A Dynamical Systems Approach to Classification of Surgical Gestures in Kinematic and Video Data

by

Benjamín Béjar Haro

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Master of Science.

Baltimore, Maryland

October, 2013

Abstract

In Computer Assisted Intervention (CAI) systems, a surgeon performs the surgery using an interface connected to a computer that remotely controls a set of surgical tools attached to a robot. Such systems are particularly appealing for minimally invasive surgeries since they allow for a larger and more precise set of movements than in traditional laparoscopic interventions, and provide enhanced vision capabilities such as 3D vision and augmented reality. These features directly translate into benefits for the patients such as smaller incisions, less pain and quicker healing. However, the benefits of the technology might be reduced due to the steep learning curve associated with CAI systems. This makes it necessary to account for a fair and objective criterion for the evaluation and assessment of the skills of a novice surgeon. Furthermore, it is desirable to automate the process in order to avoid constant supervision of an expert surgeon, a time consuming, subjective and rather inefficient method.

It is therefore necessary to develop algorithmic methods that extract information from kinematic cues provided by the robot and video recordings of the interventions. A common approach is to divide the surgical procedure into smaller actions,

ABSTRACT

forming a vocabulary able to to describe different surgical tasks. Following such an approach requires a method capable of providing temporal segmentation, recognition of the action and final skill assessment. Prior work has usually modeled the interactions between these atomic actions using generative models such as Hidden Markov Models, Factor-Analysis and Switching Linear Dynamical Systems. In this thesis, we focus on the classification problem and assume segmented data. We propose to follow a discriminative approach using Linear Dynamical Systems (LDS) to model and characterize a particular action. We develop new methods for the extraction of meaningful representations by means of averaging in the space of LDSs. These representative points are then used into a discriminative framework for surgical gesture classification. We propose a novel SVM classification method for time series of data that reduces computation at the expense of some degradation in performance. Our contributions are fairly general and can be applied to any temporal signal coming from an LDS.

Primary Reader: Prof. Gregory D. Hager

Secondary Reader: Prof. Sanjeev Khudanpur

Acknowledgments

There are many people I should thank for their contribution, in one way or another, to the present work. First of all, I would like to thank my advisor Dr. René Vidal for giving me the opportunity to join his lab. These two years have been a great experience both in professional and personal aspects. I try to learn as much as I can from the people I meet, and I have certainly done so from Dr. Vidal. I sincerely appreciate his guidance, advice and criticism during these years, and I am really grateful for his teachings. I would like to thank my lab mates Lingling and Luca, for their support and help during the project. Also, I would like to thank all the people (faculty and students) involved into the "Language of Surgery" project. Thanks to Dr. Greg Hager, Dr. Sanjeev Khudanpur and Dr. René Vidal for leading the project. Thanks to Anand, Narges, Yixin, Luca, Lingling, Swaroop, and Piyush for their contribution to the project and for their useful comments and discussions. Thanks to the rest of my lab mates: Bijan, Erdem, Rizwan, Roberto, Ehsan, Siddharth, Chong, Giann, Evan, and Manolis for being always friendly and nice to me, and for helping me when I needed it. I enjoyed a lot the social events (cakes, farewell dinners, parties,

etc) that we attended during this time. Thanks to Siddharth and Chong for their help with the moving out, and thanks to Bertrand for replacing Luca during coffee time. I wish all of them best of luck in their career and future projects. Thanks to Paco, Marta, René, Camille, Luca and Becky for being great friends. Special thanks to Luca, who made my journey much easier. I discovered in him a great researcher, a wonderful person, and a best friend. Thanks to the Talentia Fellowships program of the Andalusian government for giving me this excellent opportunity. And last, but not least, none of this would have been possible without the support and love of my family, specially of my wife Mari, who always gives me all her love and support.

Dedication

To Mari, who makes life beautiful.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations

Over 100 years ago, Dr. William Halsted created the first surgical residency training program in the United States. His training paradigm was extremely simple: "see one, do one, teach one." However, recent technological advances have changed the way in which some surgeries are performed. This has opened the opportunity for revisiting Halsted's paradigm in search for improved ways of training surgeons.

One of such technological advances is Robotically Minimally Invasive Surgery (RMIS), which has several advantages over traditional surgery, as shown in [2,3] and [4]. For example, [4] compared the post-surgery recovery of patients who underwent RMIS to that of patients who underwent traditional surgery. One of the findings was that the former group experienced a shorter length of stay, and was less likely

to receive blood transfusions or develop postoperative respiratory and miscellaneous surgical complications.

However, after a first wave of optimism about RMIS, drawbacks started to arise. In the same study, [4] observed that RMIS was associated with an almost 2-fold increase in the odds of postoperative genitourinary complications. One of the hypotheses for this increment is the steep learning curve for surgeons who want to add RMIS to their armamentarium. In fact, even for expert surgeons, training for RMIS is often considered challenging, as reported in [5]. This is exacerbated by the fact that there is a lack of fair, objective, and effective criteria for judging the skills acquired by a trainee with an RMIS system, which could ultimately reduce the benefits of such technology.

These issues have motivated a number of approaches for automatic RMIS skill assessment and gesture classification. One of the most natural approaches is to decompose a surgical task into a series of pre-defined "atomic" gestures or *surgemes*, such as "insert needle", "grab needle", "position needle", etc. Figure 1.1 shows sample frames from three different surgemes taken from the dataset presented in [1]. The problem then becomes how to segment the task in time, recognize each surgeme, and finally assess the skill level.

Even if RMIS systems are typically equipped with cameras that record the entire procedure, to the best of our knowledge, most of the studies focused mainly on the analysis of kinematic data stored by the robot. This kind of information typi-

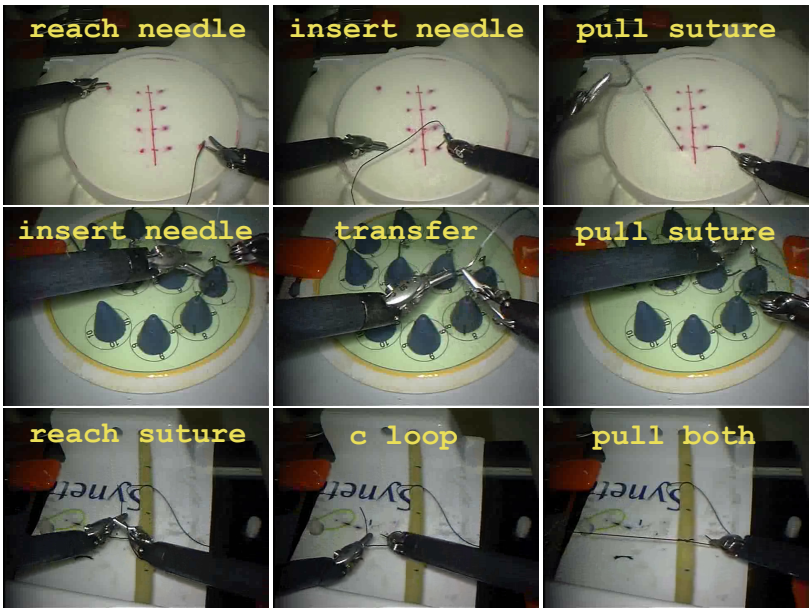

Figure 1.1: **Surgeme Examples** – A representation of some of the *surgemes* present in the set of actions that define a task.

cally involves the position of the robot tools, angles between robot joints, velocity measurements and force/torque signatures. In the medical literature, action recognition techniques from video have been applied to the analysis of the behavior of surgeons and nurses in an operating room [6–8]. However, as far as video recognition of surgical gestures, little has been done mainly because kinematic-based approaches usually outperformed those based on visual cues. For example in [9] some basic gesture recognition strategies from video were investigated, but the conclusion of this study was that kinematic-based approaches were generally more accurate. Recent approaches based on video [10–12] show that video can provide very high performances in automatic recognition of the different surgical phases, suggesting that the above

conclusion should be revisited.

Additionaly, video data can be a rich source of contextual information about a surgical task, which can be complementary to the motion information contained in kinematic data. For example, in a surgical video we can see which tool is being used, whether the tool is in contact with the tissue, whether the suture is passing through the suture line, etc. All of that information is not available in the kinematic data. Moreover, recognition methods based on video data could be more generally applicable to any MIS where video data is available but kinematic data is not. On the other hand, the automatic detection and tracking of surgical tools, and the detection of surgical events in video data is very challenging due to the occlusions and clutter present in a surgical video, as well as the variability of tool pose and motions across tasks and surgeon expertise.

Motivated by these ideas, the work in [11, 13] proposed the first steps towards automatic recognition of surgical gestures in video. Rather that aiming to a complete semantic interpretation of a surgical video, the authors proposed to use the statistical properties of features extracted from the video to build models for each gesture and use these models to classify surgical gestures in new videos. More specifically, given a video of a surgical task (*e.g.*, suturing), it is assumed that the video has been segmented into video clips corresponding to a single gesture (*e.g.*, position needle, drive needle through tissue, pull suture, etc.). The problem is then to recognize the gesture associated with each (segmented) video clip. The authors in [11,13] use Linear

Dynamical Systems (LDSs) to model the surgical gestures. Using different features, such as raw pixel intensities or optical flow, and different metrics in the space of LDSs the authors achieve state-of-the-art performance in surgical gesture classification.

1.2 Fundamental challenges

The use of system-theoretic techniques (*e.g.*, LDSs) to the modeling of high-dimensional time-series data has also found application in many other computer vision problems over the past decade [14–20]. However, if we want to characterize some process using LDSs, there are several challenges that need to be addressed. The first one is to estimate or learn the parameters of the model (identification problem). This is a well-studied subject for LDSs [16, 21, 22]. Once the model has been estimated, one may be interested in performing some statistical analysis on these models. This entails problems such as determining which process is being observed in a given time series (classification problem), finding a representative for a class of processes (averaging problem), or clustering time series according to different classes (clustering problem). Classification, recognition and clustering problems have been mainstream areas of research in machine learning for the past decades. However, most existing methods for the analysis of high-dimensional time series require a distance or kernel on the (typically Euclidean) observation space [23]. A fundamental challenge in performing pattern recognition in the space of LDSs (*i.e.*, find averages, learn clusters,

do kernel-based classification) is that the space is not Euclidean. Indeed, it has the structure of a quotient space. The problem of averaging and clustering LDSs have been previously addressed in several works: [24,25] find an embedding of the LDS into a Grassmann manifold and perform the averaging in the embedding; [26] proposes to embed the LDS points into Euclidean space using the pairwise distances and a non-linear dimensionality reduction technique such as multidimensional scaling. One limitation of such methods is that they are approximate and not able to generate novel LDSs. On the contrary, more recent approaches try to find novel averages directly in the space of LDSs without relying to any embedding of the data. The work in [27] proposes to find an average LDS using the hierarchical Expectation Maximization algorithm. The main drawback of such method is its computational complexity, since it requires running a Kalman filter on every averaging step. Also in a recent contribution [28], the authors propose a distance and an averaging method based on the equivalence of representations between LDSs.

Another key challenge shows up in the statistical analysis of large collections of dynamical models identified from high-dimensional datasets, where the computational complexity might become an issue. Therefore, it is desirable to have efficient methods to perform these statistical tasks (*e.g.*, averaging, classification).

1.3 Thesis contributions

Our main goal is to develop methods for automatic gesture recognition in surgical tasks. This clearly constitutes a first step towards having an automated system for the assessment and evaluation of novice surgeons. We take a similar approach as in [1] and decompose each task into atomic gestures called *surgemes*. Using low-level features (*e.g.*, pixel intensities and kinematic recordings), we fit an LDS to the data in order to capture both the dynamics and the appearance associated with each of the *surgemes*. Once we have extracted the models for each of the actions (we follow a similar approach as in [11, 13] and assume that the different gestures have already been segmented), we can use metrics in the space of LDS to build classifiers (*e.g.*, k-Nearest-Neighbors or Support Vector Machines) to recognize new actions. In general, we are interested in performing some statistical analysis in the space of LDSs. To that end, we will investigate averaging (extrinsic mean) and clustering of LDSs with respect to some distance metrics. Our first contribution is to propose a novel averaging method with respect to the Martin distance [29, 30]. The Martin distance is a particularly attractive metric since it is a true distance in the space of LDSs that is also invariant to basis transformations, a desired property when comparing dynamical models. To the best of our knowledge, the only previous attempt to produce an average LDS model using the Martin distance is the nonlinear dimensionality reduction method of [26]. Contrary to [26], our method is not restricted to select a point in the sample set but it is able to produce novel LDS averages. We pose the

problem as a constrained optimization problem and devise a gradient-descent type of algorithm for the computation of the minimizer. To that end we derive expressions for the efficient computation of the derivatives of the objective function with respect to the optimization variables. Our second contribution to averaging and clustering LDSs is the derivation of an alternative method for the computation of the Align distance [28]. We propose to compute the distance using the Alternating Direction Method of Multipliers [31]. The advantage of using our method is that it can be easily implemented since the updates can be computed in closed-form. The Align distance can also be used for averaging as described in [28].

These averaging methods will allow us to extract some representative points for a given class of processes. We will then use those representatives to perform classification of LDSs. Using linear predictors in the ambient space of time series, we will study the classification problem using support vector machines. The classical approach is to use a kernel in the space of LDSs. Such kernel can be built from a distance measure (*e.g.*, using a radial basis function) such as the Martin or Frobenius distances [29, 30, 32], or directly using a kernel in the space of LDSs such as the Binet-Cauchy kernels proposed in [33]. It is well-known that the optimal separating hyperplane is a linear combination of the data points (in the reproducing Hilbert space) [23]. In this thesis, and this is our third contribution, we also consider an alternative and novel classification method that restricts the optimal separating hyperplane to a specific set of representative points. By fixing some of the parameters

of the separating hyperplane and optimizing only over the initial conditions of the models, we will show how the kernel SVM classification problem reduces to a linear SVM in Euclidean space. The method also benefits from a reduced complexity as compared to the Binet-Cauchy kernels. We call this new method DynamicSVM. Finally, our fourth contribution is the application of these techniques to the particular problem of surgical action classification. We will use the dataset presented in [1] for the evaluation of the described procedures. It is worth to mention that our methods are current state-of-the-art in surgical action recognition [11, 13].

1.4 Thesis outline

The remainder of the thesis is organized as follows. In Chapter 2 we present some background material that will be used througout the thesis. Chapter 3 addresses the problem of averaging and clustering LDSs using the Martin and the Align distances. We will show the derivation of the methods and will illustrate their performance on different real datasets. In Chapter 4 we will present the DynamicSVM classification method for time series coming from LDSs. We will illustrate its relation to the Binet-Cauchy kernels of [33] and how the kernel SVM problem can be reduced to a linear SVM in Euclidean space. Finally, in Chapter 5 we will evaluate the presented techniques for the problem of surgical gesture classification.

Chapter 2

Linear Dynamical Systems

2.1 Introduction

A common approach to describe and represent data time series is to use a stochastic generative approach such as a Linear Dynamical System (LDS), where the dynamics of the observed sequence are assumed to be governed by the time evolution of some latent (unobserved) variable. Such models have been successfully applied to several problems in computer vision such as synthesis and classification of dynamic textures, action recognition or segmentation, among others. For instance, [11] uses Linear Dynamical Systems (LDSs) to model surgical gestures in kinematic and video data from the DaVinci robot; [34, 35] use LDSs to model the appearance of a deforming heart in a magnetic resonance image sequence; [16, 18, 20, 26, 36–39] use LDSs to model the appearance of dynamic textures, such as water or fire, in a video sequence; [14, 15] use

LDSs to model human gaits, such as walking or running, in motion capture and video data; [19] uses LDSs to model the appearance of moving faces; and [40] uses LDSs to model audio-visual lip articulations. Given a high-dimensional time series, one can use standard system identification techniques, *e.g.*, subspace identification [22], to learn the parameters of an LDS model. Given a model, novel time series can be synthesized by simulating the model forward. For example, impressive synthesis of dynamic textures has been demonstrated by a number of papers [16–18, 41]. The same ideas have also been used for the synthesis of lip articulations using speech as the driving input [40].

When it comes to classification/recognition problems, most of the approaches based on LDSs rely on some dissimilarity metric or distance in the space of LDSs. There exist several metrics that have been commonly used in the literature such as distances based on the Binet-Cauchy kernels [42], probabilistic metrics based on the KL-divergence [43] or metrics such as the Martin distance [29, 30]. The Martin distance was originally proposed in [30] for Single-Input Single-Output (SISO) Auto-Regressive Moving Average (ARMA) processes, and it was computed as a function of the cepstrum coefficients (inverse Fourier transform of the logarithm of the power spectrum). It was later generalized to Multiple-Input Multiple-Output (MIMO) systems as a function of the subspace angles between the observability subspaces of the dynamical models [29]. A more recent approach [28], defines a distance between LDSs based on the equivalence of representations between models. The idea is to find the

"closest" representation between two models through an orthogonal basis transformation. Once a metric has been selected, a simple approach to classify novel sequences based on that metric is to use a k-nearest neighbors classifier or a kernel support vector machine (SVM) [23] using, *e.g.*, a radial basis function kernel.

This chapter aims to provide the reader with some background material about LDSs that will be used throughout the thesis. We will start with a formal description of LDSs and then we will cover topics such as parameter estimation, metrics in the space of LDSs and their application to classification problems.

2.2 Formal description

In order to model the statistical properties of a temporal sequence of observations, it is often assumed that the observed sequence is correlated with a hidden (continuous) state variable that is evolving over time. In the case of an LDS, the state variable is assumed to be real-valued and the underlying driving and measurement noise processes are assumed to be Gaussian. More formally, we have that

$$\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{u}_t, \tag{2.1}$$

$$\mathbf{z}_t = \mathbf{C}\mathbf{s}_t + \mathbf{w}_t, \tag{2.2}$$

where $\mathbf{s}_t \in \mathbb{R}^n$ represents the (unobserved) state variable at time instant t, $\mathbf{z}_t \in \mathbb{R}^p$ is the observed signal (*e.g.*, a video frame) and $\mathbf{B} \in \mathbb{R}^{n \times z}$ is a noise-coloring matrix. Without loss of generality we will assume the sequences to start at time

instant $t = 0$. The noise sequences are assumed to be independent and identically distributed (i.i.d.) Gaussian random variables that are also jointly independent. More precisely, $\{\mathbf{u}_t\}_{t=0}^{\infty} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\{\mathbf{w}_t\}_{t=0}^{\infty} \sim \mathcal{N}(\mathbf{0}, \mathbf{W})$, where $\mathbf{I}$ is the identity matrix of appropriate dimensions and $\mathbf{W}$ is the measurement noise covariance matrix. It is important to mention that in the case of dynamic scenes (video data) the dimension p of the observed signal is usually much larger than the order n of the system ($p \gg n$). Given the state-space representation of (2.1) and (2.2), a linear dynamical model is parametrized by the tuple $\mathcal{M} = (\mathbf{A}, \mathbf{C}, \mathbf{B}, \mathbf{W}, \mathbf{s}_0)$, with $\mathbf{s}_0$ being the initial state (initial conditions).

Notice however, that this representation is not unique. This is because an equivalent representation can be found by a change of coordinates of the state variable. More specifically, if we define $\mathbf{s}_t = \mathbf{T}\,\mathbf{r}_t$, where $\mathbf{T} \in \mathbb{R}^{n \times n}$ is non-singular, then the two representations $\mathcal{M} = (\mathbf{A}, \mathbf{C}, \mathbf{B}, \mathbf{W}, \mathbf{s}_0)$ and $\tilde{\mathcal{M}} = (\mathbf{T}^{-1}\mathbf{A}\mathbf{T}, \mathbf{C}\mathbf{T}, \mathbf{T}^{-1}\mathbf{B}, \mathbf{W}, \mathbf{s}_0)$ are equivalent (*i.e.*, both represent the same process $\mathbf{z}_t$). This consideration will be important when comparing two dynamical models by looking at their parametric representation since this representation is not unique.

In what follows we will restrict ourselves to the case of stable and observable Auto-Regressive (AR) processes. Recall that an LDS is stable if the magnitude of the eigenvalues of $\mathbf{A}$ is smaller than one, and that it is observable if the observability matrix is full rank (see Section 2.3.1.2).

2.3 Metrics and kernels in the space of LDSs

In this section we will provide an overview of some of the metrics commonly used to compare dynamical models. In particular, we will focus our attention on metrics based on the subspace angles between the observability subspaces of the dynamical models such as the Martin and Frobenius distances [29, 30, 32], and the metric proposed in [28], that proposes a similarity measure based on the equivalence of representations between dynamical models. We will also review some kernels for dynamical models such as the Binet-Cauchy kernels proposed in [33].

2.3.1 Distances based in the subspace angles

For the sake of completeness, let us first provide a quick review about the notion of angles between subspaces. Let us start with the simple definition of an angle between two vectors $\mathbf{u}$ and $\mathbf{v}$ in some vector space. It is well-known that

$$\cos\theta = \frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|\|\mathbf{v}\|}, \tag{2.3}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product (*e.g.*, $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^\mathsf{T}\mathbf{v}$ in $\mathbb{R}^n$), $\|\cdot\|$ is the induced norm by the inner product (*i.e.*, $\|\mathbf{u}\|^2 = \langle \mathbf{u}, \mathbf{u} \rangle$), and θ is the angle between the two vectors. If the two vectors are unitary (unit norm) then, their inner product corresponds to the cosine of the angle between them.

2.3.1.1 Principal angles between subspaces

Consider now the case of two subspaces generated by the column space of two real matrices $\mathbf{M}_1 \in \mathbb{R}^{n\times r}$ and $\mathbf{M}_2 \in \mathbb{R}^{n\times q}$, both full column rank and with $r \geq q$. Then the cosine of the first *principal angle* θ_1 between $\mathcal{U} = \text{range}\,(\mathbf{M}_1)$ and $\mathcal{V} = \text{range}\,(\mathbf{M}_2)$ is defined as

$$\cos\theta_1 = \max_{\mathbf{u},\mathbf{v}} \left\{ \frac{\langle \mathbf{u}, \mathbf{v}\rangle}{\|\mathbf{u}\|\|\mathbf{v}\|} \;\middle|\; \mathbf{u} \in \mathcal{U}, \mathbf{v} \in \mathcal{V} \right\}. \tag{2.4}$$

If we denote by $\mathbf{u}_i$ and $\mathbf{v}_i$ the *principal vectors* associated with the principal angle θ_i, $i = 1, \ldots, q$, we can recursively compute the principal angles as

$$\cos\theta_i = \max_{\mathbf{u},\mathbf{v}} \left\{ \frac{\langle \mathbf{u}, \mathbf{v}\rangle}{\|\mathbf{u}\|\|\mathbf{v}\|} \;\middle|\; \mathbf{u} \in \mathcal{U}, \mathbf{v} \in \mathcal{V}, \text{and } \mathbf{u} \perp \mathbf{u}_j, \mathbf{v} \perp \mathbf{v}_j, j = 1, \ldots, i-1 \right\}, \tag{2.5}$$

where $\mathbf{u} \perp \mathbf{v}$ means that the two vectors are orthogonal (*i.e.*, $\langle \mathbf{u}, \mathbf{v}\rangle = 0$).

Alternatively, one can show that the principal angles can be obtained from the following generalized eigenvalue problem

$$\begin{bmatrix} \mathbf{0} & \mathbf{M}_1^\mathsf{T}\mathbf{M}_2 \\ \mathbf{M}_2^\mathsf{T}\mathbf{M}_1 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{M}_1^\mathsf{T}\mathbf{M}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_2^\mathsf{T}\mathbf{M}_2 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \tag{2.6}$$

where $\mathbf{u}^\mathsf{T}\mathbf{M}_1^\mathsf{T}\mathbf{M}_1\mathbf{u} = 1$ and $\mathbf{v}^\mathsf{T}\mathbf{M}_2^\mathsf{T}\mathbf{M}_2\mathbf{v} = 1$. The relationship to the principal angles is then given by

$$\cos(\theta_i) = \lambda_i,\ i = 1, \ldots, q, \tag{2.7}$$

with λ_i being the generalized eigenvalues of Equation (2.6).

2.3.1.2 Principal angles between LDS

Since a stable and observable LDS can be equivalently represented by its observability matrix, De Cock and De Moor [29], define the principal angles between two stable and observable AR linear dynamical systems as the principal angles between their observability subspaces.

Consider two stable and observable LDS models, $\mathcal{M}_1 = (\mathbf{A}_1, \mathbf{C}_1)$ and $\mathcal{M}_2 = (\mathbf{A}_2, \mathbf{C}_2)$. Their respective observability matrices are the infinite-dimensional matrices ($\mathbb{R}^{\infty \times n}$) given by

$$\mathbf{O}_1^\mathsf{T} = \begin{bmatrix} \mathbf{C}_1^\mathsf{T} & (\mathbf{C}_1\mathbf{A}_1)^\mathsf{T} & (\mathbf{C}_1\mathbf{A}_1^2)^\mathsf{T} & \dots \end{bmatrix} \tag{2.8}$$

$$\mathbf{O}_2^\mathsf{T} = \begin{bmatrix} \mathbf{C}_2^\mathsf{T} & (\mathbf{C}_2\mathbf{A}_2)^\mathsf{T} & (\mathbf{C}_2\mathbf{A}_2^2)^\mathsf{T} & \dots \end{bmatrix} \tag{2.9}$$

Therefore, (for stable models) the principal angles can be computed from the generalized eigenvalues of

$$\begin{bmatrix} \mathbf{0} & \mathbf{O}_1^\mathsf{T}\mathbf{O}_2 \\ \mathbf{O}_2^\mathsf{T}\mathbf{O}_1 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{O}_1^\mathsf{T}\mathbf{O}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{O}_2^\mathsf{T}\mathbf{O}_2 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \tag{2.10}$$

subject to the constraints $\|\mathbf{O}_1\mathbf{u}\| = 1$ and $\|\mathbf{O}_2\mathbf{v}\| = 1$.

2.3.1.3 The Martin and Frobenius distances

Consider again the stable and observable LDS models $\mathcal{M}_1$ and $\mathcal{M}_2$ of order n and let $\mathbf{Q}_{ij} = \mathbf{O}_i^\mathsf{T}\mathbf{O}_j$, $i, j = 1, 2$ denote the observability Grammians. The Martin

distance is then related to the subspace angles [29] by

$$d_M^2(\mathcal{M}_1, \mathcal{M}_2) = -\log \prod_{i=1}^{n} \cos^2(\theta_i), \tag{2.11}$$

where the following relation holds:

$$\cos^2(\theta_i) = \lambda_i \left(\mathbf{Q}_{11}^{-1} \mathbf{Q}_{12} \mathbf{Q}_{22}^{-1} \mathbf{Q}_{21} \right), \tag{2.12}$$

with $\lambda_i(\mathbf{X})$ being the ith largest eigenvalue of matrix $\mathbf{X}$. Therefore, we can express the Martin distance as

$$d_M^2(\mathcal{M}_1, \mathcal{M}_2) = -\log \det \mathbf{Q}_{11}^{-1} \mathbf{Q}_{12} \mathbf{Q}_{22}^{-1} \mathbf{Q}_{21}, \tag{2.13}$$

where $\mathbf{Q}_{ij}$ are obtained as the solution to the following Sylvester equation:

$$\mathbf{Q}_{ij} = \mathbf{A}_i^\mathsf{T} \mathbf{Q}_{ij} \mathbf{A}_j + \mathbf{C}_i^\mathsf{T} \mathbf{C}_j, \ i, j = 1, 2. \tag{2.14}$$

Another distance based on subspace angles that we will consider is the Frobenius distance and it is defined as

$$d_F^2(\mathcal{M}_1, \mathcal{M}_2) = 2 \sum_{i=1}^{n} \sin^2(\theta_i). \tag{2.15}$$

2.3.2 Action-induced distances

Afsari *et al.*, [28] proposed an alternative approach to comparing LDSs based on finding a basis transformation that bring their realizations as close as possible to each other. Recall that the two LDS realizations $\mathcal{M} = (\mathbf{A}, \mathbf{C}, \mathbf{B}, \mathbf{W}, \mathbf{s}_0)$ and $\tilde{\mathcal{M}} = (\mathbf{T}^{-1}\mathbf{A}\mathbf{T}, \mathbf{C}\mathbf{T}, \mathbf{T}^{-1}\mathbf{B}, \mathbf{W}, \mathbf{s}_0)$ are equivalent provided that $\mathbf{T}$ is non-singular.

In order to derive a metric, the authors in [28] further assume that the LDSs are asymptotically stable (*i.e.*, $\max_i |\lambda_i(\mathbf{A})| < 1$) and observable (*i.e.*, the observability matrix has full rank). With these considerations in mind, one could define a distance between two (stable and observable) LDS realizations (of order n) $\mathcal{M}_1$ and $\mathcal{M}_2$ as

$$\begin{aligned} d^2(\mathcal{M}_1, \mathcal{M}_2) = \inf_{\mathbf{T}_1, \mathbf{T}_2 \in GL(n)} \Big\{ & \lambda_A \|\mathbf{T}_1^{-1}\mathbf{A}_1\mathbf{T}_1 - \mathbf{T}_2^{-1}\mathbf{A}_2\mathbf{T}_2\|_F^2 \\ & + \lambda_C \|\mathbf{C}_1\mathbf{T}_1 - \mathbf{C}_2\mathbf{T}_2\|_F^2 + \lambda_B \|\mathbf{T}_1^{-1}\mathbf{B}_1 - \mathbf{T}_2^{-1}\mathbf{B}_2\|_F^2 \Big\}, \end{aligned} \tag{2.16}$$

where $\lambda_A \geq 0$, $\lambda_C \geq 0$ and $\lambda_B \geq 0$ are positive weights, $\|\cdot\|_F$ denotes the Frobenius norm of a matrix, and where $GL(n)$ denotes the set of non-singular matrices of order n.

When $p \geq n$, it is common to assume that the matrix $\mathbf{C}$ is full rank. For such cases, we can always find an equivalent representation of an LDS such that $\mathbf{C}$ is orthonormal, that is $\mathbf{C}^\mathsf{T}\mathbf{C} = \mathbf{I}$. Such an equivalent representation can be easily obtained by *e.g.*, the SVD of $\mathbf{C}$. Therefore, when comparing two LDS representations, the authors in [28] restrict themselves to realizations such that $\mathbf{C}$ is orthonormal since, for every stable and observable LDS, such a representation always exist. One important implication of using an orthonormal representation of $\mathbf{C}$ is that, whenever two LDSs are equivalent (*i.e.*, they can be related by a basis transformation), then the basis transformation that relates them must belong to the orthogonal group $O(n) = \left\{\mathbf{T} \in \mathbb{R}^{n\times n} \,|\, \mathbf{T}^\mathsf{T} = \mathbf{T}^{-1}\right\}$. This allows a more tractable formulation of the problem and it also facilitates the computation of the metric using gradient-descent type of algorithms due to the compactness of $O(n)$. The (squared) Align distance between

two models $\mathcal{M}_i$ and $\mathcal{M}_j$ is then defined as

$$d_A^2(\mathcal{M}_1, \mathcal{M}_2) = \min_{\mathbf{Q} \in O(n)} \Big\{ \lambda_A \|\mathbf{Q}^\mathsf{T} \mathbf{A}_1 \mathbf{Q} - \mathbf{A}_2\|_F^2 + \lambda_C \|\mathbf{C}_1 \mathbf{Q} - \mathbf{C}_2\|_F^2 + \lambda_B \|\mathbf{Q}^\mathsf{T} \mathbf{B}_1 - \mathbf{B}_2\|_F^2 \Big\}, \tag{2.17}$$

where, different from (2.16), we can lump the effect of both $\mathbf{T}_1$ and $\mathbf{T}_2$ into a single matrix $\mathbf{Q}$ due to the invariance of the Frobenius norm with respect to orthogonal transformations. Note from (2.17) that the computation of the Align distance requires the solution of an optimization problem over the manifold of orthogonal matrices. For that purpose, the authors in [28] propose a Riemannian gradient-descent algorithm that runs two separate optimizations over the two disjoint connected components of $O(n)$.

2.3.3 Kernels between LDSs

An alternative way to compute a similarity measure between two data points $\mathbf{x}$ and $\mathbf{x}'$ is to use their inner product. For example, if $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, where $\mathcal{X}$ is a subset of $\mathbb{R}^n$, then $\langle \mathbf{x}, \mathbf{x}' \rangle = \mathbf{x}^\mathsf{T} \mathbf{x}' = \sum_{i=1}^n x_i x_i'$ measures the angle between the two vectors, provided that both $\mathbf{x}$ and $\mathbf{x}'$ are of unit norm (*e.g.*, $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = 1$). However, it is not always the case that the data points have a vectorial representation in a dot product space (*e.g.*, if we want to compare strings). In such cases, we may resort to a map $\phi : \mathcal{X} \mapsto \mathcal{H}$ into some Hilbert space $\mathcal{H}$ where we can still compute an inner product. Therefore, we can define a similarity measure between to points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

using a map of the form:

$$\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle_{\mathcal{H}}, \tag{2.18}$$

where $\kappa : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is referred to as a *kernel*. Note that the map $\kappa(\cdot, \cdot)$ in (2.18) is *symmetric* (*i.e.*, $\kappa(\mathbf{x}, \mathbf{x}') = \kappa(\mathbf{x}', \mathbf{x})$) and *positive definite* since

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \tag{2.19}$$

for all $\mathbf{x}_i \in \mathcal{X}$, $c_i \in \mathbb{R}$, $i = 1, \ldots, m \in \mathbb{N}$.

We have used the map $\phi(\cdot)$ to define the kernel $\kappa(\cdot, \cdot)$ as an inner product in some Hilbert space $\mathcal{H}$. What is interesting is that the same relationship can be stablished the other way around *i.e.*, given a kernel map $\kappa(\cdot, \cdot)$ that satisfies (2.19) there exist a map $\phi : \mathcal{X} \mapsto \mathcal{H}$ to a *reproducing Hilbert space* such that (2.18) holds [23]. Kernels can thus be regarded as a generalization of the inner product and they can be used to design a large variety of similarity measures. One of the key advantages of using kernels is that we do not need to explicitly know the feature map $\phi(\cdot)$ in order to compute them. This fact is of practical importance for classification problems using *support vector machines* [44] (cf. Section 2.5.2.2).

2.3.3.1 Distance-based kernels

Given a distance between two data points $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ it is possible to build a kernel from the distance metric using a Radial Basis Function (RBF) kernel. The RBF kernel is given by

$$\kappa_R(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\, d^2(\mathbf{x}, \mathbf{x}')\right), \tag{2.20}$$

where $\gamma \in \mathbb{R}_+$ is a parameter and $d(\cdot, \cdot)$ is some distance. For the case where the data points are LDSs we can use any of the distance metrics presented in the previous sections.

In some cases, it is also possible to have a closed-form expression for the kernel between two LDSs $\mathcal{M}$ and $\mathcal{M}'$. For example, the Martin kernel can be computed based on the subspace angles between the LDSs as

$$\kappa_M(\mathcal{M}, \mathcal{M}') = \prod_{i=1}^{n} \cos^2(\theta_i), \tag{2.21}$$

which is nothing but an RBF kernel on the Martin distance with $\gamma = 1$.

2.3.3.2 Binet-Cauchy kernels

Vishwanathan *et al.*, [33] introduced a family of kernels for LDSs called the *Binet-Cauchy* (BC) kernels. Such kernels depend on not only on the parameters $(\mathbf{A}, \mathbf{C}, \mathbf{B})$, but also on the initial condition $\mathbf{s}_0$. More precisely, given two LDS models $\mathcal{M} = (\mathbf{A}, \mathbf{C}, \mathbf{B}, \mathbf{s}_0)$ and $\mathcal{M}' = (\mathbf{A}', \mathbf{C}', \mathbf{B}', \mathbf{s}'_0)$ whose corresponding trajectories are $\mathbf{x} = \{\mathbf{z}_t\}_{t=0}^{\infty}$ and $\mathbf{x}' = \{\mathbf{z}'_t\}_{t=0}^{\infty}$, the trace kernel is defined as

$$\kappa_T(\mathbf{x}, \mathbf{x}') = \mathrm{E}\left[\sum_{t=0}^{\infty} \lambda^t \, \mathbf{z}_t^{\mathsf{T}} \boldsymbol{\Sigma} \, \mathbf{z}'_t\right], \tag{2.22}$$

where $0 < \lambda < 1$ is a parameter to ensure convergence of the sum (2.22) and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ is a positive semi-definite matrix. The symbol $\mathrm{E}[\cdot]$ denotes expectation over the driving and measurement noise processes. Assuming the same driving noise but independent measurement noise for the two trajectories, and using $\boldsymbol{\Sigma} = \mathbf{I}$, the trace

kernel particularizes to

$$\kappa_T(\mathbf{x}, \mathbf{x}') = \kappa_T(\mathcal{M}, \mathcal{M}') = \mathbf{s}_0^\mathsf{T} \mathbf{P}\, \mathbf{s}_0' + \frac{\lambda}{1-\lambda} \operatorname{Tr} \mathbf{B}\mathbf{P}\mathbf{B}', \tag{2.23}$$

where the matrix $\mathbf{P}$ is the solution to the Sylvester equation

$$\mathbf{P} = \lambda \mathbf{A}^\mathsf{T} \mathbf{P} \mathbf{A}' + \mathbf{C}^\mathsf{T} \mathbf{C}'. \tag{2.24}$$

In order to give different importance to each of the terms in (2.23) one can consider the following heuristic definition of the trace kernel:

$$\kappa_\eta(\mathcal{M}, \mathcal{M}') = \eta\, \mathbf{s}_0^\mathsf{T} \mathbf{P}\, \mathbf{s}_0' + \frac{(1-\eta)\lambda}{1-\lambda} \operatorname{Tr} \mathbf{B}\mathbf{P}\mathbf{B}', \tag{2.25}$$

where $\eta \in (0, 1)$ is a parameter that weights the contribution of each term.

It is clear that the value of the trace kernel in (2.23) and (2.25) depends on the initial conditions of the trajectories. In some applications, however, it is convenient to avoid such dependency (*e.g.*, gait classification problems). For that purpose, we will also consider in our analysis one special case of BC kernel, called the determinant kernel, which was proposed by [32]. This kernel is independent of the initial conditions and also invariant with respect to basis transformations. The determinant kernel is given by

$$\kappa_D(\mathbf{x}, \mathbf{x}') = \det \mathbf{P}. \tag{2.26}$$

In our experiments we will use the normalized versions of the kernels (2.23) and (2.26). For a given kernel $\kappa(\cdot, \cdot)$ its normalized version $\tilde{\kappa}(\cdot, \cdot)$ is obtained by

$$\tilde{\kappa}(\mathbf{x}, \mathbf{x}') = \frac{\kappa(\mathbf{x}, \mathbf{x}')}{\sqrt{\kappa(\mathbf{x}, \mathbf{x})\kappa(\mathbf{x}', \mathbf{x}')}}, \tag{2.27}$$

2.4 Parameter estimation

So far we have assumed that we have access to the true parameters of an LDS in order to compute distances and kernels in the space of LDSs. However, in reality we would only have access to an estimate of these parameters obtained from noisy measurements. The question is now how to estimate these parameters in order to be able to compute distances between LDSs. This is a well-studied subject and there exist algorithms that optimally (in the maximum likelihood sense) solve the parameter estimation problem [21, 22] (also known as *system identification*). However, when dealing with very high-dimensional signals (such as video) where the output dimension of the system is much larger than the dimension of the state variable (*i.e.*, $p \gg n$), the use of such methods becomes impractical due to the computational demands. Alternatively, one could resort to other suboptimal but more computationally efficient methods based on Principal Component Analysis (PCA) [16]. Due to the computational advantage of such techniques, we will use them throughout the thesis.

2.4.1 PCA-based system identification

In what follows, we will give the description of the PCA-based estimation method proposed in [16]. Assume that we have a zero-mean time series of length L given by $\{\mathbf{z}_t\}$, $t = 1, ..., L$, where $\mathbf{z}_t \in \mathbb{R}^p$ is the output of an LDS as per (2.1) and (2.2) with $p \gg n$. If the sequence is not of zero-mean, we just center it by subtracting the

sample mean. Let $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_L]$ be the concatenation of all observation vectors, and let the SVD of $\mathbf{Z}$ be equal to

$$\mathbf{Z} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}, \tag{2.28}$$

where $\mathbf{U} \in \mathbb{R}^{p\times p}$, $\mathbf{\Sigma} \in \mathbb{R}^{p\times L}$ and $\mathbf{V} \in \mathbb{R}^{L\times L}$. If we write $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_p]$, then an estimate of the observation matrix $\mathbf{C}$ can be obtained as the n first columns of $\mathbf{U}$ as

$$\hat{\mathbf{C}} = [\mathbf{u}_1, \ldots, \mathbf{u}_n], \tag{2.29}$$

where, in the case of $p \gg L$, the above computation can be performed efficiently by the thin SVD of $\mathbf{Z}$. Using (2.29) we can now get an estimate of the state sequence as

$$\hat{\mathbf{S}} = \hat{\mathbf{C}}^\mathsf{T}\mathbf{Z} = \mathbf{\Sigma}_n\mathbf{V}_n^\mathsf{T} = [\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_L], \tag{2.30}$$

where $\mathbf{\Sigma}_n = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$, with σ_i being the ith singular value of $\mathbf{Z}$, and where $\mathbf{V}_n = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ are the first n columns of $\mathbf{V}$.

Let $\hat{\mathbf{S}}^+ = [\hat{\mathbf{s}}_2, \ldots, \hat{\mathbf{s}}_L]$ and $\hat{\mathbf{S}}^- = [\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_{L-1}]$, then an estimate of the transition matrix $\mathbf{A}$ can be obtained by the solution of the following least-squares problem

$$\hat{\mathbf{A}} = \arg\min_{\mathbf{A}} \|\hat{\mathbf{S}}^+ - \mathbf{A}\hat{\mathbf{S}}^-\|_F^2. \tag{2.31}$$

Note that estimates of the covariance matrices of the driving and measurement noises can be easily obtained from the corresponding residuals.

2.4.1.1 Learning stable LDSs

It is clear that the PCA-based approach in [16] is computationally appealing, particularly when the output dimension is very large as compared to the order of the

system. However, the method does not guarantee the estimated systems to be stable. Since we are assuming our systems to be stable, we need an estimation method that can guarantee stability of the estimated systems. The stability of the system only depends on the transition matrix $\mathbf{A}$. Therefore, we can still estimate $\mathbf{C}$ using (2.29) but we need to modify the way in which $\mathbf{A}$ is estimated. An intuitive way of enforcing stability is to solve a constrained least-squares problem of the form

$$\begin{aligned} \underset{\mathbf{A}}{\text{minimize}} \quad & \|\hat{\mathbf{S}}^{+} - \mathbf{A}\hat{\mathbf{S}}^{-}\|_F^2 \\ \text{subject to} \quad & \rho(\mathbf{A}) \leq 1, \end{aligned} \tag{2.32}$$

where $\rho(\mathbf{A}) = \max_i |\lambda_i(\mathbf{A})|$ is the *spectral radius* of matrix $\mathbf{A}$. However, problem (2.32) is very difficult to solve due to the non-convexity of the spectral radius function. Alternatively, we can use the approach proposed in [45], where the spectral radius is replaced by a constraint on the maximum singular value of $\mathbf{A}$ (*i.e.*, $\sigma_{\max}(\mathbf{A}) \leq 1$). This leads to the convex problem

$$\begin{aligned} \underset{\mathbf{A}}{\text{minimize}} \quad & \|\hat{\mathbf{S}}^{+} - \mathbf{A}\hat{\mathbf{S}}^{-}\|_F^2 \\ \text{subject to} \quad & \sigma_{\max}(\mathbf{A}) \leq 1, \end{aligned} \tag{2.33}$$

which can be efficiently solved using general-purpose convex optimization software such as SeDuMi [46]. Note also that the constraint in (2.36) can be equivalently expressed as a semidefinite constraint since

$$\sigma_{\max}(\mathbf{A}) \leq 1 \Longleftrightarrow \mathbf{I} - \mathbf{A}\mathbf{A}^{\mathsf{T}} \succeq \mathbf{0} \Longleftrightarrow \begin{bmatrix} \mathbf{I} & \mathbf{A}^{\mathsf{T}} \\ \mathbf{A} & \mathbf{I} \end{bmatrix} \succeq \mathbf{0}, \tag{2.34}$$

where the last equivalence holds due to the Schur complement (see, *e.g.*, [47]). The problem with the formulation in (2.36) is that the constraint might be too conservative

since it constitutes a *sufficient* but not a necessary condition for stability. To overcome that issue, the authors in [48] propose to use a constraint generation approach that iteratively adds constraints to the least-squares problem until it finds a stable solution. Initially, the method starts with the least-squares solution $\hat{\mathbf{A}}$ of (2.31). If $\hat{\mathbf{A}}$ is stable, then we are done otherwise, let $\hat{\mathbf{A}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}$ be the SVD of $\hat{\mathbf{A}}$ and consider the fact that

$$\rho(\hat{\mathbf{A}}) \leq \sigma_{\max}(\hat{\mathbf{A}}) = \tilde{\mathbf{u}}_1^{\mathsf{T}}\hat{\mathbf{A}}\tilde{\mathbf{v}}_1 = \mathrm{Tr}\left(\tilde{\mathbf{v}}_1\tilde{\mathbf{u}}_1^{\mathsf{T}}\hat{\mathbf{A}}\right) = \mathrm{Tr}\left(\mathbf{G}_1\hat{\mathbf{A}}\right), \tag{2.35}$$

where $\tilde{\mathbf{u}}_1$ and $\tilde{\mathbf{v}}_1$ are the left and right singular vectors corresponding to the largest singular value, and where $\mathrm{Tr}(\cdot)$ is the trace operator. If the matrix $\hat{\mathbf{A}}$ is unstable, then $\mathrm{Tr}\left(\mathbf{G}_1\hat{\mathbf{A}}\right) > 1$. Note that $\mathbf{G}_1$ constitutes a separating hyperplane to the set of matrices with $\sigma_{\max} \leq 1$. Therefore, we can add the constraint $\mathrm{Tr}\left(\mathbf{G}_1\mathbf{A}\right) \leq 1$ to the least-squares problem (2.31) and recompute the solution again. This process is repeated, adding one constraint at a time until the found estimate $\hat{\mathbf{A}}$ is stable. In particular, at the kth iteration, we need to solve a problem of the form

$$\begin{aligned} \underset{\mathbf{A}}{\text{minimize}} \quad & \|\hat{\mathbf{S}}^{+} - \mathbf{A}\hat{\mathbf{S}}^{-}\|_F^2 \\ \text{subject to} \quad & \mathrm{Tr}\left(\mathbf{G}_i\mathbf{A}\right) \leq 1,\ i = 1, \ldots, k-1 \end{aligned} \tag{2.36}$$

where $\mathbf{G}_i$ are the generated constraints according to the described procedure. For further details, we refer the reader to [48].

2.5 Classification methods using LDSs

We have already defined metrics in the space of LDSs and shown how the systems can be identified using computationally efficient methods. Having the notion of a distance between LDSs allows us to perform classification tasks based on the estimated LDS parameters. Given a set of training time series with parameters $\mathcal{M}_i$, $i = 1, \ldots, m$, and associated labels $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is a set of labels, the goal is to learn a classifier function $h : \mathcal{M}_i \mapsto \mathcal{Y}$ that maps the parameters of the LDS models to its corresponding label. Throughout this section we will provide a review of some of the most common approaches used for the classification of time series generated by an LDS.

2.5.1 k-nearest neighbors classification

The simplest approach for classification is to use a k-nearest neighbors (kNN) classifier. In kNN, a novel point gets the label of the majority of the closest k neighbors, where the proximity is quantified based on some dissimilarity metric. More formally, given a set of m training model-label pairs $(\mathcal{M}_i, y_i)$, $i = 1, \ldots, m$, and a novel (test) model $\mathcal{M}_x$, the kNN classifier works as follows

- Compute the distance between the test sample and all the training samples $d(\mathcal{M}_i, \mathcal{M}_x)$, $i = 1, \ldots, m$ for some metric $d(\cdot, \cdot)$.

- Sort the distances in increasing order and take the first k models (k-nearest

neighbors).

- Assign to the test sample the class of the most repeated class among the k-nearest neighbors.

A typical choice is to only use the closest neighbor ($k = 1$), then the class of the test sample is estimated as $\hat{y}_x = y_j$, where

$$j = \underset{i=1,\ldots,m}{\arg\min}\ d(\mathcal{M}_i, \mathcal{M}_x). \tag{2.37}$$

2.5.2 Kernel support vector machines

An alternative approach to classifying LDSs is to use a kernel Support Vector Machine (SVM) [23]. In this section we will provide the basic foundations of linear SVMs and how they naturally extend to non-Euclidean classification problems using kernels.

2.5.2.1 Linear SVMs

Consider a binary classification problem with m feature-label training pairs $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, m$, where $\mathbf{x}_i \in \mathbb{R}^d$ and where the class labels $y_i \in \{-1, +1\}$. A support vector machine is a linear classifier that tries to separate the two classes using a prediction function of the form

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \mathbf{w}^\mathsf{T}\mathbf{x} + b, \tag{2.38}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the separating hyperplane and where $b \in \mathbb{R}$ is an offset. The label of a data point $\mathbf{x}$ is assigned according to the sign of $f(\mathbf{x})$, that is

$$\hat{y} = h(\mathbf{x}) = \operatorname{sign}\left(f(\mathbf{x})\right). \tag{2.39}$$

The parameters $(\mathbf{w}, b)$ of the classifier need to be learned from the training set by minimizing the regularized empirical risk. The general formulation of the two-class SVM classification problem [23] is given by

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \ell\left(y_i f(\mathbf{x}_i)\right), \tag{2.40}$$

where the first term is a regularizer that controls the complexity of the solution, $\ell : \mathbb{R} \mapsto \mathbb{R}_+$ is a (convex) loss-function (*e.g.*, exponential, hinge, etc.) that penalizes the misclassification of the training samples, and $C > 0$ is a parameter that trades-off between classifier complexity and classification accuracy. The typical choice for the loss function is to use the *hinge* loss given by $\ell_h(x) = \max(1 - x, 0)$. Using the hinge loss the problem can be rewritten as

$$\begin{aligned} \underset{\mathbf{w},\{\xi_i\},b}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^{\mathsf{T}}\mathbf{x}_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, m \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{2.41}$$

where $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_m]^{\mathsf{T}}$. It is well-known that the optimal hyperplane $\mathbf{w}^\star$ is a linear combination of the feature vectors [23] given by

$$\mathbf{w}^\star = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i \tag{2.42}$$

for some $\alpha_i \in \mathbb{R}$. The optimal predictor $f^\star(\mathbf{x})$ is then given by

$$f^\star(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i^\mathsf{T} \mathbf{x} + b^\star = \sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b^\star, \tag{2.43}$$

where $b^\star$ is the optimal offset obtained by solving 2.41.

For linearly separable data, the hyperplane found by solving (2.41) can be shown to provide the largest margin to the decision boundaries. In general, only a few coefficients α_i will be different from zero (if we use the hinge loss). The corresponding training points $\mathbf{x}_i$ of the non-zero coefficients α_i are known as the *support vectors* since only those define the optimal separating hyperplane.

2.5.2.2 Kernel SVMs

In many applications, linear separability is a rare event and one has to resort to non-linear decision functions. In such cases, linear SVMs fail to correctly classify the data. However, SVMs are intrinsically suitable for non-linear classification problems as it can be guessed by looking at the form of the optimal predictor in (2.43). To better illustrate this fact, let us have a look at the dual formulation of the SVM problem. From the Lagrangian of problem (2.41), it can be easily shown that the dual of problem (2.41) is given by the following quadratic optimization problem

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \boldsymbol{\alpha}^\mathsf{T}\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\mathsf{T}(\mathbf{K} \odot \mathbf{Y})\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha}^\mathsf{T}\mathbf{y} = 0 \\ & \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1} \end{aligned} \tag{2.44}$$

where $\mathbf{y} = [y_1, \ldots, y_m]^\mathsf{T}$, the symbol $\odot$ denotes element-wise product and $\mathbf{K}$ is a kernel matrix with entries $[\mathbf{K}]_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^\mathsf{T} \mathbf{x}_j$.

The dual formulation (2.44) of the primal SVM problem (2.41) encloses an important implication. As it can be seen from (2.44), in order to solve the problem, one no longer needs to explicitly know the feature vectors $\mathbf{x}_i$. It suffices to know how to compute their inner product. Furthermore, we could now replace $\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \mathbf{x}_i^\mathsf{T} \mathbf{x}_j$ with a positive-definite kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ since, by Mercer's theorem, there exist a feature map $\phi : \mathcal{X} \mapsto \mathcal{H}$ into some Hilbert space $\mathcal{H}$, $\mathbf{x}_i \in \mathcal{X}$, such that $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\mathsf{T} \phi(\mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$. A non-linear SVM classifier can be then obtained by replacing the original kernel matrix by (also known as the *kernel trick*)

$$[\mathbf{K}]_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j),\ i, j = 1, \ldots, m. \tag{2.45}$$

The advantage of using the dual formulation (2.44) and the kernel trick (2.45) is that now we can use the same machinery to solve problems were the classes cannot be linearly separated or where the data points do not live in an Euclidean space.

2.6 Chapter summary

In this chapter we have provided an introduction to linear dynamical systems, covering aspects such as metrics in the space of LDSs, system identification, as well as their use in classification problems. This background material provides the basic foundations to properly follow the forthcoming chapters.

Chapter 3

LDS Averaging and Clustering

This chapter is devoted to studying the problem of averaging and clustering when using LDSs as data points. There exist several reasons why one would be interested in computing such averages. For example, one may describe a class of dynamical processes based only on a small subset of representative points in order to build dictionaries of LDSs for representing dynamic scenes using a Bag-of-Words (BoW) type of approach, or for classification problems using a nearest mean criterion. Note also that in the latter case, there is a clear computational advantage (particularly for large scale problems) as compared to kNN since the computation of all pairwise distances between LDS models is no longer required but, only between each data point and the set of representative points or *exemplars*. The question then is how to construct those representative points. Having a method for computing averages clearly addresses this issue since we can find representative points by means of averaging (and clustering) a

number of data points belonging to the same class. Furthermore, having the notion of an average is also beneficial in the sense that it also opens the door to some statistical analysis of the data points.

Due to the modeling capabilities of LDSs for time series (*e.g.*, video or kinematic data), it is clear that the development of methods for computing averages in the space of LDSs is a problem of practical importance in computer vision applications. However, finding an average LDS poses a hard problem since the points (LDS models) do not live in an Euclidean space. There exist several works in the literature that deal with the problem of finding an average LDS model. For example, the work in [26] proposes an extension of K-means to the case where the data points are LDSs, and applies it to the problem of dynamic texture classification. In order to extract representative points, the authors propose to use an approximate averaging method to build the codebook of LDSs. The idea is to find a low-dimensional embedding into an Euclidean space based on the pairwise dissimilarity matrix between the models (*e.g.*, using the Martin distance and a non-linear dimensionality reduction technique such as multidimensional scaling) where standard Euclidean averaging can be applied. The average point is then defined (from the sample set) as the one corresponding to the closest point in the embedding to the Euclidean average. Other approaches [24, 25] find an embedding of the LDS into a Grassmann manifold and perform the averaging in the embedding. The principle behind these techniques is that an LDS can be represented as a point on the Grassmann manifold corresponding to the column

space of the (finite-dimensional) observability matrix [25]. It is important to note that these methods rely on approximate representations of the data points and that they are not able to generate novel LDS points since the computed averages are points already present in the sample set. Instead of using approximate representations of the data points to find the averages, there exist other approaches in the literature that try to find novel average models directly in the space of LDSs. The approach in [27], proposes a method for clustering and averaging LDSs based on the generative probabilistic framework of LDSs using the hierarchical Expectation Maximization algorithm. In a more recent contribution, the authors in [28] propose an averaging method using the Align distance that is able to generate novel LDS points.

This chapter presents new methods for averaging and clustering LDSs. In Section 3.1 we present two methods for averaging LDS. The first one, described in Section 3.1.1, is based on the Martin distance. The Martin distance is a particularly attractive metric since it is a true distance in the space of LDSs that is also invariant to basis transformations, a desired property when comparing dynamical models. To the best of our knowledge, the only previous attempt to produce an average LDS model using the Martin distance is the nonlinear dimensionality reduction method of [26]. However, the latter approach has its limitations in finding good averages since, as it has been already pointed out, it is not able to generate novel models but only identify potential candidates within the sample set. In contrast, we propose a new method

along the lines of [27, 28] that is able to generate novel LDSs. We pose the problem of averaging as an optimization problem over the system's parameters that tries to minimize the sum of squared distances to the average model (Fréchet mean). We will show how to compute the derivatives of the objective function with respect to the optimization variables and will propose a gradient descent algorithm for the solution of the problem. The second approach, described in Section 3.1.2, is an alternative to the methods presented in [28, 49] for the computation of the Align distance based on the Alternating Direction Method of Multipliers (ADMM) [31]. We will also review the averaging with respect to the Align distance as described in [28]. The method starts with an initial average model that is aligned to the sample points by finding an orthogonal basis transformation using the Align distance. Once all aligning matrices (orthogonal transformations) have been computed, an averaging step of the average model's parameters follows. This averaging pulls the current average model towards the true average. The same procedure is iteratively repeated until convergence to a local minimum. In Section 3.2 we will review the Generalized K-means algorithm, illustrating how the presented averaging methods can be used for clustering LDSs. Towards the end of the chapter, we will dedicate a section for experimental evaluation where we show that the proposed averaging method finds better average models than the method in [26] while providing results comparable to the state-of-the-art [28].

3.1 Averaging LDSs

Assume that we have $m > 1$ linear dynamical models $\mathcal{M}_i = (\mathbf{A}_i, \mathbf{C}_i)$, $i = 1, \ldots, m$ that parametrize some dynamic process or phenomenon of a certain class. We further assume that the models are all stable and of the same order n. The averaging problem is to find an average model $\mathcal{M} = (\mathbf{A}, \mathbf{C})$ with respect to some dissimilarity metric over the space of LDSs. More formally, we want to find the model that minimizes the sum of squared distances to the sample points, that is

$$\underset{\mathcal{M}}{\text{minimize}} \quad \sum_{i=1}^{m} d_X^2(\mathcal{M}, \mathcal{M}_i). \tag{3.1}$$

where (in our case) $d_X^2(\cdot, \cdot)$ can be the Martin or the Align distance.

3.1.1 Averaging using the Martin distance

Let us consider the averaging problem with respect to the Martin distance. Making use of Equation (2.13), the averaging problem (3.1) can be written as

$$\underset{\mathbf{A},\mathbf{C}}{\text{minimize}} \quad \sum_{i=1}^{m} d_M^2(\mathcal{M}, \mathcal{M}_i) = -\sum_{i=1}^{m} \log\det \mathbf{X}^{-1}\mathbf{X}_i\mathbf{S}_i^{-1}\mathbf{X}_i, \tag{3.2}$$

where

$$\mathbf{X} = \mathbf{A}^\mathsf{T}\mathbf{X}\mathbf{A} + \mathbf{C}^\mathsf{T}\mathbf{C} \tag{3.3}$$

$$\mathbf{X}_i = \mathbf{A}^\mathsf{T}\mathbf{X}_i\mathbf{A}_i + \mathbf{C}^\mathsf{T}\mathbf{C}_i \tag{3.4}$$

$$\mathbf{S}_i = \mathbf{A}_i^\mathsf{T}\mathbf{S}_i\mathbf{A}_i + \mathbf{C}_i^\mathsf{T}\mathbf{C}_i. \tag{3.5}$$

Using the properties of the determinant, and neglecting the terms that do not depend on the optimization variables $(\mathbf{A}, \mathbf{C})$, we have that the averaging problem with respect to the Martin distance can be expressed as

$$\underset{\mathbf{A},\mathbf{C}}{\text{minimize}} \quad \sum_{i=1}^{m} \log\det \mathbf{X} - \log\det \mathbf{X}_i^{\mathsf{T}}\mathbf{X}_i \tag{3.6}$$

Additionally, we will consider a representation of the models where the columns of $\mathbf{C}$ are orthonormal, that is

$$\mathbf{C}^{\mathsf{T}}\mathbf{C} = \mathbf{I}. \tag{3.7}$$

As pointed out in Section 2.4.1 there is no loss of generality by making such an assumption in the case of stable and observable models. Furthermore, the basis chosen for the representation of the LDSs does not affect our averaging method since the Martin distance is invariant to basis transformations. However, in order to compare with other metrics it is convenient to adopt a common representation. Another benefit of imposing $\mathbf{C}$ to be orthonormal is that it simplifies the computation of the derivative of the cost function with respect to $\mathbf{C}$ since we get rid of the term $\mathbf{C}^{\mathsf{T}}\mathbf{C}$ in the cost function.

What it is interesting from problem (3.6) is that it is possible to compute the derivatives of the cost function with respect to the optimization variables. This allows us to devise a gradient descent algorithm that iteratively updates $\mathbf{A}$ and $\mathbf{C}$ until convergence to a (local) minimum or until a maximum number of iterations is reached. Note also that since problem (3.6) is non-convex, convergence to the global

minimizer cannot be guaranteed. Additionally, we have to take into account the orthogonality constraint of Equation (3.7). Then our optimization problem becomes

$$\begin{aligned} \underset{\mathbf{A},\mathbf{C}}{\text{minimize}} \quad & \sum_{i=1}^{m} \log\det \mathbf{X} - \log\det \mathbf{X}_i^\mathsf{T}\mathbf{X}_i \\ \text{subject to} \quad & \mathbf{C}^\mathsf{T}\mathbf{C} = \mathbf{I} \end{aligned} \tag{3.8}$$

where $\mathbf{X}_i$ is given by Equation (3.4) and $\mathbf{X}$ now particularizes (due to the orthonormality of $\mathbf{C}$) to

$$\mathbf{X} = \mathbf{A}^\mathsf{T}\mathbf{X}\mathbf{A} + \mathbf{I} \tag{3.9}$$

During the optimization we will consider both $\mathbf{X}$ and $\mathbf{X}_i$, $i = 1, \ldots, m$ as matrix functionals and use matrix calculus to find the derivatives of the objective function with respect to the optimization variables $\mathbf{A}$ and $\mathbf{C}$.

3.1.1.1 Computing the derivatives

In order to compute the derivatives of the cost function in (3.8) we will make use of some facts about matrix derivatives [50]. Assuming all the inverses exist we have that for a matrix $\mathbf{X}$

$$\frac{\partial \log\det \mathbf{X}}{\partial \mathbf{X}} = (\mathbf{X}^{-1})^\mathsf{T} \tag{3.10}$$

$$\frac{\partial \log\det \mathbf{X}^\mathsf{T}\mathbf{X}}{\partial \mathbf{X}} = 2(\mathbf{X}^\dagger)^\mathsf{T} \tag{3.11}$$

where $\mathbf{X}^\dagger$ denotes the pseudo-inverse of $\mathbf{X}$. We also make use of the chain rule for matrix functionals. Let $g : \mathbb{R}^{\ell\times k} \mapsto \mathbb{R}$ be some scalar function and $\mathbf{Z} : \mathbb{R}^{s\times q} \mapsto \mathbb{R}^{\ell\times k}$

be a matrix functional. Assuming all derivatives exist we have that

$$\frac{\partial g(\mathbf{Z})}{\partial X_{ij}} = \operatorname{Tr}\left(\left(\frac{\partial g(\mathbf{Z})}{\partial \mathbf{Z}}\right)^{\mathsf{T}} \frac{\partial \mathbf{Z}(\mathbf{X})}{\partial X_{ij}}\right) \tag{3.12}$$

where $X_{ij} = [\mathbf{X}]_{ij}$ denotes the ijth entry of matrix $\mathbf{X}$ and where $\operatorname{Tr}(\cdot)$ is the trace operator.

Let $f(\mathbf{A}, \mathbf{C})$ denote the cost function of problem (3.8). The derivative of $f(\cdot)$ with respect to the ijth entry of matrix $\mathbf{A}$ can be computed in closed-form as

$$\frac{\partial f(\cdot)}{\partial A_{ij}} = m \operatorname{Tr}\left((\mathbf{X}^{-1})^{\mathsf{T}} \frac{\partial \mathbf{X}}{\partial A_{ij}}\right) - 2\sum_{s=1}^{m} \operatorname{Tr}\left((\mathbf{X}_s^{\dagger})^{\mathsf{T}} \frac{\partial \mathbf{X}_s}{\partial A_{ij}}\right) \tag{3.13}$$

Applying the product rule for derivatives it can be easily shown that the derivatives of $\mathbf{X}$ and $\mathbf{X}_s$ are, respectively given by

$$\frac{\partial \mathbf{X}}{\partial A_{ij}} = \mathbf{A}^{\mathsf{T}} \frac{\partial \mathbf{X}}{\partial A_{ij}} \mathbf{A} + \mathbf{J}_{ij}^{\mathsf{T}} \mathbf{X} \mathbf{A} + \mathbf{A}^{\mathsf{T}} \mathbf{X} \mathbf{J}_{ij} \tag{3.14}$$

$$\frac{\partial \mathbf{X}_s}{\partial A_{ij}} = \mathbf{A}^{\mathsf{T}} \frac{\partial \mathbf{X}_s}{\partial A_{ij}} \mathbf{A}_s + \mathbf{J}_{ij}^{\mathsf{T}} \mathbf{X}_s \mathbf{A}_s \tag{3.15}$$

where $\mathbf{J}_{ij}$ is the single-entry matrix (zeros everywhere but a one at the ijth position). Note that Equations (3.14) and (3.15) correspond respectively, to a discrete Lyapunov and Sylvester equations whose solutions are unique provided $\mathbf{A}$ and $\mathbf{A}_s$ have no complementary eigenvalues, that is

$$\lambda_k(\mathbf{A}) \neq 1/\lambda_j(\mathbf{A}_s), \text{ for all } k, j. \tag{3.16}$$

Note that the above condition is always true provided that $\rho(\mathbf{A}_s) < 1$. Enforcing stability of the models does not guarantee uniqueness in the solutions of Equations

(3.14) and (3.15) since periodic systems will have associated eigenvalues on the unit circle. However, for most practical situations we could assume that stability implies eigenvalues strictly smaller than 1.

In a similar way, the derivative of $f(\cdot)$ with respect to the entries of $\mathbf{C}$ can be computed as

$$\frac{\partial f(\cdot)}{\partial C_{ij}} = -2 \sum_{s=1}^{m} \mathrm{Tr}\left((\mathbf{X}_s^{\dagger})^{\mathsf{T}} \frac{\partial \mathbf{X}_s}{\partial C_{ij}} \right) \tag{3.17}$$

where $\partial \mathbf{X}_i / \partial C_{ij}$ is the solution of the following Sylvester equation

$$\frac{\partial \mathbf{X}_s}{\partial C_{ij}} = \mathbf{A}^{\mathsf{T}} \frac{\partial \mathbf{X}_s}{\partial C_{ij}} \mathbf{A}_s + \mathbf{J}_{ij}^{\mathsf{T}} \mathbf{C}_s \ . \tag{3.18}$$

Since in problem (3.8) we are also imposing $\mathbf{C}$ to be orthonormal (*i.e.*, $\mathbf{C}$ belongs to the Stiefel manifold $\mathcal{S}_{n,p} = \{\mathbf{X} \in \mathbb{R}^{p \times n} | \mathbf{X}^{\mathsf{T}}\mathbf{X} = \mathbf{I}\}$), we also need to compute the derivative with respect to the Stiefel manifold. From Edelman *et al.* [51], we have that the gradient of $f(\cdot)$ with respect to $\mathcal{S}_{n,p}$ is given by

$$\mathbf{\Delta} = \boldsymbol{\nabla}_{\mathbf{C}} f - \mathbf{C} (\boldsymbol{\nabla}_{\mathbf{C}} f)^{\mathsf{T}} \mathbf{C} \ , \tag{3.19}$$

where $\boldsymbol{\nabla}_{\mathbf{C}} f$ denotes the derivative of $f(\cdot)$ with respect to all entries of $\mathbf{C}$ (*e.g.*, $[\boldsymbol{\nabla}_{\mathbf{C}} f]_{ij} = \partial f(\cdot) / \partial C_{ij}$) and where $\mathbf{\Delta}$ belongs to the tangent space $T_C \mathcal{S}_{n,p}$ of $f(\cdot)$ at $\mathbf{C}$. Given the tangent vector $\mathbf{\Delta} \in T_C \mathcal{S}_{n,p}$ the update of $\mathbf{C}$ along the geodesic can be computed using the exponential map

$$\mathsf{exp}_{\mathbf{C}}(\mathbf{\Delta}) = \mathbf{C}\mathbf{D} + \mathbf{\Phi}\mathbf{E}, \tag{3.20}$$

where $\mathbf{D}, \mathbf{E} \in \mathbb{R}^{n \times n}$ are given by

$$\begin{bmatrix} \mathbf{D} \\ \mathbf{E} \end{bmatrix} = \exp\left(\delta \begin{bmatrix} \mathbf{C}^{\mathsf{T}} \boldsymbol{\Delta} & -\mathbf{R}^{T} \\ \mathbf{R} & \mathbf{0} \end{bmatrix} \right) \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \tag{3.21}$$

with δ_C being the gradient step-size for the update of $\mathbf{C}$, $\boldsymbol{\Phi} \in \mathbb{R}^{p \times n}$ and $\mathbf{R} \in \mathbb{R}^{n \times n}$ are, respectively an orthonormal and upper-triangular matrices corresponding to the (economy-size) QR-decomposition of $(\mathbf{I} - \mathbf{C}\mathbf{C}^{\mathsf{T}})\boldsymbol{\Delta}$ and where

$$\exp(\mathbf{X}) = \mathbf{I} + \mathbf{X} + \frac{1}{2}\mathbf{X}^2 + \frac{1}{3!}\mathbf{X}^3 + \ldots = \sum_{k=0}^{\infty} \frac{1}{k!}\mathbf{X}^k \tag{3.22}$$

is the matrix exponential (using the convention $0! = 1$ and $\mathbf{X}^0 = \mathbf{I}$).

The complete optimization procedure is outlined in Algorithm 1 where $\boldsymbol{\nabla}_{\mathbf{A}} f(\cdot)$ denotes the derivative of $f(\cdot)$ with respect to all entries of $\mathbf{A}$, $\mathrm{qr}\,(\cdot)$ is a function that performs the QR-decomposition and where δ_A denotes the step-size used for the gradient update of $\mathbf{A}$. The algorithm updates iteratively $\mathbf{A}$ and $\mathbf{C}$ until some exit condition is reached (*e.g.*, small change in the objective function or maximum number of iterations reached). For the initialization of the optimization variables we could use one of the data points at random or we could use Euclidean averaging and projection. More specifically, we could initialize to

$$\mathbf{A}_0 = \frac{1}{m} \sum_{i=1}^{m} \mathbf{A}_i \tag{3.23}$$

$$\mathbf{C}_0 = \Pi_{\mathcal{S}_{n,p}} \left(\frac{1}{m} \sum_{i=1}^{m} \mathbf{C}_i \right) \tag{3.24}$$

where $\Pi_{\mathcal{S}_{n,p}}(\cdot)$ denotes projection on the Stiefel manifold. That is, let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$ be the singular value decomposition of matrix $\mathbf{X}$, then $\Pi_{\mathcal{S}_{n,p}}(\mathbf{X}) = \mathbf{U}\mathbf{V}^\mathsf{T}$. As an illustration, we have depicted in Figure 3.1 a sample realization of the evolution of the cost function as the number of iterations increases for a toy example where we average ten randomly generated sequences. A random LDS model is first generated with $\mathbf{A}$ stable and $\mathbf{C}$ orthonormal. Then, we generate 10 synthetic sequences of length 100 samples using the generated model parameters but with different driving and measurement noise (i.i.d. Gaussian with identity covariance in both cases). Once we have the synthetic sequences, we perform system identification of the parameters using the PCA-based method for learning stable systems of Chapter 2. We then perform the averaging over the estimated models. As it can be appreciated in Figure 3.1, the average squared distance decreases as the number of iterations increases until we reach a minimum of the objective function.

3.1.1.2 Computational complexity

In this subsection we provide a rough estimate of the computational complexity of Algorithm 1. The major computational burden of the algorithm resides in the computation of the derivatives of the cost function with respect to the optimization variables. For the computation of $\nabla_{\mathbf{A}} f$, the inversion of $m+1$ matrices of size n by n is needed, which means $O((m+1)n^3)$ operations. Additionally, each of the entries of $\nabla_{\mathbf{A}} f$ requires m times the computation of the trace of the product of two

Algorithm 1 - Average LDS Martin

Input: $\{\mathcal{M}_i = (\mathbf{A}_i, \mathbf{C}_i)\}$, $i = 1, \ldots, m$

$k \leftarrow 0$, $\mathbf{C}^{(0)} \leftarrow \mathbf{C}_0$ and $\mathbf{A}^{(0)} \leftarrow \mathbf{A}_0$

repeat

  Compute $\boldsymbol{\nabla}_{\mathbf{A}} f\left(\mathbf{A}, \mathbf{C}^{(k)}\right)$ using (3.13), (3.14) and (3.15)

  $\mathbf{A}^{(k+1)} \leftarrow \mathbf{A}^{(k)} + \delta_A \, \boldsymbol{\nabla}_{\mathbf{A}} f\left(\mathbf{A}^{(k)}, \mathbf{C}^{(k)}\right)$

  Compute $\boldsymbol{\nabla}_{\mathbf{C}} f\left(\mathbf{A}^{(k+1)}, \mathbf{C}\right)$ using (3.17) and (3.18)

  Compute tangent vector $\boldsymbol{\Delta}$ using (3.19)

  $[\boldsymbol{\Phi}, \mathbf{R}] \leftarrow \mathrm{qr}\left((\mathbf{I} - \mathbf{C}^{(k)}(\mathbf{C}^{(k)})^{\mathsf{T}})\boldsymbol{\Delta}\right)$

  Compute $\mathbf{D}$ and $\mathbf{E}$ using (3.21)

  $\mathbf{C}^{(k+1)} \leftarrow \mathbf{C}^{(k)}\mathbf{D} + \boldsymbol{\Phi}\mathbf{E}$

  $k \leftarrow k + 1$

until Exit Condition

Output: $\mathcal{M} = (\mathbf{A}^{(k)}, \mathbf{C}^{(k)})$

matrices $O((m+1)n)$ as well as the calculation of $\partial \mathbf{X}_s / \partial A_{ij}$ and $\partial \mathbf{X} / \partial A_{ij}$. A naive implementation would require the solution of one Lyapunov or Sylvester equation $O(n^3)$ for each entry of the derivatives, which in turn will translate into $O((m+1)n^5)$ and $O(mpn^4)$ operations for the derivatives with respect to $\mathbf{A}$ and $\mathbf{C}$, respectively. However, such computation can be simplified by the observation that an equation of the form $\mathbf{X} = \mathbf{A}\mathbf{X}\mathbf{A}_i + \mathbf{Z}$ is equivalent to

$$\left(\mathbf{I} - \mathbf{A}_i^{\mathsf{T}} \otimes \mathbf{A}\right) \mathrm{vec}\left(\mathbf{X}\right) = \mathrm{vec}\left(\mathbf{Z}\right), \tag{3.25}$$

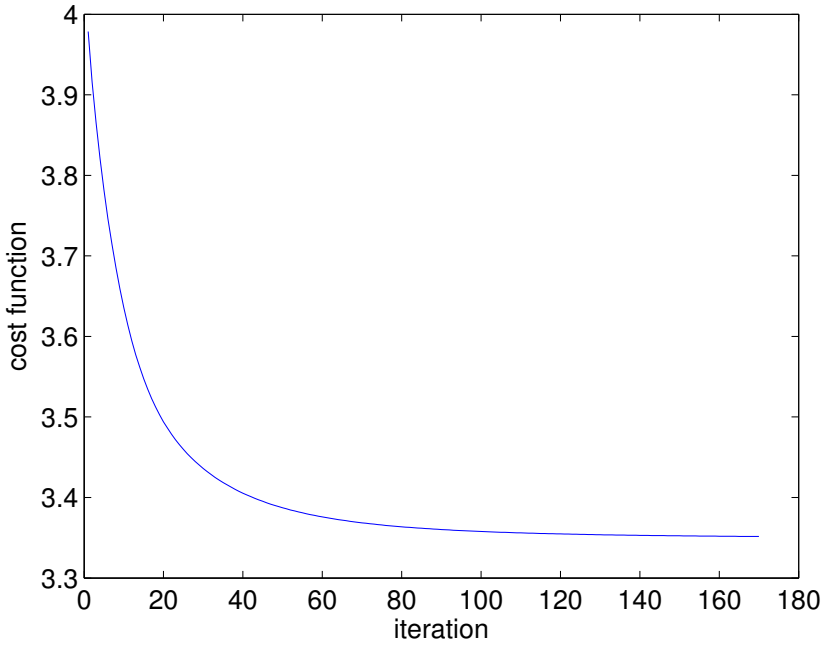

Figure 3.1: **Averaging using the Martin distance** – Sample evolution of the cost function (mean squared Martin distance to the average model) as the number of iterations increases.

where $\otimes$ is the Kronecker product and $\mathrm{vec}\,(\mathbf{X})$ is the vectorized form of matrix $\mathbf{X}$. Therefore, we only need to solve one equation of the form in (3.25) per model and particularize the result for each of the entries in the derivative since in (3.14), (3.15) and (3.18) the only term that changes is the independent term. As a result, we need to add $O((m+1)n^6)$ for the solution of an equation of the form of (3.25) at each iteration plus $O(n^3)$ and $O(pn^2)$ operations corresponding to the computation of the derivatives with respect to $\mathbf{A}$ and $\mathbf{C}$. Analogously, the computation of $\boldsymbol{\nabla}_{\mathbf{C}} f$ requires on the order of $O(pmn^2 + mn^3)$ operations plus the computation of the derivatives.

Putting everything together, the total complexity of Algorithm 1 is roughly $O(p(m+1)n^2 + (m+1)n^6 + 3(m+1)n^3)$. It is easy to realize that the described approach for the implementation presents a computational advantage provided that $p > n^2$, which is generally the case in computer vision applications. Overall, we can conclude that the complexity of Algorithm 1 increases linearly in both the number of models m as well as in the output dimension p.

3.1.2 Averaging using the Align distance

In this section we will consider again the averaging problem in (3.1), but this time with respect to the Align distance. A method for averaging w.r.t. Align was proposed in [28]. We will provide a review of the method but we will consider a novel way for the computation of the distance. The original method for the computation of the Align distance [28] is based on a gradient-descent optimization over the orthogonal group $O(n)$. In a very recent contribution, an alternative method based Jacobi-type of updates which requires finding the roots of quartic polynomials has been proposed in [49]. We will use an alternative approach based on the Alternating Direction Method of Multipliers (ADMM) [31]. The main advantage of the new formulation is that it allows the derivation of simple closed-form updates for the optimization variable, making it easy to implement and computationally appealing. Our method generally gives better averaging results than the method in [49], however convergence to a solution is generally slower. Although there are no guarantees for convergence

since the optimization set is non-convex (*i.e.*, orthogonal group) in practice we observe a good behavior.

3.1.2.1 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers is a general approach for solving optimization problems that dates back to the 60's and that has attracted the attention of the researchers over the past years. The method relies on augmented Lagrangian [52] and dual ascent and has been recently reviewed in [31] showing its application to many interesting problems in the fields of statistics and signal processing. We will consider here one special class of problems where the method can be applied. In particular, consider an optimization problem of the form

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{3.26}$$

with optimization variable $\mathbf{x}$ and where $f(\mathbf{x})$ is a convex function and $\mathcal{X}$ is some (possibly non-convex) set. The ADMM method tries to find the solution of the problem (3.26) by the following iterative procedure

$$\mathbf{x}^{(k+1)} = \arg\min_{\mathbf{x}} \; f(\mathbf{x}) + \rho/2 \|\mathbf{x} - \mathbf{q}^{(k)} + \mathbf{u}^{(k)}\|_2^2 \tag{3.27}$$

$$\mathbf{q}^{(k+1)} = \Pi_{\mathcal{X}} \left(\mathbf{x}^{(k+1)} + \mathbf{u}^{(k)} \right) \tag{3.28}$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{q}^{(k+1)} \tag{3.29}$$

where $\mathbf{q}^{(\cdot)}$ is the variable that we are interested in, $\mathbf{x}^{(\cdot)}$ is an auxiliary variable, $\mathbf{u}^{(\cdot)}$ is the dual variable, and where $\Pi_{\mathcal{X}}(\cdot)$ denotes the projection onto the set $\mathcal{X}$. When the

set is convex, the above procedure converges to the optimal solution of the problem [31]. When the set $\mathcal{X}$ is not convex, convergence (even to local minima) is not guaranteed in general. However, the procedure can still be used and is of particular interest when the projection update can be computed in closed-form.

3.1.2.2 An ADMM approach for computing the Align distance

Recall from Section 2.3.2, that the Align distance between two dynamical models $\mathcal{M}_1 = (\mathbf{A}_1, \mathbf{C}_1)$ and $\mathcal{M}_2 = (\mathbf{A}_2, \mathbf{C}_2)$ of order n can be computed as

$$\begin{aligned} d_A^2(\mathcal{M}_1, \mathcal{M}_2) &= \min_{\mathbf{Q} \in O(n)} \lambda_A \|\mathbf{Q}^\mathsf{T} \mathbf{A}_1 \mathbf{Q} - \mathbf{A}_2\|_F^2 + \lambda_C \|\mathbf{C}_1 \mathbf{Q} - \mathbf{C}_2\|_F^2 \\ &= \min_{\mathbf{Q} \in O(n)} \lambda_A \|\mathbf{A}_1 \mathbf{Q} - \mathbf{Q} \mathbf{A}_2\|_F^2 + \lambda_C \|\mathbf{C}_1 \mathbf{Q} - \mathbf{C}_2\|_F^2 \end{aligned} \tag{3.30}$$

where we have set $\lambda_B = 0$ in (2.17), and where the second equality holds because the Frobenius norm is invariant to orthogonal transformations. It is easy to realize that the Align distance falls into the category of problem (3.26). The ADMM procedure particularizes then to

$$\mathbf{X}^{(k+1)} = \arg\min_{\mathbf{X}} \; f_\rho(\mathbf{X}) \tag{3.31}$$

$$\mathbf{Q}^{(k+1)} = \Pi_{O(n)} \left(\mathbf{X}^{(k+1)} + \mathbf{U}^{(k)} \right) \tag{3.32}$$

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \mathbf{X}^{(k+1)} - \mathbf{Q}^{(k+1)}, \tag{3.33}$$

where

$$f_\rho(\mathbf{X}) = \lambda_A \|\mathbf{A}_1 \mathbf{X} - \mathbf{X} \mathbf{A}_2\|_F^2 + \lambda_C \|\mathbf{C}_1 \mathbf{X} - \mathbf{C}_2\|_F^2 + \rho/2 \|\mathbf{X} - \mathbf{Q}^{(k)} + \mathbf{U}^{(k)}\|_F^2. \tag{3.34}$$

As it can be realized, the update in (3.31) corresponds to solving an unconstrained linear least-squares problem in the variable $\mathbf{X}$. In fact, it is possible to solve (3.31) in closed-form by setting the derivative of $f_\rho(\cdot)$ to zero. For the computation of $\partial f_\rho / \partial \mathbf{X}$ we will use matrix calculus [50] and, in order to make it easier to read, we will consider the contribution of each term separately (modulo its weight). The first term of $f_\rho(\cdot)$ is given by

$$\begin{aligned} \|\mathbf{A}_1\mathbf{X} - \mathbf{X}\mathbf{A}_2\|_F^2 = \operatorname{Tr}\big(&\mathbf{A}_1\mathbf{X}\mathbf{X}^\mathsf{T}\mathbf{A}_1^\mathsf{T} - \mathbf{X}\mathbf{A}_2\mathbf{X}^\mathsf{T}\mathbf{A}_1^\mathsf{T} \\ &-\mathbf{A}_1\mathbf{X}\mathbf{A}_2^\mathsf{T}\mathbf{X}^\mathsf{T} + \mathbf{X}\mathbf{A}_2\mathbf{A}_2^\mathsf{T}\mathbf{X}^\mathsf{T}\big) \end{aligned} \tag{3.35}$$

with derivative

$$\frac{\partial \|\mathbf{A}_1\mathbf{X} - \mathbf{X}\mathbf{A}_2\|_F^2}{\partial \mathbf{X}} = 2\left(\mathbf{A}_1^\mathsf{T}\mathbf{A}_1\mathbf{X} - \mathbf{A}_1^\mathsf{T}\mathbf{X}\mathbf{A}_2 - \mathbf{A}_1\mathbf{X}\mathbf{A}_2^\mathsf{T} + \mathbf{X}\mathbf{A}_2\mathbf{A}_2^\mathsf{T}\right). \tag{3.36}$$

The second term reads

$$\|\mathbf{C}_1\mathbf{X} - \mathbf{C}_2\|_F^2 = \operatorname{Tr}\left(\mathbf{C}_1\mathbf{X}\mathbf{X}^\mathsf{T}\mathbf{C}_1^\mathsf{T} - \mathbf{C}_2\mathbf{X}^\mathsf{T}\mathbf{C}_1^\mathsf{T} - \mathbf{C}_1\mathbf{X}\mathbf{C}_2^\mathsf{T} + \mathbf{C}_2\mathbf{C}_2^\mathsf{T}\right) \tag{3.37}$$

whose derivative is

$$\frac{\partial \|\mathbf{C}_1\mathbf{X} - \mathbf{C}_2\|_F^2}{\partial \mathbf{X}} = 2\left(\mathbf{C}_1^\mathsf{T}\mathbf{C}_1\mathbf{X} - \mathbf{C}_1^\mathsf{T}\mathbf{C}_2\right). \tag{3.38}$$

Lastly, the third term is given by

$$\begin{aligned} \|\mathbf{X} - \mathbf{Q} + \mathbf{U}\|_F^2 = \frac{\rho}{2}\operatorname{Tr}\big(&(\mathbf{U} - \mathbf{Q})\mathbf{X}^\mathsf{T} + (\mathbf{U} - \mathbf{Q})(\mathbf{U} - \mathbf{Q})^\mathsf{T} \\ &+\mathbf{X}(\mathbf{U} - \mathbf{Q})^\mathsf{T} + \mathbf{X}\mathbf{X}^\mathsf{T}\big) \end{aligned} \tag{3.39}$$

and its derivative by

$$\frac{\partial \|\mathbf{X} - \mathbf{Q} + \mathbf{U}\|_F^2}{\partial \mathbf{X}} = 2\left(\mathbf{X} - \mathbf{Q} + \mathbf{U}\right). \tag{3.40}$$

If we put all the terms together with their corresponding weights and set the it to zero we have that

$$\begin{aligned} &2\lambda_A \left(\mathbf{A}_1^\top \mathbf{A}_1 \mathbf{X} - \mathbf{A}_1^\top \mathbf{X} \mathbf{A}_2 - \mathbf{A}_1 \mathbf{X} \mathbf{A}_2^\top + \mathbf{X} \mathbf{A}_2 \mathbf{A}_2^\top\right) \\ &+ 2\lambda_C \left(\mathbf{C}_1^\top \mathbf{C}_1 \mathbf{X} - \mathbf{C}_1^\top \mathbf{C}_2\right) + 2\frac{\rho}{2}\left(\mathbf{X} - \mathbf{Q} + \mathbf{U}\right) = \mathbf{0}, \end{aligned} \tag{3.41}$$

which is a linear equation in $\mathbf{X}$. Rearranging terms and getting rid of the 2 scaling factor we end up with

$$\begin{aligned} \left[\frac{\rho}{2}\mathbf{I} + \lambda_C \mathbf{C}_1^\top \mathbf{C}_1 + \lambda_A \mathbf{A}_1^\top \mathbf{A}_1\right] \mathbf{X} + \lambda_A \left(\mathbf{X} \mathbf{A}_2 \mathbf{A}_2^\top - \mathbf{A}_1^\top \mathbf{X} \mathbf{A}_2 - \mathbf{A}_1 \mathbf{X} \mathbf{A}_2^\top\right) \\ = \lambda_C \mathbf{C}_1^\top \mathbf{C}_2 + \frac{\rho}{2}\left(\mathbf{Q} - \mathbf{U}\right). \end{aligned} \tag{3.42}$$

One simple, although not the most efficient, way of solving (3.42) is by using the Kronecker product and rewriting equation (3.42) as

$$\begin{aligned} &\Big[\lambda_A \left(\mathbf{I} \otimes \mathbf{A}_1^\top \mathbf{A}_1 + \mathbf{A}_2 \mathbf{A}_2^\top \otimes \mathbf{I} - \mathbf{A}_2^\top \otimes \mathbf{A}_1^\top - \mathbf{A}_2 \otimes \mathbf{A}_1\right) \\ &+ \frac{\rho}{2}\mathbf{I} + \lambda_C \mathbf{I} \otimes \mathbf{C}_1^\top \mathbf{C}_1\Big] \operatorname{vec}\left(\mathbf{X}\right) = \operatorname{vec}\left(\lambda_C \mathbf{C}_1^\top \mathbf{C}_2 + \frac{\rho}{2}\left(\mathbf{Q} - \mathbf{U}\right)\right). \end{aligned} \tag{3.43}$$

The complete procedure for computing the Align distance is summarized in Algorithm 2. The algorithm takes as inputs two LDS models $\mathcal{M}_1$ and $\mathcal{M}_2$, and the penalty parameter ρ and it returns the Align distance between the models. The iterative procedure in Algorithm 2 stops when some exit condition is met (*e.g.*, maximum number of iterations or small relative change in the objective function and/or in the update of the optimization variables).

Remark: Since the orthogonal group has two disjoint connected components (*i.e.*, one with determinant $+1$ and another one with determinant -1), the procedure

Algorithm 2 - Computation of the Align distance using ADMM

1: **Input:** $(\mathcal{M}_1, \mathcal{M}_2, \rho)$

2: $k \leftarrow 0$, $\mathbf{U}^{(0)} \leftarrow \mathbf{0}$, $\mathbf{Q}^{(0)} \leftarrow \mathbf{0}$

3: **repeat**

4: $k \leftarrow k + 1$

5: Compute $\mathbf{X}^{(k)}$ using (3.42)

6: Projection onto $O(n)$

$[\mathbf{S}\,\mathbf{V}\,\mathbf{D}] \leftarrow \text{svd}\left(\mathbf{X}^{(k)}\right)$

$\mathbf{Q}^{(k)} \leftarrow \mathbf{S}\,\mathbf{D}^{\mathsf{T}}$

7: Update multiplier

$\mathbf{U}^{(k)} \leftarrow \mathbf{U}^{(k-1)} + \mathbf{X}^{(k)} - \mathbf{Q}^{(k)}$

8: Evaluate objective (Align distance)

$f^{(k)} = \lambda_A \left\|\mathbf{A}_1\mathbf{Q}^{(k)} - \mathbf{Q}^{(k)}\mathbf{A}_2\right\|_F^2 + \lambda_C \left\|\mathbf{C}_1\mathbf{Q}^{(k)} - \mathbf{C}_2\right\|_F^2$

9: **until** Exit Condition

10: **Output:** $(f^{(k)}, \mathbf{Q}^{(k)})$

outlined in Algorithm 2 needs to be run separately on each of the two components. The distance is then selected as the minimum over the two runs.

3.1.2.3 Align average

Using the procedure described in Algorithm 2 we can now solve the averaging problem (3.1) with respect to the Align distance following the approach presented in

Algorithm 3 - ADMM for averaging using the Align distance

Input: $\{\mathcal{M}_i = (\mathbf{A}_i, \mathbf{C}_i)\}$, $i = 1, \ldots, m$

$k \leftarrow 0$, $\mathbf{C}^{(0)} \leftarrow \mathbf{C}_0$ and $\mathbf{A}^{(0)} \leftarrow \mathbf{A}_0$

repeat

  for $i = 1, \ldots, m$ **do**

    Compute the aligning matrix $\mathbf{Q}_i$ between $\mathcal{M}_i$

    and $\mathcal{M} = (\mathbf{A}^{(k)}, \mathbf{C}^{(k)})$ using Algorithm 2

  end for

  Update of $\mathbf{A}$

  $\mathbf{A}^{(k+1)} \leftarrow \frac{1}{m} \sum_i \mathbf{Q}_i^{\mathsf{T}} \mathbf{A}_i$

  Update of $\mathbf{C}$ (Euclidean average + projection)

  $\mathbf{C}^{(k+1)} \leftarrow \Pi_{\mathcal{S}_{n,p}} \left(\frac{1}{m} \sum_i \mathbf{C}_i \mathbf{Q}_i \right)$

  $k \leftarrow k + 1$

until Exit Condition

Output: $\mathcal{M} = (\mathbf{A}^{(k)}, \mathbf{C}^{(k)})$

[28] that we outline in Algorithm 3. Starting with an initial average model $(\mathbf{A}^{(0)}, \mathbf{C}^{(0)})$, the averaging method in Algorithm 2 iteratively aligns the sample points $\mathcal{M}_i$ to the current average model. After that, an update of the current average model is performed. In the case of $\mathbf{A}$, the average is computed as the Euclidean average of the aligned LDS points while, in the case of $\mathbf{C}$, an additional projection step onto the Stiefel manifold follows the Euclidean averaging of the aligned realizations.

3.2 Clustering LDSs

A popular method for unsupervised clustering in Euclidean space is the K-means algorithm. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ be a collection of data points in Euclidean space ($\mathbf{x}_i \in \mathbb{R}^d$ for some d). The K-means clustering problem consists of finding a set of K centroids (cluster centers) $\bar{\mathcal{X}} = \{\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_K\}$ such that

$$\sum_{i=1}^{m} \min_{k=1,\ldots,K} ||\mathbf{x}_i - \bar{\mathbf{x}}_k||_2^2, \tag{3.44}$$

is minimized. Note that this problem can be generalized to non-Euclidean spaces by replacing the Euclidean distance with an appropriate distance in the non-Euclidean space. For instance, given a collection of LDS models $\mathcal{M}_1, \ldots, \mathcal{M}_m$, and a distance $d_X(\mathcal{M}_i, \mathcal{M}_j)$ in the space of LDSs, the K-means clustering problem for LDSs can be written as

$$\sum_{i=1}^{m} \min_{k=1,\ldots,K} d_X^2(\mathcal{M}_i, \bar{\mathcal{M}}_k), \tag{3.45}$$

where $\bar{\mathcal{M}}_k$, $k = 1, \ldots, K$, are the LDS cluster centers. An iterative method that tries to solve the K-means clustering problem is the *K-means algorithm*. It alternates between two steps: averaging and assignment. In the averaging step the set of cluster centers are computed based on the current assignment while, in the assignment step the different data points are assigned to belong to the cluster of the nearest centroid. The general procedure is outlined in Algorithm 4. It becomes clear that, given a distance in the space of LDSs (*e.g.*, Martin or Align) and its corresponding averaging method, Algorithm 4 can be used for clustering a collection of dynamical models $\mathcal{M}_i$,

Algorithm 4 - Generalized K-means for LDS

1: **Input:** LDS models $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ and initial cluster centers $\{\bar{\mathcal{M}}_1, \ldots, \bar{\mathcal{M}}_K\}$

2: **repeat**

3: Assignment step (for $i = 1, \ldots, m$)

Sample i gets assigned to cluster j_i where

$$j_i = \arg\min_k \; d_X(\mathcal{M}_i, \bar{\mathcal{M}}_k),\; k = 1, \ldots, K$$

4: Averaging step (for $k = 1, \ldots, K$)

$$\bar{\mathcal{M}}_k \leftarrow \arg\min_{\mathcal{M}} \; \textstyle\sum_{i \in \mathcal{I}_k} d_X^2(\mathcal{M}_i, \mathcal{M}), \text{ where } \mathcal{I}_k = \{i \,|\, j_i = k\}$$

5: **until** Exit condition

6: **Output:** LDS cluster centers $\{\bar{\mathcal{M}}_1, \ldots, \bar{\mathcal{M}}_K\}$

$i = 1, \ldots, m$. In this case, the algorithm can be stopped when a maximum number of iterations is reached or if the algorithm converges (*i.e.*, there are no changes in the assignment from one iteration to the next).

3.3 Experiments

In this section we will illustrate the performance of the proposed averaging methods by evaluating their performance on both synthetic and real datasets. Since our assumption is that the models are stable, we will use the constraint generation approach in [48] for system identification in order to enforce stability. We will refer to our averaging method using the Martin distance as "Martin Average" throughout the

experiments.

3.3.1 Experiments on synthetic data

We used the following procedure to synthesize time series data generated by an LDS. The dimensions of the generative model are set to $n = 3$ and $p = 20$. The entries of the state transition matrix $\mathbf{A}$ are generated at random from a Gaussian distribution of zero-mean and unit variance $\mathcal{N}(0, 1)$ and we pick one matrix that is stable. The entries of the observation matrix $\mathbf{C}$ are also generated from $\mathcal{N}(0, 1)$ and then, the matrix is projected to the Stiefel manifold. Once the model parameters have been generated, sequences of length $l = 100$ are synthesized using a Gaussian driving noise process with zero-mean and identity covariance. Measurement noise generated from $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$, where σ is a parameter, is then added to the synthesized sequences. At each iteration 10 such sequences are randomly generated and the parameters of the models are identified using the method in [48]. Then the 10 sequences are averaged using the nonlinear dimensionality reduction method of [26] with the Martin distance as a dissimilarity metric (MDS Average). The averages obtained are then compared with the proposed technique (Martin Average). The experiment was repeated for different values of the measurement noise standard deviation σ, and the results were averaged over 1000 realizations. We used a step-size of 10^{-4} for both the $\mathbf{A}$ and $\mathbf{C}$ updates and a maximum number of iterations equal to 100.

In Figure 3.2 we have displayed the 2-dimensional MDS embedding of ten gen-

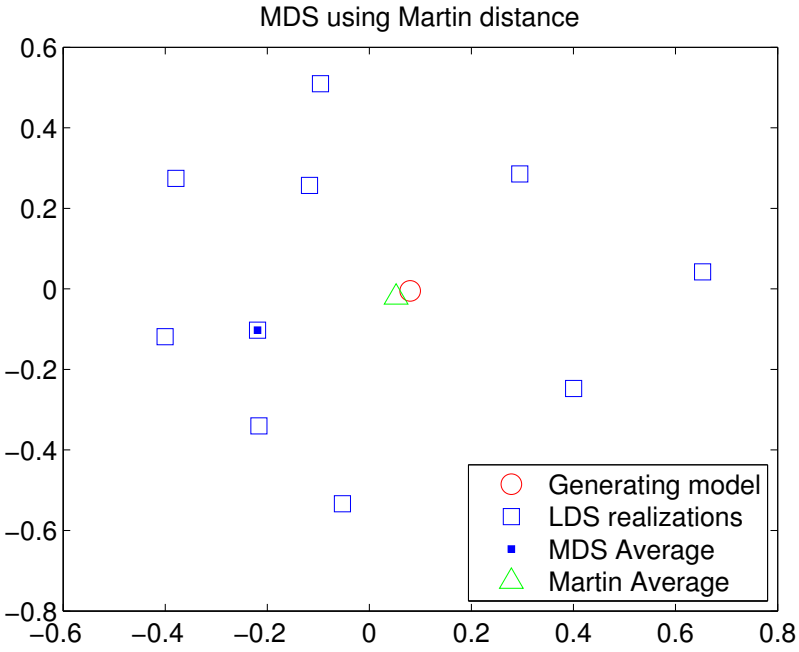

Figure 3.2: **MDS representation** – Multidimensional scaling representation of the data points together with the generating model and computed averages using the approximate MDS averaging and the proposed Martin averaging.

erated time series together with the generative and computed average models using the Martin distance as our dissimilarity measure. As it can be observed, the approximate averaging method in [26] (dark square) exhibits a higher error with respect to the true generative model since it is restricted to choose a point within the actual data samples, while the proposed technique (green triangle) is able to produce novel models that come very close to the actual underlying generative model (red circle). The average squared distance between the data points and the average models over 1000 realizations and for different values of the measurement noise standard devia-

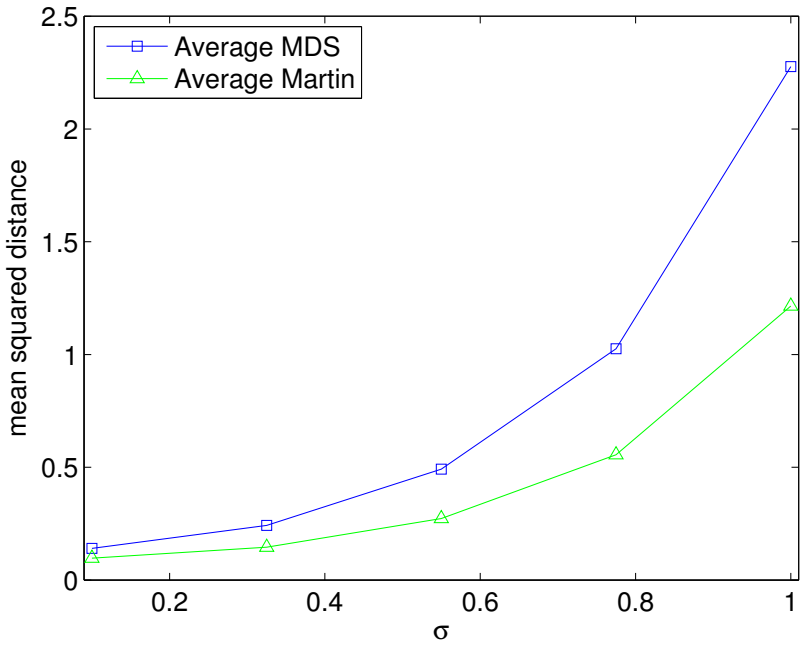

Figure 3.3: **Average error** – Average squared distance as a function of the measurement noise standard deviation.

tion is displayed in Figure 3.3. It can be appreciated that our averaging technique always provides a lower error than the MDS averaging and that the error is even more pronounced as the measurements become more noisy.

In order to compare our ADMM method for computing the Align distance to the one in [49] we generated systems at random from a Gaussian distribution with zero mean and unit variance, varying the order of the dynamical models and the output dimension. We averaged the obtained distances using the method in [49] and our method ($\rho = 50$)over 100 realizations. The results of the average squared distances using both methods are given in Table 3.1. It can be seen that our proposed scheme

achieves better averages than the one in [49], specially for smaller orders. If we look at the average computation times displayed in Table 3.2 we observe that our method is slower than the one in [49] (around 4 times slower for order 8). However, there is still room for improvement for the ADMM method since its convergence speed depends on the parameter ρ. Also for the computation of the update in (3.31) we are using the vectorized form of (), which is certainly not the most efficient way of computing the update. It is also worth to mention that the computational complexity increases slowly in the output dimension and that increasing the order of the models has a greater effect in the computation times.

3.3.2 Experiments of the Weizmann human action dataset

In order to test the performance of the proposed method in a real dataset, we performed an experiment on the Weizmann human action dataset [53]. The dataset consists of nine people performing 10 different natural actions (see Figure 3.4) like ‘‘run’’, ‘‘walk’’, ‘‘jump’’, ‘‘wave hands’’, etc. For each video, we compute the optical-flow and extract a bounding box time-series around the person of size $p = 63 \times 29 \times 2 = 3654$. From the extracted time-series we identify an LDS of order $n = 5$.

We conduct an experiment where each action is represented by its average model

Average squared distance					
	$p = 10$	$p = 50$	$p = 100$	$p = 500$	$p = 1000$
			Jacobi method		
$n = 4$	14.8647	15.5556	15.4506	16.0272	15.6611
$n = 6$	20.5571	22.4632	22.1456	22.6476	22.7694
$n = 8$	32.9441	35.2943	34.9150	36.0533	35.4104
			ADMM		
$n = 4$	12.4305	13.4099	13.5185	13.5058	13.9928
$n = 6$	18.9067	21.0748	20.3559	21.0285	21.1435
$n = 8$	32.8144	35.1214	34.5729	34.8349	35.3758

Table 3.1: Average squared distance for different orders and output dimensions.

and perform classification of a novel model based on the nearest mean. A leave-one-out cross-validation setup over all the sequences is used to evaluate the classification accuracy. The step-size of the averaging algorithm is set to $\delta_A = \delta_C = 5 \times 10^{-5}$.

The results in terms of classification accuracy are given in Table 3.3. As a comparison, we include the results for the MDS averaging method of [26] (MDS Average) and the averaging method of [28] presented in Section 3.1.2.3. In order to illustrate the improvement achieved through the optimization process, we have also included the results when no optimization is carried out (*i.e.*, the average models correspond to

Average computation time

	$p = 10$	$p = 50$	$p = 100$	$p = 500$	$p = 1000$
			Jacobi method		
$n = 4$	0.0009	0.0009	0.0008	0.0010	0.0011
$n = 6$	0.0029	0.0031	0.0030	0.0033	0.0032
$n = 8$	0.0090	0.0091	0.0102	0.0097	0.0102
			ADMM		
$n = 4$	0.0103	0.0126	0.0106	0.0117	0.0162
$n = 6$	0.0236	0.0282	0.0259	0.0305	0.0388
$n = 8$	0.0373	0.0359	0.0369	0.0460	0.0527

Table 3.2: Average computation time for different orders and output dimensions.

the initial values used for the optimization). The latter one is labeled as "Euclidean" since the initial points are obtained through Euclidean averaging (and projection onto $\mathcal{S}_{n,p}$ in the case of $\mathbf{C}_0$) as per (3.23) and (3.24).

As it can be observed, the proposed approach gives the best performance for this particular dataset. We also observe a significant improvement (around 20%) compared to the initial point where the optimization started (*i.e.*, Euclidean). Compared to the performance of a 1-NN classifier with the Martin distance (96.77%) we see that the degradation in performance due to the use of the average models is very small (around

Figure 3.4: **Weizmann dataset** – Human actions in the Weizmann dataset.

2%) while the savings in terms of the number of representative points and distance computations (at test time) is significantly reduced (by a factor of 10 since there are around 10 samples per class).

3.3.3 Experiments on the UCLA8 dynamic texture dataset

As it was already mentioned before, an averaging method is particularly useful for clustering purposes. In our case, we could use the proposed method for building a dictionary of LDS codewords and perform dynamic texture classification using Bag-of-Systems [26]. In order to build the codewords, we use the generalized K-means clustering algorithm described in Section 3.2 using both Martin and Align distances and their corresponding averaging methods. We use a view-invariant subset of the

Method	% Correct
Euclidean	75.27
MDS Average	90.32
Align [28]	93.55
Martin Average	**94.62**

Table 3.3: Nearest mean classification performance on the Weizmann human action database for different averaging methods.

UCLA8 dynamic texture database (see Figure 3.5) as in [26] and apply the Bag-of-Systems approach using the proposed averaging method. We follow a similar approach as in [28] for the representation of the videos. Firstly, squared video patches of 25 frames of length are extracted from the original videos and fitted with an LDS of order $n = 3$. Then a dictionary of LDSs is learnt using the clustering method of Section 3.2. We tried different values (25, 50 and 75) for the total number of clusters (codebook size). Once the dictionaries were built, a histogram representation of each video sequence is computed based on the frequency of occurrence of the codebook's words (LDSs). Novel sequences are then labelled with a 1-NN classifier using the χ^2-distance between histograms. Again, for comparison purposes, we included in our experiments the MDS averaging method in [26]. Each method is run several times with different random initializations for the K-means clustering part. We used a

Figure 3.5: **UCLA8 dataset** – Dynamic textures in the UCLA8 database.

step-size of $\delta_C = 10^{-3}$ and $\delta_A = 10^{-5}$ for the gradient descend optimization with a maximum number of iterations equal to 150. For the K-means clustering we set the maximum number of iterations to 10.

The results shown in Table 3.4 correspond to the best achieved correct classification rate for different combinations of patch and codebook sizes. As it can be observed, the proposed technique based on averaging with respect to the Martin distance clearly outperforms the MDS averaging method of [26]. When compared to Align, Martin averaging does better in 5 out of 9 configurations. In general the performance of all three methods improves as the patch size is reduced as well as the size of the dictionary increases. This may be attributed to the fact that smaller patches capture better the local properties of the dynamic textures which, together with an increase in the codebook size, result in better descriptors in a Bag-of-Systems approach. From the results of the table, it can also be observed that Align does better

on the 20×20 patches while Martin generally does better for patch sizes of 60×60 and 30×30. It is also interesting to note that Align is quite sensitive if the size of the patch is not appropriate (differences in performance around 30%) while Martin appears to be more robust to the chosen size of the patches giving reasonable performance even with the largest patches (differences around 12%). This is a nice property to have in real applications where it is not known a priori what a suitable size of the patches is.

3.4 Chapter summary

In this chapter we have addressed the problem of averaging & clustering LDSs. We have considered the so-called extrinsic averaging problem using distances in the space of LDSs. In particular, we have presented a novel approach for averaging linear dynamical models based on the Martin distance. The method can be used to build meaningful representations with a reduced number of points. We have shown by means of numerical experiments that the method can be successfully applied to classification/recognition tasks in computer vision applications outperforming the method in [26] and providing a performance comparable to the state of the art. Also in this chapter we have provided a new method for computing the Align distance using the ADMM method that can be used for averaging w.r.t. that metric.

Patch Size	60×60×25	30×30×25	20×20×25
	25 clusters		
MDS Average	54.55	61.36	61.36
Align Average	47.73	**68.18**	**81.82**
Martin Average	**63.64**	65.91	75.00
	50 clusters		
MDS Average	59.09	61.36	61.36
Align Average	52.27	70.45	**84.09**
Martin Average	**63.64**	**79.55**	77.27
	75 clusters		
MDS Average	54.55	65.91	61.36
Align Average	59.09	70.45	**90.91**
Martin Average	**72.73**	**72.73**	84.09

Table 3.4: Bag-of-Systems dynamic texture classification accuracy on the UCLA8 dataset for different patch sizes and different number of clusters.

Chapter 4

DynamicSVM

In this chapter we will present a novel classification approach for time series data generated from an LDS. We will work directly with linear prediction functions in the ambient space of infinite dimensional time series. We will propose to use an SVM type of classifier in the infinite dimensional space of time series generated from an LDS. By restricting the set of possible solutions to a specific subset we will show how the problem can be reduced to a linear SVM problem in Euclidean space.

4.1 SVMs and LDSs

In Section 2.5.2.2 we presented the SVM classification approach in Euclidean space and showed how this approach can be generalized to any space where an inner product or a kernel can be defined. In this section we will consider the classification of time

series generated by an LDS using an SVM-like formulation in the Hilbert space of infinite dimensional time series generated by an LDS. Let us denote such space as $\mathcal{Z}$. Since the inner product between two elements in $\mathcal{Z}$ can be easily computed from the finite dimensional parameters of the LDSs, we can follow a regularized empirical risk minimization approach in order to compute a linear classifier in $\mathcal{Z}$. For the regularization term we will use the induced norm of the standard dot product in the ambient space $\mathcal{Z}$.

Recall that in a binary classification problem with m training feature-label pairs $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathcal{Z}$ are infinite dimensional time series and where $y_i \in \{-1, +1\}$, the formal formulation of the SVM optimization problem would be

$$\begin{aligned} \underset{\mathbf{w}\in\mathcal{Z},b,\{\xi_i\}}{\text{minimize}} \quad & \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle_{\mathcal{Z}} + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle_{\mathcal{Z}} + b) \geq 1 - \xi_i,\ i = 1, \ldots, m \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{4.1}$$

where we have used the standard dot product in $\mathcal{Z}$ and have replaced the Euclidean norm by the induced norm of the dot product (*i.e.*, $\|\cdot\|_{\mathcal{Z}}^2 = \langle \cdot, \cdot \rangle_{\mathcal{Z}}$).

From the representer theorem [23] it is well known that the optimal separating hyperplane is a linear combination of the data points

$$\mathbf{w}^\star = \sum_i \alpha_i \mathbf{x}_i. \tag{4.2}$$

Note that $\mathbf{w}^\star$ also corresponds to the output of an LDS. In order to see this, consider a situation where $\mathbf{w}^\star$ is only determined by the linear combination of two samples,

that is $\mathbf{w}^\star = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2$. It is straightforward to see that

$$\begin{aligned}\mathbf{w}^\star = \{\alpha_1\mathbf{x}_t^1 + \alpha_2\mathbf{x}_t^2\}_{t=0}^\infty = \{&\alpha_1\mathbf{C}_1\mathbf{x}_0^1 + \alpha_2\mathbf{C}_2\mathbf{x}_0^2,\\ &\alpha_1\mathbf{C}_1\mathbf{A}_1\mathbf{x}_0^1 + \alpha_2\mathbf{C}_2\mathbf{A}_2\mathbf{x}_0^2,\\ &\alpha_1\mathbf{C}_1\mathbf{A}_1^2\mathbf{x}_0^1 + \alpha_2\mathbf{C}_2\mathbf{A}_2^2\mathbf{x}_0^2, \ldots\},\end{aligned} \tag{4.3}$$

where we have assumed no driving or measurement noise. It is easy to realize that the sequence can be expressed as the output of the following LDS:

$$\bar{\mathbf{x}}_t = \begin{bmatrix}\mathbf{A}_1 & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_2\end{bmatrix}\bar{\mathbf{x}}_{t-1}, \text{ with } \bar{\mathbf{x}}_0 = \begin{bmatrix}\mathbf{x}_0^1\\ \mathbf{x}_0^2\end{bmatrix} \tag{4.4}$$

$$\mathbf{w}_t = \begin{bmatrix}\alpha_1\mathbf{C}_1 & \alpha_2\mathbf{C}_2\end{bmatrix}\bar{\mathbf{x}}_t. \tag{4.5}$$

Note that the optimal hyperplane is determined by an LDS of the same output dimension and possibly of higher order than the original points. According to (4.2) the optimal predictor is given by

$$f(\mathbf{x}) = \langle\mathbf{w}^\star, \mathbf{x}\rangle_{\mathcal{Z}} + b = \sum_{i=1}^m \alpha_i\langle\mathbf{x}_i, \mathbf{x}\rangle_{\mathcal{Z}} + b = \sum_{i=1}^m \alpha_i\kappa_{\mathcal{Z}}(\mathbf{x}_i, \mathbf{x}) + b, \tag{4.6}$$

where $\kappa_{\mathcal{Z}}(\cdot,\cdot)$ is a linear kernel in $\mathcal{Z}$. For example, consider two time series $\mathbf{x} = \{\mathbf{x}_t\}_{t=0}^\infty$ and $\mathbf{x}' = \{\mathbf{x}_t'\}_{t=0}^\infty$, both (generated from an LDS) belonging to $\mathcal{Z}$. Let us consider the inner product definition

$$\langle\mathbf{x}, \mathbf{x}'\rangle_{\mathcal{Z}} = \sum_{t=0}^\infty \lambda^t\mathbf{x}_t^\mathsf{T}\mathbf{x}_t', \tag{4.7}$$

with $\lambda \in (0, 1)$ is a forgetting factor that ensures the convergence of the sum in (4.7). Then we recover the Binet-Cauchy trace kernel between LDSs.

4.2 DynamicSVM

We have shown how the formulation of the SVM classification problem naturally relates to the concept of Binet-Cauchy kernels by using the inner product definition of (4.7). We propose in this section an alternative classifier by restricting the search space to a specific subset of $\mathcal{Z}$. From (4.2) it is clear that the optimal hyperplane is defined by a vector that is obtained as the linear combination of some elements in the training set. We can think of $\mathbf{w}^\star$ as a sort of "average" time series obtained from the training data points. Based on this observation we propose to use a linear predictor function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle_{\mathcal{Z}} + b$, where the vector (time series) $\mathbf{w}$ is generated from an LDS of the same order and output dimensions as the training data points. Let $\mathbf{w}$ be parameterized by $\bar{\mathcal{M}} = (\bar{\mathbf{A}}, \bar{\mathbf{C}}, \bar{\mathbf{s}}_0)$, where $\bar{\mathbf{A}}$ and $\bar{\mathbf{C}}$ are the parameters of an average LDS, while $\bar{\mathbf{s}}_0$ represent the initial conditions of the time series $\mathbf{w}$.

Consider now a time series $\mathbf{x}$ that is parametrized by $\mathcal{M} = (\mathbf{A}, \mathbf{C}, \mathbf{s}_0)$. If we neglect the noise terms in the state-space representation of (2.1) and (2.2), the time series $\mathbf{x}$ generated by the LDS model $\mathcal{M}$ (similarly for $\mathbf{w}$ and $\bar{\mathcal{M}}$) is given by

$$\mathbf{x} = \{\mathbf{z}_t\}_{t=0}^{\infty} = \{\mathbf{C}\mathbf{s}_0, \mathbf{C}\mathbf{A}\mathbf{s}_0, \mathbf{C}\mathbf{A}^2\mathbf{s}_0, \mathbf{C}\mathbf{A}^3\mathbf{s}_0, \ldots\}. \tag{4.8}$$

Keeping in mind (4.8), we can now write the dot product between $\mathbf{w}$ and $\mathbf{x}$ as

$$\langle \mathbf{w}, \mathbf{x} \rangle_{\mathcal{Z}} = \sum_{t=0}^{\infty} \lambda^t \mathbf{w}_t^\mathsf{T} \mathbf{z}_t = \sum_{t=0}^{\infty} \lambda^t \bar{\mathbf{s}}_0^\mathsf{T} (\bar{\mathbf{A}}^t)^\mathsf{T} \bar{\mathbf{C}}^\mathsf{T} \mathbf{C}\mathbf{A}^t \mathbf{s}_0 = \bar{\mathbf{s}}_0^\mathsf{T} \mathbf{P}\, \mathbf{s}_0, \tag{4.9}$$

where the matrix $\mathbf{P}$ corresponds to the infinite sum

$$\mathbf{P} = \sum_{t=0}^{\infty} \lambda^t (\bar{\mathbf{A}}^t)^\mathsf{T} \bar{\mathbf{C}}^\mathsf{T} \mathbf{C} \mathbf{A}^t. \tag{4.10}$$

Note that if the sum in (4.10) converges, $\mathbf{P}$ can be expressed as

$$\begin{aligned} \mathbf{P} &= \sum_{t=0}^{\infty} \lambda^t (\bar{\mathbf{A}}^t)^\mathsf{T} \bar{\mathbf{C}}^\mathsf{T} \mathbf{C} \mathbf{A}^t \\ &= \sum_{t=1}^{\infty} \lambda^t (\bar{\mathbf{A}}^t)^\mathsf{T} \bar{\mathbf{C}}^\mathsf{T} \mathbf{C} \mathbf{A}^t + \bar{\mathbf{C}}^\mathsf{T} \mathbf{C} \\ &= \lambda \bar{\mathbf{A}}^\mathsf{T} \left(\sum_{t=0}^{\infty} \lambda^t (\bar{\mathbf{A}}^t)^\mathsf{T} \bar{\mathbf{C}}^\mathsf{T} \mathbf{C} \mathbf{A}^t \right) \mathbf{A} + \bar{\mathbf{C}}^\mathsf{T} \mathbf{C} \\ &= \lambda \bar{\mathbf{A}}^\mathsf{T} \mathbf{P} \mathbf{A} + \bar{\mathbf{C}}^\mathsf{T} \mathbf{C}, \end{aligned} \tag{4.11}$$

and therefore, it can be efficiently computed as the solution to the Sylvester's equation $\mathbf{P} = \lambda \bar{\mathbf{A}}^\mathsf{T} \mathbf{P} \mathbf{A} + \bar{\mathbf{C}}^\mathsf{T} \mathbf{C}$. It is important to mention that, in order for the sum to converge, it is necessary that $\lambda \bar{\mathbf{A}} \otimes \mathbf{A}$ be stable. Note that this condition is automatically satisfied for stable LDS models.

Let us recover our binary classification problem with m training samples $\mathbf{x}_i$, $i = 1, \ldots, m$. Let $\mathbf{x}_i$ be time series generated from an LDS with parameters $\mathcal{M}_i = (\mathbf{A}_i, \mathbf{C}_i, \mathbf{s}_0^{(i)})$. We know from (4.9) that $\langle \mathbf{w}, \mathbf{x}_i \rangle_{\mathcal{Z}}$ can be equivalently computed as

$$\bar{\mathbf{s}}_0^\mathsf{T} \mathbf{P}_i \mathbf{s}_0^{(i)} = \bar{\mathbf{s}}_0^\mathsf{T} \bar{\mathbf{x}}_i, \tag{4.12}$$

$$\mathbf{P}_i = \lambda \bar{\mathbf{A}}^\mathsf{T} \mathbf{P}_i \mathbf{A}_i + \bar{\mathbf{C}}^\mathsf{T} \mathbf{C}_i. \tag{4.13}$$

Plugging (4.12) into problem (4.1) leads to

$$\begin{aligned} \underset{\mathbf{w}\in\mathcal{Z},b,\{\xi_i\}}{\text{minimize}} \quad & \frac{1}{2}\bar{\mathbf{s}}_0^{\mathsf{T}}\mathbf{S}\,\bar{\mathbf{s}}_0 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\bar{\mathbf{s}}_0^{\mathsf{T}}\bar{\mathbf{x}}_i + b) \geq 1-\xi_i,\ i=1,\ldots,m \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{4.14}$$

where $\langle \mathbf{w}, \mathbf{w}\rangle_{\mathcal{Z}} = \bar{\mathbf{s}}_0^{\mathsf{T}}\mathbf{S}\,\bar{\mathbf{s}}_0$ and where the matrix $\mathbf{S}$ is the solution to the Lyapunov equation

$$\mathbf{S} = \lambda\bar{\mathbf{A}}^{\mathsf{T}}\mathbf{S}\bar{\mathbf{A}} + \bar{\mathbf{C}}^{\mathsf{T}}\bar{\mathbf{C}}. \tag{4.15}$$

Note that $\mathbf{S}$ is symmetric and positive semi-definite, and it is positive definite for full-rank $\bar{\mathbf{C}}$.

It is easy to realize now that if we fix the $\bar{\mathbf{A}}$ and $\bar{\mathbf{C}}$ parameters of the separating hyperplane $\mathbf{w} \in \mathcal{Z}$, and optimize only over the initial conditions $\bar{\mathbf{s}}_0$, the problem reduces to a linear SVM problem in $\mathbb{R}^n$. To better illustrate this, assume that $\bar{\mathbf{C}}$ is of full-rank (therefore $\mathbf{S}$ is positive definite) and consider the change of variables

$$\tilde{\mathbf{w}} = \mathbf{S}^{1/2}\bar{\mathbf{s}}_0. \tag{4.16}$$

We can now rewrite problem (4.23) as

$$\begin{aligned} \underset{\tilde{\mathbf{w}}\in\mathbb{R}^n,\{\xi_i\},b}{\text{minimize}} \quad & \frac{1}{2}\left\|\tilde{\mathbf{w}}\right\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to} \quad & y_i(\tilde{\mathbf{w}}^{\mathsf{T}}\tilde{\mathbf{x}}_i + b) \geq 1-\xi_i,\ i=1,\ldots,m \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{aligned} \tag{4.17}$$

where

$$\tilde{\mathbf{x}}_i = \mathbf{S}^{-1/2}\bar{\mathbf{x}}_i = \mathbf{S}^{-1/2}\mathbf{P}_i\mathbf{s}_0^i. \tag{4.18}$$

Note that, since we know $\bar{\mathbf{A}}$ and $\bar{\mathbf{C}}$, we can compute the "projected" features $\tilde{\mathbf{x}}_i$ from the associated LDS model parameters $\mathcal{M}_i$ of the time series $\mathbf{x}_i$. In that regard, we are effectively mapping an infinite dimensional time series $\mathbf{x}_i$ parametrized by $\mathcal{M}_i = (\mathbf{A}_i, \mathbf{C}_i, \mathbf{s}_0^{(i)})$ to a finite dimensional space (*i.e.*, $\mathbb{R}^n$). This mapping allows the reduction of the problem to a linear SVM in $\mathbb{R}^n$. Therefore, we can interpret the DynamicSVM approach as a preprocessing step that reduces the dimensionality of the problem to Euclidean space where we run a linear SVM classifier.

4.2.1 Multiple representatives

In the previous section we have illustrated how to exploit the fact that the considered time series are generated from an LDS in order to compute a linear classifier in $\mathcal{Z}$. Further, we have seen that if we fix some of the parameters of the separating hyperplane $\mathbf{w} \in \mathcal{Z}$ the problem reduces to a linear SVM problem in Euclidean space. It is clear that for a binary classification problem it suffices to have one separating hyperplane. In order to reduce the problem to a linear SVM in Euclidean space, we could use one representative model for one of the two classes.

In principle, one could choose such representative as an average model $\bar{\mathcal{M}} = (\bar{\mathbf{A}}, \bar{\mathbf{C}})$ for any of the two classes. These average models could be obtained using the methods presented in Chapter 3. However, it is not clear which one of the two classes should be used. But in addition, one may also want to consider the possibility of using more than just one representative point per class. In fact, in the original

SVM problem (without fixing some of the parameters of the hyperplane) the optimal separating hyperplane is obtained by combining points coming from the two classes. This suggests that having several representative points could lead to an increase in the discriminative power of the classifier since we would be able to better capture the variability within each class. One important property of the DynamicSVM formulation that we propose is that it can easily handle multiple representative points per class. To better illustrate this idea let us note that the kernel implicitly used in problem (4.23) is given by

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tilde{\mathbf{x}}_i^\mathsf{T} \tilde{\mathbf{x}}_j = (\mathbf{s}_0^{(i)})^\mathsf{T} \mathbf{P}_i^T \mathbf{S}^{-1} \mathbf{P}_j \mathbf{s}_0^{(j)}. \tag{4.19}$$

Also note that using multiple representatives could be handled in a straightforward way by considering a new kernel that is a weighted combination of the different data-dependent kernels induced by each one of the representatives. More formally, consider again a binary classification problem where we extract r_1 and r_2 representative points from class 1 and class 2, respectively. Let the total number of representatives be $R = r_1 + r_2$, whose parameters are given by $\bar{\mathcal{M}}_r = (\bar{\mathbf{A}}_r, \bar{\mathbf{C}}_r)$, $r = 1, \ldots, R$. Consider then a kernel of the form

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{r=1}^{R} d_r (\mathbf{s}_0^{(i)})^\mathsf{T} \mathbf{P}_{ri}^T \mathbf{S}_r^{-1} \mathbf{P}_{rj} \mathbf{s}_0^{(j)} = \sum_{r=1}^{R} d_r \kappa_r(\mathbf{x}_i, \mathbf{x}_j), \tag{4.20}$$

where $d_r \in \mathbb{R}_+$ are positive weights and where

$$\mathbf{P}_{ri} = \lambda \bar{\mathbf{A}}_r^\mathsf{T} \mathbf{P}_{ri} \mathbf{A}_i + \bar{\mathbf{C}}_r^\mathsf{T} \mathbf{C}_i \tag{4.21}$$

$$\mathbf{S}_r = \lambda \bar{\mathbf{A}}_r^\mathsf{T} \mathbf{S}_r \bar{\mathbf{A}}_r + \bar{\mathbf{C}}_r^\mathsf{T} \bar{\mathbf{C}}_r. \tag{4.22}$$

4.2.2 Finding the weights

Effectively, the kernel definition in (4.20) implies that we are using a concatenation of the equivalent feature vectors (also of the time series). There is however, one question remaining and it is how to choose the kernel weights d_r. A simple alternative would be to give the same weight for each kernel component (*e.g.*, $d_r = 1/R$ for all $r = 1, \ldots, R$). Alternatively, one could try to simultaneously optimize the classifier parameters together with the weights d_r. The latter approach can be done using the Multiple Kernel Learning (MKL) framework [54]. There exist several approaches in the literature on how to solve the MKL problem. A possible approach is the one in [55] where and additional regularizer on the weights d_r is added to the objective of the primal SVM problem. It is easy to see that following our formulation, the problem to solve is

$$\begin{aligned} \underset{\mathbf{d},\tilde{\mathbf{w}},\{\xi_i\},b}{\text{minimize}} \quad & \frac{1}{2}\left\|\tilde{\mathbf{w}}\right\|^2 + C\sum_{i=1}^{m}\xi_i + \beta(\mathbf{d}) \\ \text{subject to} \quad & y_i(\tilde{\mathbf{w}}^{\mathsf{T}}\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i,\ i = 1, \ldots, m \\ & \boldsymbol{\xi} \geq \mathbf{0},\ d_r \geq 0,\ r = 1, \ldots, R \end{aligned} \tag{4.23}$$

where $\mathbf{d} = [d_1, \ldots, d_R]^{\mathsf{T}}$, $\beta(\cdot)$ is a regularizer (ℓ_1 or ℓ_2 norm in Euclidean space) and where the mapping $\phi : \mathcal{Z} \mapsto \mathbb{R}^{nR}$ is given by

$$\phi(\mathbf{x}_i) = \begin{bmatrix} \sqrt{d_1}\,\mathbf{S}_1^{-1/2}\mathbf{P}_{1i}\,\mathbf{s}_0^{(i)} \\ \vdots \\ \sqrt{d_R}\,\mathbf{S}_R^{-1/2}\mathbf{P}_{Ri}\,\mathbf{s}_0^{(i)} \end{bmatrix}. \tag{4.24}$$

The way in which problem (4.23) is solved is by alternating the updates of the classifier's parameters $(\tilde{\mathbf{w}}, b)$ and the kernel weights d_r. Note that fixing the weights, problem (4.23) reduces to a linear SVM problem that can be solved using any SVM solver such as `libsvm` [56]. Once the classifier has been learnt, a gradient update of the kernel weights can be performed as described in [55]. This alternating process is successively repeated until convergence.

To better illustrate the learning process, let us rewrite problem (4.23) as the following optimization problem

$$\begin{aligned} \underset{\mathbf{d}}{\text{minimize}} \quad & \beta(\mathbf{d}) + g(\mathbf{d}) \\ \text{subject to} \quad & \mathbf{d} \geq \mathbf{0}, \end{aligned} \tag{4.25}$$

where the function $g(\mathbf{d}) = \min_{\tilde{\mathbf{w}},b} \frac{1}{2} \|\tilde{\mathbf{w}}\|^2 + \sum_i \ell(y_i(\tilde{\mathbf{w}}^\mathsf{T}\phi(\mathbf{x}_i) + b))$. Since $g(\mathbf{d})$ corresponds to the solution of a convex optimization problem, it can be equivalently computed from the dual problem (*i.e.*, strong duality holds provided some qualification constraints). We can therefore express $g(\mathbf{d})$ as

$$\begin{aligned} g(\mathbf{d}) = \quad \underset{\alpha}{\text{maximize}} \quad & \boldsymbol{\alpha}^\mathsf{T}\mathbf{1} - \frac{1}{2}\boldsymbol{\alpha}^\mathsf{T}\left(\sum_r d_r \mathbf{K}_r \odot \mathbf{Y}\right)\boldsymbol{\alpha} \\ \text{subject to} \quad & \boldsymbol{\alpha}^\mathsf{T}\mathbf{y} = 0,\, \mathbf{0} \leq \boldsymbol{\alpha} \leq C\mathbf{1}, \end{aligned} \tag{4.26}$$

where $\mathbf{K}_r$ is the kernel matrix corresponding to the rth representative and whose entries are given by $[\mathbf{K}_r]_{ij} = \kappa_r(\mathbf{x}_i, \mathbf{x}_j)$. The fact that $g(\mathbf{d})$ is the solution of a strictly convex optimization problem makes it a differentiable function (see [55] and references therein), and its derivative can be computed from the derivative of the dual objective

particularized to the optimal solution, that is

$$\frac{\partial g}{\partial d_r} = -\frac{1}{2}\boldsymbol{\alpha}^{\star\mathsf{T}}(\mathbf{K}_r \odot \mathbf{Y})\boldsymbol{\alpha}^{\star}. \tag{4.27}$$

Therefore, if $\beta(\mathbf{d})$ is differentiable, we can now easily compute the derivative of the cost function in (4.25) in order to perform a gradient update of the kernel weights. In particular, consider that $\beta(\cdot) = \|\cdot\|_2^2$, then the weights at the kth iteration can be updated as

$$\begin{aligned} d_r^{(k)} &= d_r^{(k-1)} - \delta_d \left(\frac{\partial \beta}{\partial d_r} + \frac{\partial g}{\partial d_r} \right) \\ &= d_r^{(k-1)} - \delta_d \left(2d_r - \frac{1}{2}\boldsymbol{\alpha}^{\star\mathsf{T}}(\mathbf{K}_r \odot \mathbf{Y})\boldsymbol{\alpha}^{\star} \right), \end{aligned} \tag{4.28}$$

where δ_d is the step size for the update. After updating the weights, the parameters of the SVM classifier are computed again, and the process is repeated until convergence.

4.3 Chapter summary

In this chapter we have presented a novel classification approach for time series of data generated from an LDS. We call this approach DynamicSVM. Starting from the formulation problem in the Hilbert space of infinite dimensional time series, and fixing some of the parameters of the classifier to the average value, we can optimize over the initial conditions. The approach offers a clear computational advantage as compared to the Binet-Cauchy kernels, since only the kernel to a few representative points needs to be computed. The approach can also be interpreted as a mapping from

the space of parameters (or from the time series) to an Euclidean space in $\mathbb{R}^n$. In the experiments chapter we will show how our method not only reduces the complexity of the classifier but also achieves a comparable performance in terms of classification accuracy.

Chapter 5

Surgical Gesture Classification

In this chapter we will investigate some of the techniques presented in [11, 13] (also covered in the previous chapters) to the problem of surgical gesture classification from video. More specifically, we will use linear dynamical systems (LDSs) to model the time series of the raw pixel intensities extracted from each video clip and the kinematic measurements provided by the *da Vinci Surgical System* of Intuitive Surgical[1]). We will use the metrics between the parameters of the LDSs presented in Chapter 2 to train classifiers for each gesture. In addition, we will also investigate the performance of the DynamicSVM approach discussed in Chapter 4 where the different LDS data points are mapped to $\mathbb{R}^n$ using a set of pre-computed representative points or exemplars. For the computation of these representative points we will use the techniques presented in Chapter 3 using the extrinsic mean of a set of LDS points

[1]Intuitive Surgical Inc., Sunnyvale, CA

with respect to some metric (*e.g.*, Martin and Align distances).

5.1 State of the art

Previous work on skill evaluation in RMIS mainly exploited kinematic data recorded by the robot. Many works used global measurements of the task, such as time to completion [57, 58], speed and number of hand movements [57], distance travelled [58], and force and torque signatures [58–60]. These methods are generally easy to implement. However, they perform a global assessment neglecting the fact that a surgical task is composed of many different gestures. Such global approaches have two main drawbacks. First, they use a single model for a complex task as a whole, while the decomposition of a task into atomic gestures allows for the use of a simpler model for each gesture. Second, they assume that a trainee is either skilled or unskilled at all gestures. In practice, different gestures have different levels of complexity, and one would expect a trainee to learn quickly how to perform simple gestures, and to require more training to perform complex ones.

To address these drawbacks, several works (see, *e.g.*, [1,61–63]) have considered the problem of decomposing a surgical task into atomic gestures, usually called *surgemes*. Such a decomposition not only addresses the drawbacks of global approaches, but also has the advantage of exploiting the set of rules that govern how different surgemes are related to each other. In other words, it allows one to describe a surgical task

using a grammar that, for each task, describes which transitions between gestures are allowed. One can leverage this grammar to help the recognition of a surgeme, *e.g.*, by exploiting the fact that the set of surgemes that follows an already labelled surgeme is smaller. One can also use such a grammar as an additional measure of assessment. For instance, each gesture in isolation could be executed perfectly, but the sequence of gestures may not make sense for the given task (*e.g.*, inserting the needle before grabbing the needle). Given the many similarities with the structure of natural languages, this approach to surgical skill assessment is also known as *the language of surgery*. This approach proceeds in three steps: task segmentation, gesture recognition, and skill evaluation (assessment of the quality of the execution and the feasibility of the sequence of gestures). Since this thesis deals with the recognition phase, we will limit the discussion of previous work to those related to surgical gesture recognition.

Most of the prior work on surgical gesture recognition (see, *e.g.*, [64–66]) uses HMMs to analyze kinematic data stored by the robot. All these approaches model each surgeme as one or more states of an HMM. The main difference is in how these approaches model the observations within each surgeme. For example, [65] vector-quantize the observations into discrete symbols, [67] use a Gaussian model combined with linear discriminant analysis (LDA), [66] assumes that the observations are generated from a lower-dimensional latent space using Factor Analyzed HMMs (FA-HMMs) and Switched Linear Dynamical Systems (SLDSs), [68] use a Gaussian

mixture model (GMM), and [69] model the observations as a linear combination of atomic motions with sparse coefficients. All of these models have significantly improved surgical gesture classification over a standard HMM.

In addition to kinematic measurements, RMIS systems are also typically equipped with cameras that record the entire procedure. Early work on video data analysis, such as [6], focus on recognizing the (coarse) phases of a surgery by also observing surgeons and nurses in the operating room. In [70] an automatic feature extraction mechanism from the videos is proposed based on genetic programming. They use the extracted features to classify the (coarse) phases of a surgery but the average recognition accuracy is around 50%. The work in [71] and [12] propose to recognize the different coarse phases of a surgery (e.g. CO2 inflation, abdominal suturing, etc.) using laparoscopic videos. For example, the work in [12] uses binary signals that indicate the presence or not of a set of tools in the operating room. Using those signatures they use Dynamic Time Warping (DTW) and HMMs in order to classify new sequences. Also in [10] an application-dependent framework for the recognition of high-level surgical phases is proposed. The method applies DTW and HMMs on top of a set of SVM classifiers. In a recent contribution [72], the same authors extend their approach in order to provide additional granularity by further decomposing each of the surgical phases into basic actions. The authors combine then the approach in [10] together with the detection of tools and organs in order to determine the surgical action being performed. A recognition accuracy of around 64% on a frame by frame

basis can be achieved with the proposed technique when applied to cataract surgeries. A limitation of the methodology is that it is application dependent and needs to be tuned to target a specific type of intervention. It would be desirable to have a general methodology that can be abstracted from the surgery at hand and that is based on the recognition of elementary actions that can be used to describe almost any surgery.

One attempt to automatic classification of skill and surgical gestures (rather than coarse phases) from video is that of [9]. [9] uses different flavors of HMM where the observation is the histogram of optical flow concatenated with the mean flow computed in spatially separated regions of the image. The conclusion of this study is that kinematic-based approaches are generally more accurate than vision-based methods. However, the recent work in [11, 13] shows that video-based techniques can perform equally-well as kinematic-based approaches. They propose the use of LDSs and features extracted from the videos to build classifiers of the surgical gestures. Further, the work also suggests the combination of different kinds of data using Multiple Kernel Learning (MKL) as a possible way of boosting performance.

In the experimental section, we will study the performance of the techniques proposed in the previous chapters based on LDSs to the problem of surgical gesture recognition analyzing their advantages and drawbacks as compared to the state-of-the-art.

5.2 Dataset

In order to test our algorithms, we will use the surgical dataset presented in [1]. The dataset is a collection of surgical trials on three different tasks: suturing (SU, 39 trials), needle passing (NP, 26 trials) and knot tying (KT, 36 trials). Each task is performed by 8 trainees with different skill levels (expert, intermediate and novice). Typically each user performed around 3 to 5 trials for each task. Each trial lasts, on average, 2 minutes and both kinematic and video data are recorded at a rate of 30 frames per second. Kinematic data consists of 78 motion variables (positions, rotation angles, and velocities of the master/patient side manipulators), whereas the videos are converted into JPEG images of size 320×240 for each frame.

The data was manually segmented based on the surgeme's definition of [1]. Specifically, the vocabulary of possible atomic actions consisted of 15 surgemes (or gestures): 1) reaching for needle with right hand, 2) positioning needle, 3) pushing needle through tissue, 4) transferring needle from left to right, 5) moving to center with needle in grip, 6) pulling suture with left hand, 7) pulling suture with right hand, 8) orienting needle, 9) using right hand to help tighten suture, 10) loosening more suture, 11) dropping suture at end and moving to end points, 12) reaching for needle with left hand, 13) making 'C' loop around right hand, 14) right hand reaches for suture and 15) both hands pull. Note that, although there are a total of 15 surgemes, not all of them appear in a given task. For example, suturing and needle passing typically involves 10 of these 15 surgemes, while knot tying involves only 6 surgemes.

Surgeme	Description	SU	NP	KT
G1	Reach needle right hand	29	30	19
G2	Positioning needle	166	113	–
G3	Push needle through tissue	164	106	–
G4	Transfer needle left to right	119	81	–
G5	Move to center with needle	37	30	–
G6	Pull suture with left hand	163	108	–
G7	Pull suture with right hand	–	–	–
G8	Orienting needle	48	27	–
G9	Tighten suture right hand	24	1	–
G10	Loosening more suture	4	1	–
G11	Drop suture, move to end points	39	23	36
G12	Reach needle left hand	–	–	70
G13	C loop around right hand	–	–	75
G14	Right hand reaches suture	–	–	98
G15	Pull with both hands	–	–	73
Total		793	519	371

Table 5.1: Definition of the different gestures or surgemes in [1] and the total number of occurrences within each task (suturing, needle passing, and knot tying).

A detailed description about the total number of gestures in the dataset for each of the tasks is given in Table 5.1. In terms of length, a typical suturing trial is a collection of about 20 video clips, while a needle passing has an average of 13 video clips, and knot tying is composed of about 9 video clips.

5.3 Experimental setup

In order to compare the accuracy of the surgeme recognition task using kinematic versus video data, we have created two different test setups following [11, 13, 66, 69]. The first setup is the *leave-one-super-trial-out* (LOSO), where we leave one trial for each one of the users out for testing. For example, we leave the first trial of every user for testing and use the remaining trials as training data. The second setup is the *leave-one-user-out* (LOUO), where we leave all the trials from one user out for testing. This clearly corresponds to a more challenging scenario since, contrary to the LOSO setup, we are testing on a novel user, not previously seen in training. For each task we performed a training and a test phase using only the surgemes that appeared in that task. To be more precise, we will compare three classification approaches: nearest-neighbor, kernel SVMs and dynamicSVM. For the nearest-neighbor approach (1-NN) we will use the distances (*e.g.*, Martin, Frobenius and Align) and kernels (BC-Det and BC-Trace) described in Chapter 2. For the case of the kernels we will use the kernel to distance formula, that is, given a kernel $\kappa(\cdot, \cdot)$ we can define a distance

as

$$d_\kappa(\mathcal{M}, \mathcal{M}') = \kappa(\mathcal{M}, \mathcal{M}) + \kappa(\mathcal{M}', \mathcal{M}') - 2\kappa(\mathcal{M}, \mathcal{M}'), \tag{5.1}$$

where $\mathcal{M}$ and $\mathcal{M}'$ are two LDSs. For the kernel SVM approach we will use RBF kernels on the LDS distances as well as the Binet-Cauchy kernels. In the case of RBF kernels, we will combine ten different kernels with $\gamma = 1, \ldots, 10$ using MKL. The dynamicSVM approach will be evaluated for two different numbers of clusters per gesture (*e.g.*, one and three clusters). In order to compute the representatives we will employ the averaging methods described in Chapter 3 (with respect to Martin and Align distances). For the SVM classifiers we will evaluate the performance for different values of the penalty parameter C (*e.g.*, $C \in \{2^{-6}, 2^{-5}, \ldots, 2^3\}$). Finally, the BC-Trace kernel will be evaluated for two different values of the η parameter, one that takes into account the driving noise matrices ($\eta = 0.5$) and another one that only uses the $\mathbf{A}$, $\mathbf{C}$ and the initial conditions ($\eta = 1$).

5.4 Experimental results

In this section we evaluate the performance of the methods described in Chapters 2, 3 and 4 for the problem of surgical gesture classification. We present the classification accuracy separately for each of the tasks, namely suturing, knot tying and needle passing, and for each one of the two setups (LOSO and LOUO) described in the previous section.

In Tables 5.2 to 5.10 we report the best average classification rates obtained in the simulations (*i.e.*, numbers within the same table may correspond to different values of the penalty C). We report two metrics in the tables. The metric labelled as "Macro-avg" corresponds to the average correct classification of the gestures without differentiating their class. The second metric, labelled as "Micro-avg" corresponds to an average computed over the per-class averages. More formally, in a K-class classification problem, denote N_k the number of test samples of class k and let $N = \sum_k N_k$ be the total number of test samples. Then, the macro- and micro-averages are defined as

$$\mu_{macro} = \frac{\sum_{i=1}^{N} \mathbb{I}\left[\hat{c}_i = c_i\right]}{\sum_{k=1}^{K} N_k} \tag{5.2}$$

$$\mu_{micro} = \frac{1}{K} \sum_{k=1}^{K} \frac{1}{N_k} \sum_{i=1}^{N} \mathbb{I}\left[\hat{c}_i = c_i \,\&\, c_i = k\right], \tag{5.3}$$

where $\hat{c}_i, c_i \in \{1, \ldots, K\}$ are the estimated and the true class of sample i, respectively, $i = 1, \ldots, N$, and where $\mathbb{I}[\cdot]$ is the indicator function (1 if the argument is true, 0 otherwise). For example, suppose that we had a two-class dataset with 9 gestures in the test set, 5 gestures of class 1 and 4 gestures of class 2. Suppose that we correctly classify 4 out of 5 for class 1 and all of them for class 2 (4 out of 4). Then the macro- and micro-averages will be computed as

$$\mu_{macro} = \frac{4+4}{9} = \frac{8}{9} = 0.89,$$
$$\mu_{micro} = \frac{4/5 + 4/4}{2} = \frac{9}{10} = 0.90.$$

From Tables 5.2 to 5.10, we can extract several conclusions regarding the perfor-

mance of the considered approaches. For instance, it can be observed that, in general, the best results correspond to the combination of several RBF kernels with different γ values using MKL. Note that, in the tables with the title KernelSVM, and for the metrics based on subspace angles, we are actually combining several RBF kernels with different values of γ (*i.e.*, from 1 to 10). Therefore, it is not surprising that those metrics typically outperform the Binet-Cauchy kernel counterparts. To be fair when comparing metrics based on subspace angles with metrics based on the Binet-Cauchy kernels, one should look at the performance of the 1-NN classifiers, since in that case, we are directly using the distances between the models without any further combination.

Looking at the numbers coming from the 1-NN classifier, one realizes that, indeed, the BC kernels provide comparable performance to metrics based on the subspace angles. Furthermore, in the more challenging LOUO setup, the BC kernels typically provide the best results, showing more robustness in their accuracy against novel data samples. It is also interesting to mention that the BC determinant kernel seems to be performing reasonably well using nearest neighbors and that a significant degradation in the classification performance occurs when using SVMs. When comparing the performance of the BC kernels, clearly the trace kernel provides the best results. More interestingly, the kernel with $\eta = 1$ almost always outperforms its balanced counterpart (*i.e.*, $\eta = 0.5$). This result is telling us that the inclusion of the noise coloring matrices $\mathbf{B}$ when comparing two LDSs might not be beneficial for the purpose

of classification, and that its consideration might increase the uncertainty among the models. When comparing the Martin, Frobenius and Align distances, we observe that for kinematic data Frobenius seems to be the most appropriate metric whereas so it is Align for the case of video data.

From the analysis of the numbers coming from the DynamicSVM approach, we can also draw some conclusions. First of all, it is worth to remember, that the DynamicSVM approach should be directly compared with the Binet-Cauchy kernels, and more particularly, to the trace kernel with $\eta = 1$, since in the DynamicSVM approach we are also neglecting the driving noise coloring matrices $\mathbf{B}$. Overall, there is a degradation in terms of accuracy of the DynamicSVM when compared to the Binet-Cauchy trace kernel. This degradation is due to the fact that the classifiers are built. Recall that the DynamicSVM uses a few exemplars per class rather than the entire set of training points at the SVM classification step. Also note that, in general, the performance of the DynamicSVM approach increases when the number of representatives is larger. This effect is more pronounced in the case of video data due to the high dimensionality of the frame (320×240), while with the kinematic data the improvement is more moderate. These results suggest that the number of representative points per class should be larger when the output dimension is bigger. However, it is important to mention that the DynamicSVM presents a significant advantage in the computational complexity as compared to the BC kernels since the Sylvester equation needs to be solved between each point and the set of representatives and

not between every training-test pair. These savings in the computational complexity come at the expense of a small degradation in terms of classification accuracy.

Regarding the question of whether video or kinematic data is more discriminative, the obtained results support the conclusions drawn in [11,13] where it was concluded that both type of data can be equally discriminative. If we look at the KernelSVM or 1-NN results, we observe similar performances for both video and kinematic data with the only exception of the Needle Passing task where we observe a significant benefit of kinematic over video. In the DynamicSVM approach there is a clear benefit of using kinematic over video but this might be attributed to the fact that we are using too few representatives given the high dimensionality of the video signals.

Finally, we would like to comment on the difference between the macro- and micro-averages reported in Tables 5.2 to 5.10. The fact that the macro-averages are typically higher than their micro- counterparts is due to the reduced number of samples of some of the gestures. For example, if we look at Table 5.1, we observe that G10 and G9 only appears once in the Needle Passing task (also in the Suturing task G10 appears only 4 times). This means that those gestures are never going to be correctly classified since they will either belong to the test or to the training set. When averaging the per-class averages (*i.e.*, micro-average) this will cause bias in the classification accuracy. It is clear then that this imbalance between the number of gestures per class makes the two metrics deviate from each other.

5.5 Chapter summary

In this chapter we have analyzed the performance of the classification methods presented in the thesis on the problem of surgical gesture classification. Assuming known segmentation of the data, we have used LDSs to model the evolution of both kinematic and video data time series within each segment. Overall, our results are in agreement with the state of the art [11, 13], with classification accuracies around 90% for the Suturing and Knot Tying tasks, and around the 85% for the Needle Passing task. Also in this chapter we have analyzed the performance of the DynamicSVM approach and compared it to the Binet-Cauchy kernels. Our simulations also show that, in a typical surgical training setup, video data can be as discriminative as kinematic data for the purpose of surgical gesture classification. Overall, the combination of several RBF distance-based kernels using LDSs outperforms the use of Binet-Cauchy kernels and the DynamicSVM approach. However, the latter approach is computationally more attractive than the former two since, at testing, it only requires the computation of the projected features with respect to the set of representatives. This is a considerably reduction in the computational complexity since we are no longer required to compute the set of all pairwise distances/kernels at testing. However, this computational advantage of the DynamicSVM with respect to the Binet-Cauchy kernels comes at the expense of some degradation in the classification accuracy.

1-NN – Suturing Task

Metric	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO	*Kinematic*		*Video*	
Martin	80.32±5.29	64.34±9.58	77.37±5.79	67.43±7.50
Frobenius	83.85±6.26	70.91±10.63	79.78±7.66	71.21±9.34
Align	74.44±5.67	58.94±8.48	81.78±7.61	73.78±9.03
BC-Det	80.79±4.93	66.05±6.06	82.04±5.76	75.64±8.26
BC-Trace $\eta = \frac{1}{2}$	82.88±3.22	66.33±3.85	81.51±7.37	69.24±8.28
BC-Trace $\eta = 1$	85.28±3.87	71.64±5.10	81.81±7.82	70.16±8.79
LOUO	*Kinematic*		*Video*	
Martin	64.93±9.45	49.12±8.37	57.89±7.53	48.04±7.76
Frobenius	65.92±8.91	50.57±10.39	56.12±5.77	47.26±6.40
Align	59.94±6.22	45.31±8.39	57.72±7.58	49.55±10.40
BC-Det	65.21±8.15	51.97±11.90	61.31±10.24	53.30±9.82
BC-Trace $\eta = \frac{1}{2}$	74.05±7.55	57.93±8.75	63.39±6.52	49.23±8.84
BC-Trace $\eta = 1$	78.05±8.84	62.85±11.37	63.39±6.81	50.38±9.36

Table 5.2: Average classification rates for the Suturing task using a 1-NN classifier.

KernelSVM – Suturing Task

Metric	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO	*Kinematic*		*Video*	
Martin	80.92±3.71	63.56±6.15	81.12±6.25	65.56±8.35
Frobenius	90.67±3.85	81.58±8.60	82.95±4.99	69.89±6.20
Align	83.20±3.71	64.97±5.39	90.73±4.53	80.65±6.94
BC-Det	39.16±7.09	20.14±3.02	20.38±0.86	10.74±0.69
BC-Trace $\eta = \frac{1}{2}$	82.59±2.71	66.08±3.09	82.80±4.88	67.00±7.43
BC-Trace $\eta = 1$	85.63±2.84	71.00±4.49	83.08±5.47	67.56±8.56
LOUO	*Kinematic*		*Video*	
Martin	72.82±9.37	58.78±10.67	71.82±8.18	57.68±9.35
Frobenius	81.57±10.23	68.50±9.11	72.18±7.58	60.56±6.92
Align	72.25±12.04	52.79±9.28	79.57±8.81	65.78±11.18
BC-Det	29.29±5.59	16.41±3.37	20.13±1.44	11.49±0.91
BC-Trace $\eta = \frac{1}{2}$	78.44±8.63	61.94±10.02	70.27±8.28	54.27±8.44
BC-Trace $\eta = 1$	80.26±8.23	63.47±10.42	70.49±8.19	54.99±8.91

Table 5.3: Average classification rates for the Suturing task using RBF and BC kernels on LDSs.

DynamicSVM – Suturing Task

Metric	Clusters	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO		*Kinematic*		*Video*	
Martin	01	81.01±5.14	65.11±4.91	61.30±6.27	49.55±6.09
Martin	03	82.28±3.68	65.76±4.28	73.05±6.44	61.68±8.11
Align	01	80.92±3.01	64.68±2.11	56.53±4.16	45.79±6.61
Align	03	81.32±3.67	64.88±4.93	72.35±3.53	60.28±3.59
LOUO		*Kinematic*		*Video*	
Martin	01	73.67±8.89	59.29±12.07	51.67±6.58	40.56±8.88
Martin	03	75.14±8.11	56.58±8.99	60.65±10.09	46.63±9.34
Align	01	76.07±9.48	61.30±11.37	48.29±13.01	37.16±10.15
Align	03	76.39±9.80	62.19±13.59	60.98±8.32	47.95±9.97

Table 5.4: Average classification rates for the Suturing task using the DynamicSVM approach.

1-NN – Knot Tying Task

Metric	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO	*Kinematic*		*Video*	
Martin	74.23±4.72	70.03±7.41	63.54±8.05	65.30±9.69
Frobenius	83.02±1.50	75.69±4.03	72.24±3.20	74.43±2.74
Align	68.91±3.99	63.90±3.61	75.43±7.80	75.09±10.59
BC-Det	79.38±6.29	72.07±7.97	67.19±6.46	67.34±8.50
BC-Trace $\eta = \frac{1}{2}$	77.77±5.92	73.27±7.21	72.90±5.05	76.36±4.45
BC-Trace $\eta = 1$	81.62±3.45	76.97±5.14	73.77±6.19	76.96±5.17
LOUO	*Kinematic*		*Video*	
Martin	62.45±3.50	59.76±5.41	55.21±10.74	56.58±10.40
Frobenius	65.16±12.94	59.41±12.56	59.19±10.62	62.44±13.09
Align	56.93±10.67	52.61±11.83	58.43±11.00	58.90±13.55
BC-Det	63.11±7.47	58.30±6.68	58.31±8.75	58.96±7.35
BC-Trace $\eta = \frac{1}{2}$	72.01±5.17	66.30±6.12	65.70±12.11	66.96±11.43
BC-Trace $\eta = 1$	68.08±10.17	61.94±8.13	65.92±11.27	67.50±10.92

Table 5.5: Average classification rates for the Knot Tying task using a 1-NN classifier.

Kernel SVM – Knot Tying Task

Metric	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO	*Kinematic*		*Video*	
Martin	78.79±2.61	68.19±1.55	80.57±2.12	82.35±3.33
Frobenius	85.95±3.13	76.20±2.75	81.07±3.17	80.68±1.43
Align	81.67±4.07	71.42±4.44	89.41±1.90	87.79±2.64
BC-Det	26.94±0.96	17.13±0.64	26.39±0.69	16.67±0.00
BC-Trace $\eta = \frac{1}{2}$	79.59±4.06	72.26±4.55	78.27±4.87	77.54±4.84
BC-Trace $\eta = 1$	86.32±3.07	81.99±3.83	78.83±3.82	78.59±3.69
LOUO	*Kinematic*		*Video*	
Martin	74.96±5.19	68.75±6.38	72.10±12.30	70.61±12.72
Frobenius	82.94±8.64	75.63±10.26	74.73±10.28	70.90±10.02
Align	77.56±7.34	70.45±8.94	82.30±7.31	79.14±9.45
BC-Det	26.60±3.29	17.50±1.54	26.60±3.29	17.50±1.54
BC-Trace $\eta = \frac{1}{2}$	74.02±8.88	66.90±9.06	73.43±7.54	73.02±6.99
BC-Trace $\eta = 1$	75.11±10.40	67.16±10.77	72.13±8.40	71.58±7.57

Table 5.6: Average classification rates for the Knot Tying task using RBF and BC kernels on LDSs.

DynamicSVM – Knot Tying Task

Metric	Clusters	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO		*Kinematic*		*Video*	
Martin	01	79.82±1.65	72.94±1.97	65.62±4.52	66.44±3.50
Martin	03	79.90±3.56	73.01±3.87	67.39±7.05	69.35±4.92
Align	01	76.65±4.35	69.90±3.66	57.44±7.54	59.79±9.23
Align	03	79.30±2.82	73.97±1.22	64.85±8.63	64.16±8.34
LOUO		*Kinematic*		*Video*	
Martin	01	67.75±8.43	62.18±9.03	61.33±6.30	61.00±8.10
Martin	03	71.26±12.45	64.85±12.48	64.82±6.13	65.36±8.65
Align	01	70.57±7.97	63.44±9.60	56.55±9.13	57.80±7.37
Align	03	74.29±7.78	67.61±9.76	62.23±9.16	62.45±10.75

Table 5.7: Average classification rates for the Knot Tying task using the DynamicSVM approach.

1-NN – Needle Passing Task

Metric	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO	*Kinematic*		*Video*	
Martin	64.32±5.05	54.08±4.69	54.27±6.52	52.86±8.31
Frobenius	69.32±6.99	61.20±7.08	52.81±6.98	50.06±10.00
Align	61.60±6.55	52.16±7.81	56.81±5.95	53.38±7.25
BC-Det	66.04±5.35	55.80±6.89	50.77±8.83	51.86±9.48
BC-Trace $\eta = \frac{1}{2}$	68.62±5.49	57.94±4.31	54.52±5.94	53.11±7.69
BC-Trace $\eta = 1$	71.16±6.78	63.12±7.49	54.62±6.36	53.20±7.68
LOUO	*Kinematic*		*Video*	
Martin	57.59±6.58	48.88±8.23	43.94±4.61	43.78±6.40
Frobenius	60.67±8.92	52.00±8.54	37.10±8.14	37.88±6.48
Align	52.60±7.30	40.55±6.85	43.39±10.50	44.24±8.90
BC-Det	56.91±8.68	45.70±7.82	32.66±6.75	39.00±5.85
BC-Trace $\eta = \frac{1}{2}$	60.60±7.28	50.66±7.49	42.28±6.51	46.14±11.42
BC-Trace $\eta = 1$	62.43±9.12	54.78±9.19	43.19±7.01	46.79±11.91

Table 5.8: Average classification rates for the Needle Passing task using a 1-NN classifier.

Kernel SVM – Needle Passing Task

Metric	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO	*Kinematic*		*Video*	
Martin	68.51±4.48	50.66±2.40	63.26±4.66	50.60±7.65
Frobenius	80.23±6.52	69.24±6.83	60.36±3.31	52.88±5.35
Align	74.95±4.98	58.89±4.52	67.49±4.41	57.25±6.42
BC-Det	22.43±1.22	12.83±0.30	21.91±1.03	12.50±0.00
BC-Trace $\eta = \frac{1}{2}$	67.87±4.88	51.76±3.64	59.15±6.34	48.50±7.05
BC-Trace $\eta = 1$	70.86±6.73	59.32±7.25	58.54±6.25	48.03±6.67
LOUO	*Kinematic*		*Video*	
Martin	64.70±10.13	50.33±8.81	56.41±6.39	45.43±8.95
Frobenius	73.06±10.17	60.78±9.81	49.14±8.06	37.43±8.12
Align	69.35±8.63	53.85±7.73	56.54±7.51	46.37±8.88
Det	54.86±13.00	37.50±11.46	22.83±1.48	13.62±1.60
BC-Det	21.94±0.97	13.10±1.57	21.94±0.97	13.10±1.57
BC-Trace $\eta = \frac{1}{2}$	60.95±9.56	47.11±6.53	50.76±11.20	43.64±9.67
BC-Trace $\eta = 1$	62.93±8.61	51.41±7.29	50.15±10.87	42.97±9.62

Table 5.9: Average classification rates for the Needle Passing task using RBF and BC kernels on LDSs.

DynamicSVM – Needle Passing Task

Metric	Clusters	Macro-avg	Micro-avg	Macro-avg	Micro-avg
LOSO		*Kinematic*		*Video*	
Martin	01	60.94±7.83	45.28±5.63	44.27±1.21	40.03±3.81
Martin	03	62.78±9.31	47.25±7.56	49.46±5.16	48.17±4.56
Align	01	63.32±7.88	50.31±7.28	42.74±7.91	37.85±7.01
Align	03	65.43±6.58	49.89±4.34	46.54±2.42	42.50±1.66
LOUO		*Kinematic*		*Video*	
Martin	01	52.85±16.94	40.37±9.58	40.14±8.77	42.52±8.74
Martin	03	55.41±11.29	43.49±8.77	42.11±8.00	44.84±9.41
Align	01	57.31±17.03	41.99±12.74	32.30±7.34	33.29±11.71
Align	03	59.76±7.78	48.67±6.20	37.74±10.37	39.81±12.21

Table 5.10: Average classification rates for the Needle Passing task using the DynamicSVM approach.

Chapter 6

Conclusions

In this thesis we have proposed the use of linear dynamical systems as discriminative models for the purpose of surgical gesture classification. These models allow the use of different metrics and a large variety of kernels in the space of LDSs to perform the classification task and their utility has been validated through experiments in both synthetic and real data. One of the advantages of using LDSs is that they can be used for both video and kinematic data. We have validated by means of simulations that the both types of data can be equally discriminative in a typical surgical training-test setup. Additionally, the present work has made several contributions to the problem of statistical analysis using LDSs. This contributions are general and can thus be applied to different problems related to the analysis of time series. We have proposed a new averaging method based on the minimization of the extrinsic mean with respect to the Martin distance. The method outperforms existing approaches

based on approximate averaging with the Martin distance and has comparable performance to the state of the art. Another advantage of our formulation is that it is able to generate novel LDSs. These averaging method can be used for clustering purposes (*e.g.*, extraction of representative points) and for building meaningful representations using the bag of dynamical systems approach. We have also proposed a new method for the computation of the Align distance using the ADMM method. This approach is particularly attractive since an iterative procedure with closed-form updates is possible. The method provides better averages than the state of the art but it suffers from a higher complexity. However, the algorithm can still be improved for further efficiency. Finally, we have proposed a new classification method for time series generated from an LDS. The approach relies on the idea of working directly in the space of infinite dimensional sequences generated by an LDS and defining a linear classifier in such space. By doing this, one ends up with the definition of the Binet-Cauchy kernels, that require the solution of a Sylvester equation for each entry of the kernel matrix. If we fix some of the parameters of the classifier to some restricted subset of representative points and optimize over the initial conditions, then the problem can be cast as a linear SVM problem in Euclidean space. Effectively, we are defining a map from the space of parameters to an Euclidean space. This map requires the solution of a Sylvester equation between any sample in our training-test set and the set of representatives. Since the number of representatives per class is typically much smaller than the number of samples, this approach constitutes a computational ad-

vantage over the Binet-Cauchy kernels. The final kernel is then computed as an inner product of the mapped features. However, this computational advantage comes at the expense of some degradation in the classification accuracy. This effect is particularly accentuated when working with video signals of very high dimensions. At the same time we have also observed that the number of representatives per class affects the final performance, and that increasing the number of representatives, yields better results. Not surprisingly, these facts suggest that for high-dimensional signals the number of representative points should be larger than for signals of much smaller dimension.

There exist several ways in which our methods could be improved. For example, we have extracted the exemplars in a completely unsupervised fashion (*i.e.*, using generalized K-means) by considering all the samples in the training set. However, we know that in SVMs the separating hyperplane is defined by those points close to the decision boundary. This suggests that it might be beneficial to extract the exemplars not from all the points in the dataset but from those that lie "close" to the decision boundary (*e.g.*, those that are not too far from the other cluster). Despite the fact that we have used fairly low-level features in our analysis we have obtained very high classification performance. However, the dataset used in the simulations has been generated in a very controlled environment. This is not the case when coming to real surgeries where there are different factors that make the scene more variable. Factors such as blood, moving organs, smoke, changes in the viewpoint of

the camera, occlusions, etc, may require an additional step of pre-processing in order to extract better features before its use as time serie data. Towards the end goal of having an automated system for training and skill assessment, it would be desirable to jointly perform segmentation and categorization of the time series. Therefore, as a future line of research, it would be interesting to study the capability of such models to jointly segment and classify time-series data since the current approaches assume segmented data.

Bibliography

[1] C. E. Reiley, H. C. Lin, B. Varadarajan, B. Vagolgyi, S. Khudanpur, D. D. Yuh, and G. D. Hager, "Automatic recognition of surgical motions using statistical modeling for capturing variability," in *Medicine Meets Virtual Reality*, 2008, pp. 396–401.

[2] M. Menon and A. Tewari, "Robotic radical prostatectomy and the vattikuti urology institute technique: an interim analysis of results and technical points," *Urology*, vol. 61, no. 4, Supplement 1, pp. 15–20, 2003.

[3] C. C. Abbou, A. Hoznek, L. Salomon, L. E. Olsson, A. Lobontiu, F. Saint, A. Cicco, P. Antiphon, and D. Chopin, "Laparoscopic radical prostatectomy with a remote controlled robot," *The Journal of Urology*, vol. 165, no. 6 Pt 1, pp. 1964–1966, 2001.

[4] W. T. Lowrance, E. B. Elkin, L. M. Jacks, D. S. Yee, T. L. Jang, V. P. Laudone, B. D. Guillonneau, P. T. Scardino, and J. A. Eastham, "Comparative effectiveness of prostate cancer surgical treatments: A population based analysis of

postoperative outcomes," *The Journal of Urology*, vol. 183, no. 4, pp. 1366–1372, 2010.

[5] J. P. Lenihan, C. Kovanda, and U. Seshadri-Kreaden, "What is the learning curve for robotic assisted gynecologic surgery?" *Journal of Minimally Invasive Gynecology*, vol. 15, no. 5, pp. 589 – 594, 2008.

[6] F. Miyawaki, K. Masamune, S. Suzuki, K. Yoshimitsu, and J. Vain, "Scrub nurse robot system - intraoperative motion analysis of a scrub nurse and timed-automata-based model for surgery," *Transactions on Industrial Electronics*, vol. 52, no. 5, pp. 1227–1235, 2005.

[7] N. Padoy, T. Blum, I. Essa, H. Feussner, M. Berger, and N. Navab, "A boosted segmentation method for surgical workflow analysis," in *Int. Conference on Medical Image Computing and Computer Assisted Intervention*, 2007, pp. 102–109.

[8] T. Blum, N. Padoy, H. Feussner, and N. Navab, "Workflow mining for visualization and analysis of surgeries," *International Journal of Computer Assisted Radiology and Surgery*, vol. 3, no. 5, pp. 379–386, 2008.

[9] H. Lin, "Structure in surgical motion," Ph.D. dissertation, Johns Hopkins University, 2010.

[10] F. Lalys, L. Riffaud, D. Bouget, and P. Jannin, "An application-dependent framework for the recognition of high-level surgical tasks in the OR," in *Int.*

Conference on Medical Image Computing and Computer Assisted Intervention, 2011, pp. 331–338.

[11] B. Béjar, L. Zappella, and R. Vidal, "Surgical gesture classification from video data," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012*, ser. Lecture Notes in Computer Science, N. Ayache, H. Delingette, P. Golland, and K. Mori, Eds., vol. 7510. Springer, 2012, pp. 34–41.

[12] N. Padoy, T. Blum, S. Ahmadi, H. Feussner, M. Berger, and N. Navab, "Statistical modeling and recognition of surgical workflow," *Medical Image Analysis*, vol. 16, no. 3, pp. 632 – 641, 2012.

[13] L. Zappella, B. Béjar, G. Hager, and R. Vidal, "Surgical gesture classification from video and kinematic data," *Medical Image Analysis*, vol. 17, no. 7, pp. 732 – 745, 2013.

[14] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto, "Recognition of human gaits," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2001, pp. 52–58.

[15] 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.*, june 2009, pp. 1932 –1939.

[16] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

[17] M. Szummer and R. W. Picard, "Temporal texture modeling," in *IEEE International Conference on Image Processing*, vol. 3, 1996, pp. 823–826.

[18] L. Yuan, F. Wen, C. Liu, and H. Shum, "Synthesizing dynamic texture with closed-loop linear dynamic system," in *European Conference on Computer Vision*, 2004, pp. 603–616.

[19] G. Aggarwal, A. Roy-Chowdhury, and R. Chellappa, "A system identification approach for video-based face recognition," in *International Conference on Pattern Recognition*, 2004, pp. 23–26.

[20] A. Ravichandran, R. Chaudhry, and R. Vidal, "Categorizing dynamic textures using a bag of dynamical systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[21] R. Shumway and D. Stoffer, "An approach to time series smoothing and forecasting using the EM algorithm," *Journal of Time Series Analysis*, vol. 3, no. 4, pp. 253–264, 1982.

[22] P. V. Overschee and B. D. Moor, "Subspace algorithms for the stochastic identification problem," *Automatica*, vol. 29, no. 3, pp. 649–660, 1993.

[23] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines,*

Regularization, Optimization, and Beyond. Cambridge, MA, USA: MIT Press, 2001.

[24] R. Chaudhry and Y. Ivanov, "Fast approximate nearest neighbor methods for non-euclidean manifolds with applications to human activity analysis in videos," in *European Conference on Computer Vision, 2010. ECCV 2010*, 2010.

[25] P. Turaga, A. Veeraraghavan, and R. Chellappa, "Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision," in *CVPR 2008.*, 2008.

[26] A. Ravichandran, R. Chaudhry, and R. Vidal, "View-invariant dynamic texture recognition using a bag of dynamical systems," in *CVPR 2009.*, june 2009, pp. 1651 –1657.

[27] A. B. Chan, E. Coviello, and G. R. G. Lanckriet, "Clustering dynamic textures with the hierarchical em algorithm," in *CVPR 2010.* IEEE, 2010, pp. 2022–2029.

[28] B. Afsari, R. Chaudhry, A. Ravichandran, and R. Vidal, "Group action induced distances for averaging and clustering linear dynamical systems with applications to the analysis of dynamic scenes," in *CVPR 2012.*, june 2012, pp. 2208 –2215.

[29] K. De Cock and B. De Moor, "Subspace angles between arma models," *System Control Letters*, vol. 46, no. 4, pp. 265 –270, Jul 2002.

[30] R. Martin, "A metric for arma processes," *Signal Processing, IEEE Transactions on*, vol. 48, no. 4, pp. 1164 –1170, apr 2000.

[31] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning, 2011, vol. 3, no. 1.

[32] R. Chaudhry and R. Vidal, "Recognition of visual dynamical processes: Theory, kernels and experimental evaluation," Department of Computer Science, Johns Hopkins University, Tech. Rep. 09-01, 2009.

[33] S. Vishwanathan, A. Smola, and R. Vidal, "Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 95–119, 2007.

[34] A. Ghoreyshi and R. Vidal, "Epicardial segmentation in dynamic cardiac MR sequences using priors on shape, intensity, and dynamics, in a level set framework," in *IEEE International Symposium on Biomedical Imaging*, 2007, pp. 860–863.

[35] A. Ravichandran, R. Vidal, and H. Halperin, "Segmenting a beating heart using polysegment and spatial GPCA," in *IEEE International Symposium on Biomedical Imaging*, 2006, pp. 634–637.

[36] A. Ravichandran and R. Vidal, "Video registration using dynamic textures," in *European Conference on Computer Vision*, 2008.

[37] ——, "Video registration using dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 158–171, January 2011.

[38] R. Vidal and A. Ravichandran, "Optical flow estimation and segmentation of multiple moving dynamic textures," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. II, 2005, pp. 516–521.

[39] A. Chan and N. Vasconcelos, "Classifying video with kernel dynamic textures," in *CVPR 2007.*, 2007, pp. 1–6.

[40] P. Saisan, A. Bissacco, A. Chiuso, and S. Soatto, "Modeling and synthesis of facial motion driven by speech," in *European Conference on Computer Vision*, vol. 3, 2004, pp. 456–467.

[41] G. Doretto and S. Soatto, "Editable dynamic textures," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. II, 2003, pp. 137–142.

[42] S. Vishwanathan, A. Smola, and R. Vidal, "Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes," *International Journal of Computer Vision*, vol. 73, no. 1, pp. 95–119, 2007.

[43] A. Chan and N. Vasconcelos, "Probabilistic kernels for the classification of auto-regressive visual processes," in *CVPR 2005.*, vol. 1, 2005, pp. 846–851.

[44] V. Vapnik, *Statistical Learning Theory.* New York: Wiley, 1998.

[45] S. Lacy and D. Bernstein, "Subspace identification with guaranteed stability using constrained optimization," in *American Control Conference, 2002. Proceedings of the 2002*, vol. 4, 2002, pp. 3307–3312 vol.4.

[46] J. F. Sturm, "Using SeDuMi 1.02, A MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1, pp. 625 – 653, 1999.

[47] J. Gallier, "The schur complement and symmetric positive semidefinite (and definite) matrices," *December*, vol. 44, pp. 1–12, 2010.

[48] S. Siddiqi, B. Boots, and G. J. Gordon, "A constraint generation approach to learning stable linear dynamical systems," in *In Advances in Neural Information Processing Systems (NIPS)*, Vancouver, Canada, December 2007.

[49] N. D. Jimenez, B. Afsari, and R. Vidal, "Fast jacobi-type algorithm for computing distances between linear dynamical systems," in *ECCV 2013.* IEEE, 2013, pp. 3682–3687.

[50] K. B. Petersen and M. S. Pedersen, "The matrix cookbook," oct 2008, version 20081110. [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?3274

[51] A. Edelman, T. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM Journal of Matrix Analysis Applications*, vol. 20, no. 2, pp. 303–353, 1998.

[52] Bertsekas, D. P., *Constrained optimization and Lagrange multiplier methods*. New York: Academic Press, 1982.

[53] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.

[54] G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research*, vol. 5, pp. 27–72, 2004.

[55] M. Varma and R. Babu, "More generality in efficient multiple kernel learning," in *International Conference on Machine Learning*, 2009, pp. 1065–1072.

[56] C.-C. Chang and C.-J. Lin, *LIBSVM : a library for support vector machines*, Manual, 2001, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[57] V. Datta, S. Mackay, M. Mandalia, and A. Darzi, "The use of electromagnetic motion tracking analysis to objectively measure open surgical skill in laboratory-based model," *Journal of the American College of Surgery*, vol. 193, pp. 479–485, 2001.

[58] T. Judkins, D. Oleynikov, and N. Stergiou, "Objective evaluation of expert and novice performance during robotic surgical training tasks," *Surgical Endoscopy*, vol. 1, no. 4, 2008.

[59] C. Richards, J. Rosen, B. Hannaford, C. Pellegrini, and M. Sinanan, "Skills evaluation in minimally invasive surgery using force/torque signatures," *Surgical Endoscopy*, vol. 14, pp. 791–798, 2000.

[60] Y. Yamauchi, J. Yamashita, O. Morikawa, R. Hashimoto, M. Mochimaru, Y. Fukui, H. Uno, and K. Yokoyama, "Surgical skill evaluation by force data for endoscopic sinus surgery training system," in *Int. Conference on Medical Image Computing and Computer Assisted Intervention*, 2002, pp. 44–51.

[61] J. Rosen, M. Solazzo, B. Hannaford, and M. Sinanan, "Task decomposition of laparoscopic surgery for objective evaluation of surgical residents' learning curve using hidden Markov model," *Computer Aided Surgery*, vol. 7, no. 1, pp. 49–61, 2002.

[62] C. McKenzie, J. Ibbotson, C. Cao, and A. Lomax, "Hierarchical decomposition of laparoscopic surgery: A human factors approach to investigating the operating room environment," *Journal of Minimally Invasive Therapy and Allied Technologies*, vol. 10(3), pp. 121–127, 2001.

[63] H. C. Lin, I. Shafran, D. Yuh, and G. D. Hager, "Towards automatic skill evaluation: detection and segmentation of robot-assisted surgical motions," *Computer Aided Surgery*, 2006.

[64] A. Dosis, F. Bello, D. Gillies, S. Undre, R. Aggarwal, and A. Darzi, "Laparoscopic

task recognition using hidden Markov models," *Studies in Health Technology and Informatics*, vol. 111, pp. 115–122, 2005.

[65] C. E. Reiley and G. D. Hager, "Task versus subtask surgical skill evaluation of robotic minimally invasive surgery," in *Int. Conference on Medical Image Computing and Computer Assisted Intervention*, 2009, pp. 435–442.

[66] B. Varadarajan, "Learning and inference algorithms for dynamical system models of dextrous motion," Ph.D. dissertation, Johns Hopkins University, 2011.

[67] B. Varadarajan, C. Reiley, H. Lin, S. Khudanpur, and G. Hager, "Data-derived models for segmentation with application to surgical assessment and training," in *Int. Conference on Medical Image Computing and Computer Assisted Intervention*, 2009, pp. 426–434.

[68] J. Leong, M. Nicolaou, L. Atallah, G. Mylonas, A. Darzi, and G. Yang, "HMM assessment of quality of movement trajectory in laparoscopic surgery," in *Int. Conference on Medical Image Computing and Computer Assisted Intervention*, 2006, pp. 752–759.

[69] L. Tao, E. Elhamifar, S. Khudanpur, G. Hager, and R. Vidal, "Sparse hidden markov models for surgical gesture classification and skill evaluation," in *Information Processing in Computed Assisted Interventions*, 2012.

[70] U. Klank, N. Padoy, H. Feussner, and N. Navab, "Automatic feature generation in endoscopic images," *IJCARS*, vol. 3, pp. 331–339, 2008.

[71] T. Blum, H. Feussner, and N. Navab, "Modeling and segmentation of surgical workflow from laparoscopic video," in *Int. Conference on Medical Image Computing and Computer Assisted Intervention*, 2010, pp. 400–407.

[72] F. Lalys, D. Bouget, L. Riffaud, and P. Jannin, "Automatic knowledge-based recognition of low-level tasks in ophthalmological procedures," *International Journal on Computer Assisted Radiology and Surgery*, vol. 8, no. 1, pp. 39–49, 2013.

Vita

Benjamín Béjar Haro received the Electrical Engineering degree from both the Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, and the Technische Universität Darmstadt (TUD), Darmstadt, Germany, in 2006 under the framework of the Double Degree Exchange program. In Sept. 2011 he joined the Vision, Dynamics and Learning lab at the Johns Hopkins University (JHU), Baltimore, MD, USA, as part of the MSE program in Biomedical Engineering. His current research interests include the development and application of convex optimization and machine learning techniques to the problem of automatic surgical gesture recognition in computer assisted intervention systems. In 2012, he was awarded with the 2012 Best Paper Award in Medical Robotics and Computer Assisted Intervention Systems from the Medical Image Computing and Computer Assisted Intervention Society (MICCAI).