we restrict ourselves to the submanifold  $\mathcal{OL}_{m,n,p} \subset \mathcal{L}_{m,n,p}$ , we can assume that  $GL(n)$  acts on it only via orthogonal transformations. Next, we define  $\mathcal{OM}_{m,n,p}$  as the quotient space of  $\mathcal{OL}_{m,n,p}$  under the action of  $O(n) \subset GL(n)$ , the subgroup of orthogonal matrices in  $GL(n)$ . The action of  $O(n)$  on  $\mathcal{OL}_{m,n,p}$  is a free action, i.e.,  $Q \cdot (A, B, C) = (A, B, C)$  implies  $Q = I_{n \times n}$ , because  $C$  is full-rank and  $p \geq n$ . It follows from a general result in differential geometry that  $\mathcal{OL}_{m,n,p}$  is a smooth manifold [15, Theorem 9.16].

Since  $\mathcal{OL}_{m,n,p}$  is a subset of  $\mathcal{L}_{m,n,p}$ , we endow it with the same distance, i.e.,  $d_{\mathcal{OL}_{m,n,p}} \equiv d_{\mathcal{L}_{m,n,p}}$ . For representations  $R_i = (A_i, B_i, C_i)$  ( $i = 1, 2$ ) of two systems  $M_1, M_2 \in \mathcal{OM}_{m,n,p}$ , we define the distance between the two systems as

$$d_{\mathcal{OM}_{m,n,p}}^2(M_1, M_2) = \min_{Q \in O(n)} d_{\mathcal{OL}_{m,n,p}}^2(Q \cdot R_1, R_2). \quad (10)$$

Here, the role of  $P_1, P_2$  in (9) has been lumped into one matrix  $Q$  (due to unitary invariance of  $\|\cdot\|_F$ ). It is easy to verify that  $d_{\mathcal{OM}_{m,n,p}}(\cdot, \cdot)$  is a distance in  $\mathcal{OM}_{m,n,p}$ .

Alternatively, we could define a Riemannian metric on  $\mathcal{OL}_{m,n,p}$  and based on that define a Riemannian metric on  $\mathcal{OM}_{m,n,p}$ , compute Riemannian distance and geodesic between two elements in  $\mathcal{OL}_{m,n,p}$ , and finally define a Riemannian mean or average.

**Pseudo distance**  $d_{\mathcal{OM}_{m,n,p}}$ . If we set  $\lambda_B = 0$ ,  $d_{\mathcal{OM}_{m,n,p}}$  is only a pseudo distance in  $\mathcal{OM}_{m,n,p}$ , which we denote by  $d_{\mathcal{OM}_{m,n,p}}$ . Despite this drawback  $d_{\mathcal{OM}_{m,n,p}}$  enables us to compare the dynamics-observation part of two systems in  $\mathcal{OM}_{m,n,p}$  with more freedom which seems especially useful for our applications in which stochastic equivalence is important (see §2.1).

#### 3.1.2 Computing align distances by gradient descent

Let us define  $f(Q) = d_{\mathcal{OL}_{m,n,p}}^2(Q \cdot R_1, R_2)$ . To find the distance between two systems  $M_1, M_2 \in \mathcal{OM}_{m,n,p}$  with

---

**Algorithm 1 (LDS Align)**


---

1. Input:  $\{R_i = (A_i, B_i, C_i)\}_{i=1}^N \subset \mathcal{OM}_{m,n,p}$ .
  2. Set  $\gamma > 0$  and choose  $Q_+^0 \in SO(n)$  and  $Q_-^0 \in O^-(n)$ .
  3. repeat until convergence:
    - $Q_+^{l+1} = Q_+^l \exp(-\gamma(Q_+^l)^\top \nabla f(Q_+^l))$
    - $Q_-^{l+1} = Q_-^l \exp(-\gamma(Q_-^l)^\top \nabla f(Q_-^l))$
  4. if  $f(Q_+^{l+1}) < f(Q_-^{l+1})$  then  $Q = Q_+^{l+1}$  else  $Q = Q_-^{l+1}$
  5. Output:  $Q$  and  $f(Q) = d_{\mathcal{OL}_{m,n,p}}^2(Q \cdot R_1, R_2)$ .
- 

representations  $R_1$  and  $R_2$ , we need to minimize  $f$  in  $O(n)$ . We call this process aligning  $R_1$  w.r.t.  $R_2$ . Notice that if  $Q$  aligns  $R_1$  w.r.t.  $R_2$ , then  $Q^\top$  aligns  $R_2$  w.r.t.  $R_1$ . There are several possible approaches to solve this problem. Here we choose the simplest one: Riemannian gradient descent [1]. We equip  $O(n)$  with the standard left invariant Riemannian metric

$$\langle \eta_1, \eta_2 \rangle_Q = \text{tr}((Q^\top \eta_1)^\top Q^\top \eta_2) = \text{tr}(\Delta_1^\top \Delta_2), \quad (11)$$

where  $\eta_i = Q \Delta_i$  ( $i = 1, 2$ ) and  $\Delta_i$  is an  $n \times n$  skew-symmetric matrix which represents a tangent vector to  $O(n)$  at  $Q$ . Here  $\text{tr}(\cdot)$  is the matrix trace function. Next, based on the definition of gradient with respect to this Riemannian metric and some simple algebraic manipulations, one can see that  $\nabla f$ , the gradient of  $f$ , is given by

$$\begin{aligned} \nabla f(Q) = & -Q((Q^\top C_1^\top C_2 - C_2^\top C_1 Q) - \lambda_A([Q^\top A_1 Q, A_2^\top] \\ & - [Q^\top A_1 Q, A_2^\top]^\top) - \lambda_B(Q^\top B_1 B_2^\top - B_2 B_1^\top Q)), \end{aligned} \quad (12)$$

where  $[X, Y] = XY - YX$  is the matrix Lie bracket.

Knowing  $\nabla f$  enables us to run a gradient descent algorithm for alignment and finding the distance. However, recall that  $O(n)$  has two disjoint connected components  $O^+(n)$  (also denoted as  $SO(n)$ ) and  $O^-(n)$ , which comprise of the elements of  $O(n)$  with determinant  $+1$  and  $-1$ , respectively. Therefore, we run two parallel gradient descent algorithms with initial conditions in  $SO(n)$  and  $O^-(n)$  and choose the one which gives smaller  $f$  as the solution to (10). Algorithm 1 gives a summary of a simple constant step-size gradient descent algorithm for minimizing  $f$  on  $O(n)$ . Here  $\exp(\cdot)$  is the matrix exponential. In this algorithm once we compute  $C_1^\top C_2$  all computations can be done by matrices of size  $n \times n$  (independent of  $p$ ). The algorithm can be implemented with transformations that are cheaper to compute than  $\exp(\cdot)$  [1]. In practice one might need to use more sophisticated step-size rules (such as Armijo's rule) and stopping criteria.

### 3.2. A Procrustean mean for LDSs (Align + average)

Consider  $N$  LDS  $\{M_i\}_{i=1}^N \subset \mathcal{OM}_{m,n,p}$  with representations  $\{R_i = (A_i, B_i, C_i)\}_{i=1}^N$ . We want to define an aver-

age LDS  $\bar{M}$  with representation  $\bar{R} = (\bar{A}, \bar{B}, \bar{C})$  as

$$\bar{M} = \arg \min_{M \in \mathcal{OM}_{m,n,p}} \frac{1}{N} \sum_{i=1}^N d_{\mathcal{OM}_{m,n,p}}^2(M_i, M). \quad (13)$$

In the pre-LDS space  $\mathcal{OL}_{m,n,p}$  this translates into

$$\bar{R} = \arg \min_{R \in \mathcal{OL}_{m,n,p}} \frac{1}{N} \sum_{i=1}^N \min_{Q_i \in O(n)} d_{\mathcal{OL}_{m,n,p}}^2(Q_i \cdot R_i, R). \quad (14)$$

This results in a simple *align-and-average* algorithm, which is in fact a coordinate descent algorithm in  $\mathcal{OL}_{m,n,p} \times O(n)^N$ , and consists of alignment steps followed by an averaging step. We start with an initial representation of the average system  $\bar{R}^0 = (\bar{A}^0, \bar{B}^0, \bar{C}^0)$ . In each iteration of the algorithm, we align each  $R_i$  to  $\bar{R}^l$  by Algorithm 1, which gives the aligning  $Q_i^l$ . We then average the aligned representations  $\{Q_i^l \cdot R_i\} \subset \mathcal{OL}_{m,n,p}$ . This averaging is w.r.t. the distance  $d_{\mathcal{OL}_{m,n,p}}(\cdot, \cdot)$  and has a simple form. In fact, once the alignment is performed, the averaging problem decouples to three separate averaging sub-problems. For  $\bar{A}^{l+1}$  and  $\bar{B}^{l+1}$  the solution is the simple Euclidean average:

$$\bar{A}^{l+1} = \frac{1}{N} \sum_{i=1}^N (Q_i^l)^\top A_i Q_i^l, \quad \bar{B}^{l+1} = \frac{1}{N} \sum_{i=1}^N (Q_i^l)^\top B_i. \quad (15)$$

Notice that if  $\|A_i\|_2 < 1$  ( $1 \leq i \leq N$ ), then  $\|\bar{A}^{l+1}\|_2 < 1$ ; therefore, the average system, in fact, is also asymptotically stable. To find  $\bar{C}^{l+1}$ , since we are using the standard embedding of  $\text{ST}(n, p)$  in  $\mathbb{R}^{p \times n}$ , we notice that  $\bar{C}^{l+1}$  solves

$$\bar{C}^{l+1} = \arg \min_{C \in \text{ST}(n,p)} \frac{1}{N} \sum_{i=1}^N \|C - C_i Q_i^l\|_F^2. \quad (16)$$

This means that  $\bar{C}^{l+1}$  is the so-called extrinsic mean of  $\{C_i Q_i^l\} \subset \text{ST}(p, n)$ . It is easy to see that the extrinsic mean equals the projection of the Euclidean mean to  $\text{ST}(n, p)$ ,

$$\bar{C}^{l+1} = \text{Proj}_{\text{ST}(n,p)} \left( \frac{1}{N} \sum_{i=1}^N C_i Q_i^l \right), \quad (17)$$

where  $\text{Proj}_{\text{ST}(n,p)} : \mathbb{R}^{p \times n} \rightarrow \text{ST}(n, p)$  is the standard projection of  $X \in \mathbb{R}^{p \times n}$  to  $\text{ST}(n, p)$  (see [2] for a proof in the general case). The projection can be found, *e.g.*, via the economy SVD factorization  $X = U \Sigma V^\top$ , where we have  $\text{Proj}_{\text{ST}(n,p)}(X) = UV^\top$ . The extrinsic mean  $\bar{C}^{l+1}$  is unique as long as the Euclidean mean  $\frac{1}{N} \sum_{i=1}^N C_i Q_i^l$  is full-rank (which happens almost surely, see [2] for details and more general results). Therefore, the computed average system almost surely belongs to  $\mathcal{OM}_{m,n,p}$ , which is desirable. Compared with the Grassmann embedding approach, the above computation of  $\bar{C}^{l+1}$  can be performed very fast especially since we do not deal with matrices of size  $kp \times n$ .

---

**Algorithm 2 (LDS Align + average)**

---

1. Input:  $\{R_i = (A_i, B_i, C_i)\}_{i=1}^N \subset \mathcal{O}\mathcal{L}_{m,n,p}$ .
  2. Choose  $\bar{R}^0 = (\bar{A}^0, \bar{B}^0, \bar{C}^0) \in \mathcal{O}\mathcal{L}_{m,n,p}$ .
  3. repeat until convergence:
    - for  $i = 1$  to  $N$ , align  $R_i$  to  $\bar{R}^l = (\bar{A}^l, \bar{B}^l, \bar{C}^l)$  using Algorithm 1 and call the aligning  $Q_i, Q_i^l$ .
    - find  $\bar{R}^{l+1} = (\bar{A}^{l+1}, \bar{B}^{l+1}, \bar{C}^{l+1})$  from (15) and (17).
    - find  $\bar{d}_{i+1}^2 = \frac{1}{N} \sum_{i=1}^N d_{\mathcal{O}\mathcal{L}_{m,n,p}}^2(Q_i^l \cdot R_i, \bar{R}^{l+1})$
  4. Output:  $\bar{R} = \bar{R}^{l+1}$  and  $\bar{d}_{i+1}^2$ .
- 

The align and average algorithm is briefly described in Algorithm 2. As usual one could have several choices for the stopping criteria. Notice that in addition to the average system representation  $\bar{R}$ , the algorithm outputs the final value of the cost function  $\bar{d}_{i+1}^2$ , which is a useful quantity in many applications.

### 3.3. Clustering LDSs

Given a distance in the space of LDSs and a method for averaging LDSs, we can immediately use them for clustering a collection of LDSs  $\{M_i\}_{i=1}^N$  using a generalized  $k$ -means algorithm. Given a current estimate of  $K$  cluster centers  $\{\bar{M}_k\}_{k=1}^K$ , we can use the distance to assign each LDS to the closest cluster center. Given the clustering of the LDSs, we can compute a new cluster center by computing the average, and then iterate the two steps. This approach was used in [17] with the approximate distance described in §2.4, which effectively reduces the problem to standard  $k$ -means in the Euclidean embedding of the LDSs. The  $k$ -means algorithm was also used in [21, 7] with the Grassmann embedding approach described in §2.4. In this paper, we use the proposed Align distance Eq. (10), and the Align + average algorithm (Algorithm 2) as part of a generalized  $k$ -means algorithm. Because we use an exact averaging technique in the space of LDSs, rather than in an ambient space, we obtain better clustering results, as the experiments will show.

## 4. Experimental Results

We will first evaluate our approach on synthetic examples where we compute the mean dynamical system of an increasingly large set of samples with various levels of noise. We will also test the accuracy of our approach against some state-of-the-art approaches for computing means of LDSs. We will then provide results on some computer vision applications. For reasons explained earlier, we limit our experiments to *partial* LDS in  $\mathcal{O}\mathcal{M}_{m,n,p}$  and only consider the parameters  $(A, C)$ . We will use the name **Align** for our proposed distance,  $d_{\mathcal{O}\mathcal{M}_{m,n,p}}$  and interchangeably for the procedure for computing the mean of LDS by minimizing this distance. We use an Armijo step-size version of

Algorithm 1 in our experiments. When comparing against state-of-the-art approaches, we use **Grass mean** to denote the un-projected Grassmann mean method proposed by [21] as described in §2.4 and its corresponding projections using embeddings in  $k'$ -dimensions as **Grass  $k = k'$** . We denote the method in [17] and described in §2.4 as **MDS Average** and simply averaging the  $(A, C)$  pairs as **Naive**.

### 4.1. Experiments on synthetic data

**Data generation.** We randomly generate a matrix  $Z \in \mathbb{R}^{p \times n}$  ( $n = 5, p = 20$ ), compute the singular value decomposition (SVD),  $Z = U\Sigma V^T$ , and assign  $\bar{C} = U$ . This gives the  $C$  matrix for a mean LDS. Similarly, we generate another random matrix  $L \in \mathbb{R}^{n \times n}$  and divide by the norm of the matrix times a random value,  $0 < c < 1$ , to get  $\bar{A} = \frac{c}{\|L\|}L$ . This ensures that the system is stable and provides a ground-truth mean LDS against which we can compare several algorithms. To generate sample LDS data around this mean, we compute  $C_i = \bar{C} + \sigma E_i$ , where  $E_i \in \mathbb{R}^{p \times n}$  is a randomly generated matrix with unit Gaussian elements and  $\sigma$  is a measure of noise, *e.g.*,  $\sigma = 0.1$ . Since  $C_i^T C_i \neq I_{n \times n}$ , we compute the SVD,  $C_i = U\Sigma V^T$  and assign  $C_i = UV^T$ . Similarly, we use  $A_i = \bar{A} + \sigma L_i$ , where  $L_i \in \mathbb{R}^{n \times n}$ . To add the effects of an orthogonal basis transformation, we generate a random orthogonal matrix,  $Q_i \in O(n)$ . We then compute the sample LDS point  $(A_i, C_i)$  around the mean  $(\bar{A}, \bar{C})$  as  $A_i = Q_i^T \bar{A} Q_i$  and  $C_i = \bar{C} Q_i$ . We generate  $N = 10,000$  samples with noise levels in  $\sigma \in [0, 0.2]$ .

**Comparison with state-of-the-art methods.** To compare the performance of our method against state-of-the-art methods, we need to compare the distance between the true mean and the sample mean found using these methods using the same distance. We choose to use the distance in the Grassmann manifold for this purpose, as it will allow us to compare the mean found in the Grassmann manifold as well as its projections with other algorithms. We compare the performance of several algorithms against increasing 1) number of samples and 2) noise levels. As we can see in Fig. 1 and Fig. 2, Naive, as expected, performs the worst. MDS Average only provides one point from the samples and uses that as the mean and therefore stays away from the true mean. Grass mean, Grass ( $k = 2, \dots, 5$ ), and Align seem to become closer to the true mean as the number of samples increases. However in the zoomed in portions, we can see that not doing the projection (Grass mean) leads to larger errors. Moreover, our algorithm, Align, gives a smaller error than the projected points. Similarly, as the noise level increases, Grass mean gives larger errors.

**Computational complexity.** For a comparison of the run-time of our algorithm, Align, against Grass ( $k = 5$ ) we generate systems for output dimensions of size  $p =$

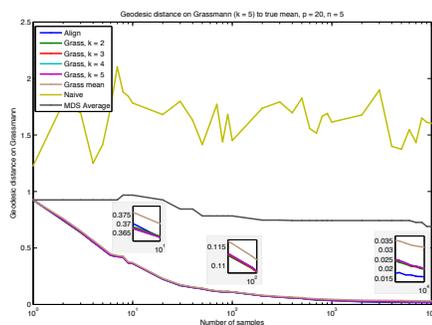


Figure 1. Grass ( $k = 5$ ) distance between true mean and sample means with increasing number of samples.

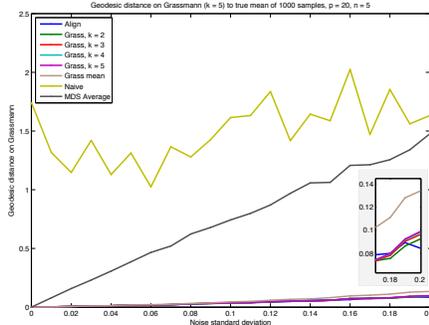


Figure 2. Grass ( $k=5$ ) distance between true mean and sample means with increasing noise level.

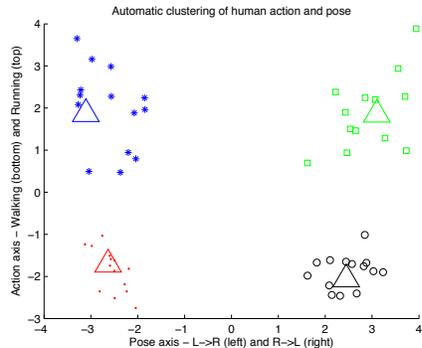


Figure 3. Clustering human actions. Color/shape represent cluster association, triangles are cluster centers.

$\{1000, 10^4\}$ . We note that Align is somewhat expensive for  $p = 1000$  (Align: 37 sec vs Grass ( $k = 5$ ): 20.8 sec) but is much faster for  $p = 10^4$  (Align: 440 sec vs Grass ( $k = 5$ ): 2475 sec). One reason for this is that as  $p$  increases, the observability matrix grows  $k$ -fold. A large amount of memory is required and the computational overhead with dealing with very large matrices slows the approach. Moreover, instead of computing an SVD of  $p \times n$  matrices, Grass  $k = k'$  requires computation of  $k'p \times n$  matrices leading to a computational complexity of  $O(k'pn^2)$  as opposed to  $O(n^2)$  for our method. However, since our method uses an iterative optimization technique for which our current MATLAB implementation is not the most efficient possible, these gains are not apparent unless  $p$  is large which is generally the case in computer vision applications.

## 4.2. Experiments on real datasets

**Human activity datasets.** We collect a set of 55 sequences of 2 human actions - walking and running. From each video, we extract a bounding box around the moving person by using background subtraction and scale them to a size of  $111 \times 67$ . We then model the optical flow in the bounding box at each time instant as the output of a LDS with  $p = 111 \times 67 \times 2 = 13,542$ . For each video, we compute the LDS parameters for order  $n = m = 5$  as described in §2. We then cluster all the systems into four clusters using the proposed LDS averaging method in a  $k$ -means clustering scheme. Once we have computed the clusters, we compute the all-pair distance,  $d_{\mathcal{O}\mathcal{M}_{n,p}}$  and perform Multi-Dimensional Scaling (MDS) to get a low-dimensional embedding of the points and the cluster centers. Fig. 3 shows the 2-D embedding obtained by MDS. As we can see, the actions are automatically divided into four clusters. Upon examination, we find that all the points are correctly clustered according to the pose of the action (moving left to right (L→R) or right to left (R→L)), as well as the action (walking or running). Moreover, not surprisingly, it appears that the space of activity LDS is composed of two principal directions: pose and action. Note that these are unsuper-

vised clustering results and we still get perfect action/pose classification.

We also apply our framework to the Weizmann human action database [13] (10 actions performed by 9 actors). We extract scaled optical-flow bounding box time-series from each video of size  $p = 63 \times 29 \times 2 = 3654$  and learn an order  $n = 5$  LDS. Table 1 shows the results when we use a simple 1-NN classification approach based on several distances. As we can see, using our proposed distance we get 100% recognition rate, which is the state-of-the-art result and better than the commonly used Martin distance and Grass ( $k = 5$ ). The superior performance of our approach supports the fact that our approach has a more natural definition for a distance between LDS. Performing 1-NN classification in general is a computationally expensive approach. As discussed in the introduction, one method for reducing this computational complexity is to use a class representative *e.g.* the mean, and use the nearest class mean for classification. We compute the means of each of the 10 classes by leaving one test sequence out and then classifying it by assigning it the label of the nearest class mean. Table 2 shows the classification results and the average class mean computation time. As we can see, our algorithm provides better classification results and requires much smaller computation time. As the size of the dataset is small, computing the approximate mean based on the Martin distance requires much less time, however the corresponding results are very poor.

**Dynamic texture datasets.** In [17], the authors proposed a Bag-of-Systems (BoS) approach to dynamic texture recognition, which generalizes the Bag-of-Features approach from Euclidean to LDS features. One step of the BoS algorithm is to cluster a collection of LDSs to obtain a dictionary of codewords, which is computed using the approximate averaging algorithm described in §2.4. Here, we use our LDS averaging framework to construct the codewords and compare with the original BoS approach. We use the UCLA8 dynamic texture database from [17] which

1-NN classification	Rec (%)
Martin	96.77
Grass ( $k = 5$ )	95.70
Align	<b>100</b>

Table 1. Action classification results using 1-nearest neighbor on Weizmann database.

Method	Rec (%)	Time (s)
Martin	44.09	0.04
Grass ( $k = 5$ )	92.47	267
Align	<b>93.55</b>	<b>16</b>

Table 2. Action classification results using nearest class mean on Weizmann database.

Method	Recognition (%)	
MDS average [17]	68.18	72.73
Align + average	<b>75.00</b>	<b>79.55</b>
Patch size ( $\sigma$ )	20	30

Table 3. Dynamic texture classification results using the BoS method on UCLA8 database.

is a view-invariant subset of the database in [11]. We use the Term-Frequency (TF) approach to compute training and testing histograms and the  $\chi^2$ -distance between histograms to perform 1-NN classification. Table 3 shows the classification result when using 56 clusters for computing code-words on LDS ( $n = 5$ ) obtained by densely sampling the original videos in  $\sigma \times \sigma \times 25$  patches for  $\sigma = \{20, 30\}$ . Here  $p = 20 \times 20 = 400$ , and  $30 \times 30 = 900$  respectively. As we can see, our method provides better classification results than the original approximate average approach. This can be attributed to the use of the exact mean of the clusters. Moreover, since our algorithm does not require computation of an all-pair distance matrix for computing the code-words, it is also much more efficient. We could not perform clustering using Grass due to large number of systems ( $\approx 7000$ ) of very high dimension ( $900 \times 5$ ).

## 5. Conclusion

We have presented a novel distance for comparing LDSs, which can be used for several applications in computer vision. The proposed distance is defined directly on the space of LDSs, rather than on an embedding. As a consequence, one can compute averages directly on the space of LDSs, rather than approximately or via a projection. Our experiments show how the exact and rigorous computation of distances and averages improves classification and clustering results in computer vision applications.

## 6. Acknowledgements

This work has been supported in part by the grants ONR N00014-09-10084, NSF CAREER 0447739, NSF 0941362, NSF 0941463 and NSF 0931805.

## References

- [1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton, NJ, 2008.
- [2] R. Bhattacharya and V. Patrangenaru. Large sample theory of intrinsic and extrinsic sample means on manifolds. I. *Ann. Statist.*, 2003.
- [3] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *CVPR*, 2001.
- [4] A. Chan and N. Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual processes. In *CVPR*, 2005.
- [5] A. Chan and N. Vasconcelos. Classifying video with kernel dynamic textures. In *CVPR*, 2007.
- [6] A. Chan and N. Vasconcelos. Layered dynamic textures. *TPAMI*, 2009.
- [7] R. Chaudhry and Y. Ivanov. Fast approximate nearest neighbor methods for non-euclidean manifolds with applications to human activity analysis in videos. In *ECCV*, 2010.
- [8] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *CVPR*, 2009.
- [9] R. Chaudhry and R. Vidal. Recognition of visual dynamical processes: Theory, kernels and experimental evaluation. TR 09-01, Dept. CS, JHU, 2009.
- [10] K. D. Cock and B. D. Moor. Subspace angles and distances between ARMA models. *System and Control Letters*, 2002.
- [11] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. Dynamic textures. *IJCV*, 2003.
- [12] A. Edelman, T. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM J Mat. Anal. App.*, 1998.
- [13] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *TPAMI*, 2007.
- [14] B. Hanzon. *Identifiability, Recursive Identification and Spaces of Linear Dynamical Systems*, volume CWI Tracts 63 and 64. Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1989.
- [15] J. M. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer, 2002.
- [16] A. Martin. A metric for ARMA processes. *TSP*, 2000.
- [17] A. Ravichandran, R. Chaudhry, and R. Vidal. View-invariant dynamic texture recognition using a bag of dynamical systems. In *CVPR*, 2009.
- [18] A. Ravichandran, P. Favaro, and R. Vidal. A unified approach to segmentation and categorization of dynamic textures. In *ACCV*, 2010.
- [19] A. Ravichandran and R. Vidal. Video registration using dynamic textures. In *ECCV*, 2008.
- [20] P. Saisan, A. Bissacco, A. Chiuso, and S. Soatto. Modeling and synthesis of facial motion driven by speech. In *ECCV*, 2004.
- [21] P. Turaga, A. Veeraraghavan, and R. Chellappa. Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision. In *CVPR*, 2008.
- [22] S. Vishwanathan, A. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *IJCV*, 2007.
- [23] L. Younes. *Shapes and Diffeomorphisms*, vol. 171 *App. Mathematical Sciences*. Springer, 2010.