International Journal of Computer Vision

DOI: 10.1007/s11263-006-9352-0

# Binet-Cauchy Kernels on Dynamical Systems and its Application to the Analysis of Dynamic Scenes*

S.V.N. VISHWANATHAN AND ALEXANDER J. SMOLA
*Statistical Machine Learning Program, National ICT Australia, Canberra, 0200 ACT, Australia*
SVN.Vishwanathan@nicta.com.au
Alex.Smola@nicta.com.au

RENÉ VIDAL
*Center for Imaging Science, Department of Biomedical Engineering, Johns Hopkins University, 308B Clark Hall, 3400 N. Charles St., Baltimore, MD 21218*
rvidal@cis.jhu.edu

*Received November 22, 2005; Revised May 30, 2006; Accepted May 31, 2006*

*First online version published in July 2006*

**Abstract.** We propose a family of kernels based on the Binet-Cauchy theorem, and its extension to Fredholm operators. Our derivation provides a unifying framework for all kernels on dynamical systems currently used in machine learning, including kernels derived from the behavioral framework, diffusion processes, marginalized kernels, kernels on graphs, and the kernels on sets arising from the subspace angle approach. In the case of linear time-invariant systems, we derive explicit formulae for computing the proposed Binet-Cauchy kernels by solving Sylvester equations, and relate the proposed kernels to existing kernels based on cepstrum coefficients and subspace angles. We show efficient methods for computing our kernels which make them viable for the practitioner.

Besides their theoretical appeal, these kernels can be used efficiently in the comparison of video sequences of dynamic scenes that can be modeled as the output of a linear time-invariant dynamical system. One advantage of our kernels is that they take the initial conditions of the dynamical systems into account. As a first example, we use our kernels to compare video sequences of dynamic textures. As a second example, we apply our kernels to the problem of clustering short clips of a movie. Experimental evidence shows superior performance of our kernels.

**Keywords:** Binet-Cauchy theorem, ARMA models and dynamical systems, sylvester equation, kernel methods, reproducing kernel Hilbert spaces, dynamic scenes, dynamic textures

## 1. Introduction

The past few years have witnessed an increasing interest in the application of system-theoretic techniques to modeling visual dynamical processes. For instance, Doretto et al. (2003) and Soatto et al. (2001) model the appearance of dynamic textures, such as water, foliage, hair, etc. as the output of an Auto-Regressive Moving Average (ARMA) model; Saisan et al. (2001) use ARMA models to represent human gaits, such as walking, running, jumping, etc.; Aggarwal et al. (2004) use ARMA models to describe the appearance of moving

*Parts of this paper were presented at SYSID 2003 and NIPS 2004.

faces; and Vidal et al. (2005) combine ARMA models with GPCA for clustering video shots in dynamic scenes.

However, in practical applications we may not only be interested in obtaining a *model* for the process (identification problem), but also in determining whether two video sequences correspond to the same process (classification problem) or identifying which process is being observed in a given video sequence (recognition problem). Since the space of models for dynamical processes typically has a non-Euclidean structure,[1] the above problems have naturally raised the issue of how to do estimation, classification and recognition on such spaces.

The study of classification and recognition problems has been the mainstream areas of research in machine learning for the past decades. Among the various methods for nonparametric estimation that have been developed, *kernel methods* have become one of the mainstays as witnessed by a large number of books (Vapnik, 1995, 1998; Cristianini and Shawe-Taylor, 2000; Herbrich, 2002; Schölkopf and Smola, 2002). However, not much of the existing literature has addressed the design of kernels in the context of dynamical systems. To the best of our knowledge, the metric for ARMA models based on comparing their cepstrum coefficients (Martin, 2000) is one of the first papers to address this problem. De Cock and De Moor (2002) extended this concept to ARMA models in state-space representation by considering the subspace angles between the observability subspaces. Recently, Wolf and Shashua (2003) demonstrated how subspace angles can be efficiently computed by using kernel methods. In related work, Shashua and Hazan (2005) show how tensor products can be used to construct a general family of kernels on sets.

A downside of these approaches is that the resulting kernel is independent of the initial conditions. While this might be desirable in many applications, it might cause potential difficulties in others. For instance, not every initial configuration of the joints of the human body is appropriate for modeling human gaits.

### 1.1. Paper Contributions

In this paper, we attempt to bridge the gap between nonparametric estimation methods and dynamical systems. We build on previous work using explicit embedding constructions and regularization theory (Smola and Kondor, 2003; Wolf and Shashua, 2003; Smola and Vishwanathan, 2003) to propose a family of kernels explicitly designed for dynamical systems. More specifically, we propose to compare two dynamical systems by computing traces of compound matrices of order $q$ built from the system trajectories. The Binet-Cauchy theorem (Aitken, 1946) is then invoked to show that such traces satisfy the properties of a kernel.

We then show that the two strategies which have been used in the past are particular cases of the proposed unifying framework.

- The first family is based directly on the time-evolution properties of the dynamical systems (Smola and Kondor, 2003; Wolf and Shashua, 2003; Smola and Vishwanathan, 2003). Examples include the diffusion kernels of Kondor and Lafferty (2002), the graph kernels of Gärtner et al. (2003) Kashima et al. (2003), and the similarity measures between graphs of Burkhardt (2004). We show that all these kernels can be found as particular cases of our framework, the so-called *trace kernels*, which are obtained by setting $q = 1$.
- The second family is based on the idea of extracting coefficients of a dynamical system (viz. ARMA models), and defining a metric in terms of these quantities. Examples include the Martin distance (Martin, 2000) and the distance based on subspace angles between observability subspaces (De Cock and De Moor, 2002). We show that these metrics arise as particular cases of our framework, the so-called *determinant kernels*, by suitable preprocessing of the trajectory matrices and by setting the order of the compound matrices $q$ to be equal to the order of the dynamical systems. The advantage of our framework is that it can take the initial conditions of the systems into account explicitly, while the other methods cannot.

Finally, we show how the proposed kernels can be used to classify dynamic textures and segment dynamic scenes by modeling them as ARMA models and computing kernels on the ARMA models. Experimental evidence shows the superiority of our kernels in capturing differences in the dynamics of video sequences.

### 1.2. Paper Outline

We briefly review kernel methods and Support Vector Machines (SVMs) in Section 2, and introduce the Binet-Cauchy theorem and associated kernels in Section 3. We discuss methods to compute the

Binet-Cauchy kernels efficiently in Section 3.3. We then concentrate on dynamical systems and define kernels related to them in Section 4. We then move on to ARMA models and derive closed form solutions for kernels defined on ARMA models in Section 5. In Section 6 we concentrate on formally showing the relation between our kernels and many well known kernels on graphs, and sets. We extend our kernels to deal with non-linear dynamical systems in Section 7 and apply them to the analysis of dynamic scenes in Section 8. Finally, we conclude with a discussion in Section 9.

## 2. Support Vector Machines and Kernels

In this section we give a brief overview of binary classification with SVMs and kernels. For a more extensive treatment, including extensions such as multi-class settings, the $\nu$-parameterization, loss functions, etc., we refer the reader to Schölkopf and Smola (2002), Cristianini and Shawe-Taylor (2000), Herbrich (2002) and the references therein.

Given $m$ observations $(x_i, y_i)$ drawn iid (independently and identically distributed) from a distribution over $\mathcal{X} \times \mathcal{Y}$ our goal is to find a function $f : \mathcal{X} \to \mathcal{Y}$ which classifies observations $x \in \mathcal{X}$ into classes $+1$ and $-1$. In particular, SVMs assume that $f$ is given by

$$f(x) = \text{sign}(\langle w, x\rangle + b),$$

where the sets $\{x|f(x) \geq 0\}$ and $\{x|f(x) \leq 0\}$ denote half-spaces separated by the hyperplane $H(w, b) := \{x|\langle w, x\rangle + b = 0\}$.

In the case of linearly separable datasets, that is, if we can find $(w, b)$ such that all $(x_i, y_i)$ pairs satisfy $y_i f(x_i) = 1$, the *optimal separating hyperplane* is given by the $(w, b)$ pair for which all $x_i$ have maximum distance from $H(w, b)$. In other words, it is the hyperplane which separates the two sets with the largest possible margin.

Unfortunately, not all sets are linearly separable, hence we need to allow a slack in the separation of the two sets. Without going into details (which can be found in Schölkopf and Smola (2002)) this leads to the optimization problem:

$$\begin{aligned} \underset{w,b,\xi}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{subject to } & y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i,\ \xi_i \geq 0, \\ & \forall 1 \leq i \leq m. \end{aligned} \tag{1}$$

Here the constraint $\forall i \;\; y_i\left(\langle w, x_i\rangle + b\right) \geq 1$ ensures that each $(x_i, y_i)$ pair is classified correctly. The slack variable $\xi_i$ relaxes this condition at penalty $C\xi_i$. Finally, minimization of $\|w\|^2$ ensures maximization of the margin by seeking the shortest $w$ for which the condition $\forall i \;\; y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i$ is still satisfied.

An important property of SVMs is that the optimal solution $w$ can be computed as a linear combination of some of the data points via $w = \sum_i y_i\alpha_i x_i$. Typically many coefficients $\alpha_i$ are 0, which considerably speeds up the estimation step, since these terms can be dropped from the calculation of the optimal separating hyperplane. The data points associated with the nonzero terms are commonly referred to as *support vectors*, since they support the optimal separating hyperplane. Efficient codes exist for the solution of (1) (Platt, 1999; Chang and Lin, 2001; Joachims, 1998; Vishwanathan et al., 2003).

### 2.1. Kernel Expansion

To obtain a nonlinear classifier, one simply replaces the observations $x_i$ by $\Phi(x_i)$. That is, we extract *features* $\Phi(x_i)$ from $x_i$ and compute a linear classifier in terms of the features. Note that there is no need to compute $\Phi(x_i)$ explicitly, since $\Phi$ only appears in terms of dot products:

- $\langle\Phi(x), w\rangle$ can be computed by exploiting the linearity of the scalar product, which leads to $\sum_i \alpha_i y_i \langle\Phi(x), \Phi(x_i)\rangle$.
- Likewise $\|w\|^2$ can be expanded in terms of a linear combination scalar products by exploiting the linearity of scalar products twice to obtain $\sum_{i,j} \alpha_i\alpha_j y_i y_j \langle\Phi(x_i), \Phi(x_j)\rangle$.

Furthermore, note that if we define

$$k(x, x') := \langle\Phi(x), \Phi(x')\rangle,$$

we may use $k(x, x')$ wherever $\langle x, x'\rangle$ occurs. The mapping $\Phi$ now takes $\mathcal{X}$ to what is called a Reproducing Kernel Hilbert Space (RKHS). This mapping is often referred to as the *kernel trick* and the resulting hyperplane (now in the RKHS defined by $k(\cdot, \cdot)$) leads to the following classification function

$$f(x) = \langle\Phi(x), w\rangle + b = \sum_{i=1}^{m}\alpha_i y_i k(x_i, x) + b.$$

The family of methods which relies on the kernel trick are popularly called *kernel methods* and SVMs are one of the most prominent kernel methods. The main advantage of kernel methods stems from the fact that they do not assume a vector space structure on $\mathcal{X}$. As a result, non-vectorial representations can be handled elegantly as long as we can define meaningful kernels. In the following sections, we show how to define kernels on dynamical systems, i.e., kernels on a set of points $\{x_i\}$ whose temporal evolution is governed by a dynamical system.

## 3. Binet-Cauchy Kernels

In this section we provide a general framework for defining kernels on Fredholm operators. We show that our kernels are positive definite and hence admissible. Finally, we present algorithms for computing our kernels efficiently. While the kernels defined in this section are abstract and generic, we will relate them to matrices and dynamical systems in the following section.

### 3.1. The General Composition Formula

We begin by defining compound matrices. They arise by picking subsets of entries of a matrix and computing their determinants.

*Definition 1* (Compound Matrix). Let $A \in \mathbb{R}^{m \times n}$. For $q \leq \min(m, n)$, define $I_q^n = \{\mathbf{i} = (i_1, i_2, \ldots, i_q) : 1 \leq i_1 < \cdots < i_q \leq n,\ i_i \in \mathbb{N}\}$, and likewise $I_q^m$. The compound matrix of order $q$, $C_q(A)$, is defined as

$$[C_q(A)]_{\mathbf{i},\mathbf{j}} := \det(A(i_k, j_l))_{k,l=1}^q \quad \text{where } \mathbf{i} \in I_q^n \text{ and } \mathbf{j} \in I_q^m.$$

Here $\mathbf{i}, \mathbf{j}$ are assumed to be arranged in lexicographical order.

**Theorem 2** (Binet-Cauchy). *Let $A \in \mathbb{R}^{l \times m}$ and, $B \in \mathbb{R}^{l \times n}$. For $q \leq \min(m, n, l)$ we have $C_q(A^\top B) = C_q(A)^\top C_q(B)$.*

When $q = m = n = l$ we have $C_q(A) = \det(A)$, and the Binet-Cauchy theorem becomes the well known identity $\det(A^\top B) = \det(A)\det(B)$. On the other hand, when $q = 1$ we have $C_1(A) = A$, and Theorem 2 reduces to a tautology.

**Theorem 3** (Binet-Cauchy for Semirings). *When the common semiring $(\mathbb{R}, +, \cdot, 0, 1)$ is replaced by an abstract semiring $(\mathbb{K}, \oplus, \odot, \bar{0}, \bar{1})$ the equality $C_q(A^\top B) = C_q(A)^\top C_q(B)$ still holds. Here all operations occur on the monoid $\mathbb{K}$, addition and multiplication are replaced by $\oplus, \odot$, and $(\bar{0}, \bar{1})$ take the role of $(0, 1)$.*

For ease of exposition, in the rest of the paper we will use the common semiring $(\mathbb{R}, +, \cdot, 0, 1)$, but with minor modifications our results hold for general semirings as well.

A second extension of Theorem 2 is to replace matrices by Fredholm operators, as they can be expressed as integral operators with corresponding kernels. In this case, Theorem 3.2 becomes a statement about convolutions of integral kernels.

*Definition 4* (Fredholm Operator). A Fredholm operator is a bounded linear operator between two Hilbert spaces with closed range and whose kernel and cokernel are finite-dimensional.

**Theorem 5** (Kernel Representation of Fredholm Operators). *Let $A : L_2(\mathcal{Y}) \to L_2(\mathcal{X})$ and, $B : L_2(\mathcal{Y}) \to L_2(\mathcal{Z})$ be Fredholm operators. Then there exists some $k_A : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ such that for all $f \in L_2(\mathcal{X})$ we have*

$$[Af](x) = \int_{\mathcal{Y}} k_A(x, y) f(y) dy.$$

*Moreover, for the composition $A^\top B$ we have $k_{A^\top B}(x, z) = \int_{\mathcal{Y}} k_{A^\top}(x, y) k_B(y, z) dy$.*

Here the convolution of kernels $k_A$ and $k_B$ plays the same role as the matrix multiplication. To extend the Binet-Cauchy theorem we need to introduce the analog of compound matrices:

*Definition 6* (Compound Kernel and Operator). Denote by $\mathcal{X}, \mathcal{Y}$ ordered sets and let $k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Define $I_q^{\mathcal{X}} = \{\mathbf{x} \in \mathcal{X}^q : x_1 \leq \cdots \leq x_q\}$, and likewise $I_q^{\mathcal{Y}}$. Then the compound kernel of order $q$ is defined as

$$k^{[q]}(\mathbf{x}, \mathbf{y}) := \det(k(x_k, y_l))_{k,l=1}^q \quad \text{where } \mathbf{x} \in I_q^{\mathcal{X}} \text{ and } \mathbf{y} \in I_q^{\mathcal{Y}}.$$

If $k$ is the integral kernel of an operator $A$ we define $C_q(A)$ to be the integral operator corresponding to $k^{[q]}$.

**Theorem 7** (General Composition Formula (Pinkus, 1996)). *Let $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ be ordered sets and let $A : L_2(\mathcal{Y}) \to L_2(\mathcal{X})$, $B : L_2(\mathcal{Y}) \to L_2(\mathcal{Z})$ be Fredholm operators. Then for $q \in \mathbb{N}$ we have*

$$C_q(A^\top B) = C_q(A)^\top C_q(B).$$

To recover Theorem 2 from Theorem 7 set $\mathcal{X} = [1..m]$, $\mathcal{Y} = [1..n]$ and $\mathcal{Z} = [1..l]$.

### 3.2. Kernels

The key idea in turning the Binet-Cauchy theorem and its various incarnations into a kernel is to exploit the fact that $\operatorname{tr} A^\top B$ and $\det A^\top B$ are kernels on operators $A, B$. Following (Wolf and Shashua, 2003) we extend this by replacing $A^\top B$ with some functions $\psi(A)^\top \psi(B)$ involving compound operators.

**Theorem 8** (Trace and Determinant Kernel). *Let $A, B : L_2(\mathcal{X}) \to L_2(\mathcal{Y})$ be Fredholm operators and let $S : L_2(\mathcal{Y}) \to L_2(\mathcal{Y})$, $T : L_2(\mathcal{X}) \to L_2(\mathcal{X})$ be positive trace-class operators. Then the following two kernels are well defined, and they are positive semi-definite:*

$$k(A, B) = \operatorname{tr}[SA^\top TB] \tag{2}$$

$$k(A, B) = \det[SA^\top TB]. \tag{3}$$

Note that determinants are not defined in general for infinite dimensional operators, hence our restriction to Fredholm operators $A, B$ in (3).

**Proof:** Recall that $k(A, B)$ is a valid positive semi-definite kernel if it can be written as $\psi(A)^\top \psi(B)$ for some function $\psi(\cdot)$. Observe that $S$ and $T$ are positive and compact. Hence, they admit a decomposition into $S = V_S V_S^\top$ and $T = V_T^\top V_T$. By virtue of the commutativity of the trace we have that

$$k(A, B) = \operatorname{tr}\left( \underbrace{[V_T A V_S]^\top}_{\psi(A)^\top} \underbrace{[V_T B V_S]}_{\psi(B)} \right).$$

Analogously, using the Binet-Cauchy theorem, we can decompose the determinant. The remaining terms $V_T A V_S$ and $V_T B V_S$ are again Fredholm operators for which determinants are well defined. □

Next we use special choices of $A, B, S, T$ involving compound operators directly to state the main theorem of our paper.

**Theorem 9** (Binet-Cauchy Kernel). *Under the assumptions of Theorem 8 it follows that for all $q \in \mathbb{N}$ the kernels $k(A, B) = \operatorname{tr} C_q\left[SA^\top TB\right]$ and $k(A, B) = \det C_q\left[SA^\top TB\right]$ are positive semi-definite.*

**Proof:** We exploit the factorization $S = V_S V_S^\top, T = V_T^\top V_T$ and apply Theorem 7. This yields $C_q(SA^\top TB) = C_q(V_T A V_S)^\top C_q(V_T B V_S)$, which proves the theorem. □

Finally, we define a kernel based on the Fredholm determinant itself. It is essentially a weighted combination of Binet-Cauchy kernels. Fredholm determinants are defined as follows (Pinkus, 1996):

$$D(A, \mu) := \sum_{q=1}^{\infty} \frac{\mu^q}{q!} \operatorname{tr} C_q(A).$$

This series converges for all $\mu \in \mathbb{C}$, and is an entire function of $\mu$. It suggests a kernel involving weighted combinations of the kernels of Theorem 9. We have the following:

**Corollary 10** (Fredholm Kernel). *Let $A, B, S, T$ as in Theorem 9 and let $\mu > 0$. Then the following kernel is positive semi-definite:*

$$k(A, B) := D(A^\top B, \mu) \text{ where } \mu > 0.$$

$D(A^\top B, \mu)$ is a weighted combination of the kernels discussed above, and hence a valid positive semi-definite kernel. The exponential down-weighting via $\frac{1}{q!}$ ensures that the series converges even in the case of exponential growth of the values of the compound kernel.

### 3.3. Efficient Computation

At first glance, computing the kernels of Theorem 9 and Corollary 10 presents a formidable computational task even in the finite dimensional case. If $A, B \in \mathbb{R}^{m \times n}$, the matrix $C_q(A^\top B)$ has $\binom{n}{q}$ rows and columns and each of the entries requires the computation of a determinant of a $q$-dimensional matrix. A brute-force approach would involve $O(q^3 n^q)$ operations (assuming $2q \leq n$). Clearly we need more efficient techniques.

When computing determinants, we can take recourse to Franke's Theorem which states that (Gröbner, 1965):

$$\det C_q(A) = (\det A)^{\binom{n-1}{q-1}}.$$

This can be seen as follows: the compound matrix of an orthogonal matrix is orthogonal and consequently its determinant is unity. Subsequently, use the SVD factorization (Golub and Van Loan, 1996) of $A$ to compute the determinant of the compound matrix of a diagonal matrix which yields the desired result. Consequently,

$$k(A, B) = \det C_q[SA^\top TB] = (\det[SA^\top TB])^{\binom{n-1}{q-1}}.$$

This indicates that the determinant kernel may be of limited use, due to the typically quite high power in the exponent. Kernels building on $\operatorname{tr} C_q$ are not plagued by this problem, and we give an efficient recursion below. It follows from the ANOVA kernel recursion of Burges and Vapnik (1995).

**Lemma 11.** *Denote by $A \in \mathbb{C}^{m\times m}$ a square matrix and let $\lambda_1, \ldots, \lambda_m$ be its eigenvalues. Then* $\operatorname{tr} C_q(A)$ *can be computed by the following recursion:*

$$\operatorname{tr} C_q(A) = \frac{1}{q}\sum_{j=1}^{q}(-1)^{j+1}\bar{C}_{q-j}(A)\bar{C}_j(A)$$

$$\text{where } \bar{C}_q(A) = \sum_{j=1}^{n}\lambda_j^q. \tag{4}$$

**Proof:** We begin by writing $A$ in its Jordan normal form as $A = PDP^{-1}$ where $D$ is a block diagonal, upper triangular matrix. Furthermore, the diagonal elements of $D$ consist of the eigenvalues of $A$. Repeated application of the Binet-Cauchy Theorem yields

$$\begin{aligned}\operatorname{tr} C_q(A) &= \operatorname{tr} C_q(P)C_q(D)C_q(P^{-1}) \\ &= \operatorname{tr} C_q(D)C_q(P^{-1})C_q(P) = \operatorname{tr} C_q(D)\end{aligned}$$

For a triangular matrix the determinant is the product of its diagonal entries. Since all the square submatrices of $D$ are upper triangular, to construct $\operatorname{tr}(C_q(D))$ we need to sum over all products of exactly $q$ eigenvalues. This is analog to the requirement of the ANOVA kernel of Burges and Vapnik (1995). In its simplified version it can be written as (4), which completes the proof. □

We can now compute the Jordan normal form of $SA^\top TB$ in $O(n^3)$ time and apply Lemma 11 directly to it to compute the kernel value.

Finally, in the case of Fredholm determinants, we can use the recursion directly, because for $n$-dimensional matrices the sum terminates after $n$ terms. This is no more expensive than computing $\operatorname{tr} C_q$ directly.

## 4. Kernels and Dynamical Systems

In this section we will concentate on dynamical systems. We discuss the behavioral framework and show how the Binet-Cauchy kernels can be applied to define kernels. In particular, we will concentrate on the trace kernel ($q = 1$) and the determinant kernel ($q = n$). We will show how kernels can be defined using model parameters, and on initial conditions of a dynamical system.

### *4.1. Dynamical Systems*

We begin with some definitions. The state of a dynamical system is defined by some $x \in \mathcal{H}$, where $\mathcal{H}$ is assumed to be a RKHS with its kernel denoted by $\kappa(\cdot, \cdot)$. Moreover, we assume that the temporal evolution of $x \in \mathcal{H}$ is governed by

$$x_{\mathbf{A}}(t) := \mathbf{A}(x, t) \text{ for } t \in \mathcal{T},$$

where $t$ is the time of the measurement and $\mathbf{A} : \mathcal{H} \times \mathcal{T} \to \mathcal{H}$ denotes a set of operators indexed by $\mathcal{T}$. Furthermore, unless stated otherwise, we will assume that for every $t \in \mathcal{T}$, the operator $\mathbf{A}(\cdot, t) : \mathcal{H} \to \mathcal{H}$ is linear and drawn from some fixed set $\mathcal{A}$.[2] We will choose $\mathcal{T} = \mathbb{N}_0$ or $\mathcal{T} = \mathbb{R}_0^+$, depending on whether we wish to deal with discrete-time or continuous-time systems.

We can also assume that $x$ is a random variable drawn from $\mathcal{H}$ and $\mathbf{A}(\cdot, t)$ is a random operator drawn from the set $\mathcal{A}$. In this case, we need to assume that both $\mathcal{H}$ and $\mathcal{A}$ are endowed with suitable probability measures. For instance, we may want to consider initial conditions corrupted by additional noise or Linear Time Invariant (LTI) systems with additive noise.

Also note that $x_{\mathbf{A}}(t)$ need not be the only variable involved in the time evolution process. For instance, for partially observable models, we may only see $y_{\mathbf{A}}(t)$ which depends on the evolution of a hidden state $x_{\mathbf{A}}(t)$. These cases will be discussed in more detail in Section 5.

### 4.2. Trajectories

When dealing with dynamical systems, one may compare their similarities by checking whether they satisfy similar functional dependencies. For instance, for LTI systems one may want to compare the transfer functions, the system matrices, the poles and/or zeros, etc. This is indeed useful in determining when systems are similar whenever suitable parameterizations exist. However, it is not difficult to find rather different functional dependencies, which, nonetheless, behave almost identically, e.g. as long as the domain of initial conditions is sufficiently restricted. For instance, consider the maps

$$x \leftarrow a(x) = |x|^p \text{ and } x \leftarrow b(x) = \min(|x|^p, |x|)$$

for $p > 1$. While $a$ and $b$ clearly differ, the two systems behave identically for all initial conditions satisfying $|x| \leq 1$. On the other hand, for $|x| > 1$ system $b$ is stable while $a$ could potentially diverge. This example may seem contrived, but for more complex maps and higher dimensional spaces such statements are not quite as easily formulated.

One way to amend this situation is to compare *trajectories* of dynamical systems and derive measures of similarity from them. The advantage of this approach is that it is independent of the parameterization of the system. This approach is in spirit similar to the behavioral framework of (1986a, b, 1987), which identifies systems by identifying trajectories. However, it has the disadvantage that one needs efficient mechanisms to compute the trajectories, which may or may not always be available. We will show later that in the case of ARMA models or LTI systems such computations can be performed efficiently by solving a series of Sylvester equations.

In keeping with the spirit of the behavioral framework, we will consider the pairs $(x, \mathbf{A})$ only in terms of the trajectories which they generate. We will focus our attention on the map

$$\text{Traj}_{\mathbf{A}} : \mathcal{H} \to \mathcal{H}^{\mathcal{T}} \quad \text{where} \quad \text{Traj}_{\mathbf{A}}(x) := \mathbf{A}(x, \cdot). \tag{5}$$

In other words, we define $\text{Traj}_{\mathbf{A}}(x)(t) = \mathbf{A}(x, t)$. The following lemma provides a characterization of this mapping.

**Lemma 12.** *Let $\mathbf{A} : \mathcal{H} \times \mathcal{T} \to \mathcal{H}$ be such that for every $t \in \mathcal{T}$, $\mathbf{A}(\cdot, t)$ is linear. Then the mapping* $\text{Traj}_{\mathbf{A}}$ *defined by* (5) *is linear.*

**Proof:** Let, $x, y \in \mathcal{H}$ and $\alpha, \beta \in \mathbb{R}$. For a fixed $t$ the operator $\mathbf{A}(\cdot, t)$ is linear and hence we have

$$\begin{aligned}\text{Traj}_{\mathbf{A}}(\alpha x + \beta y)(t) &= \mathbf{A}(\alpha x + \beta y, t) \\ &= \alpha\, \mathbf{A}(x, t) + \beta\, \mathbf{A}(y, t).\end{aligned}$$

Since the above holds for every $t \in \mathcal{T}$ we can write

$$\text{Traj}_{\mathbf{A}}(\alpha x + \beta y) = \alpha\, \text{Traj}_{\mathbf{A}}(x) + \beta\, \text{Traj}_{\mathbf{A}}(y),$$

from which the claim follows. □

The fact that the state space $\mathcal{H}$ is a RKHS and $\text{Traj}_{\mathbf{A}}$ defines a linear operator will be used in the sequel to define kernels on trajectories.

To establish a connection between the Binet-Cauchy theorem and dynamical systems we only need to realize that the trajectories $\text{Traj}(x, \mathbf{A})$ are linear operators. As we shall see, (2) and some of its refinements lead to kernels which carry out comparisons of state pairs. Equation (3), on the other hand, can be used to define kernels via subspace angles. In this case, one whitens the trajectories before computing their determinant. We give technical details in Section 6.1.

### 4.3. Trace Kernels

Computing $\text{tr}\, A^\top B$ means taking the sum over scalar products between the rows of $A$ and $B$. For trajectories this amounts to summing over $\kappa(x_{\mathbf{A}}(t), x'_{\mathbf{A}}(t)) = \langle x_{\mathbf{A}}(t), x'_{\mathbf{A}'}(t) \rangle$ with respect to $t$. There are two possible strategies to ensure that the sum over $t$ converges, and consequently ensure that the kernel is well defined: We can make strong assumptions about the operator norm of the linear operator $\mathbf{A}$. This is the approach followed, for instance, by Kashima et al. (2004). Alternatively, we can use an appropriate probability measure $\mu$ over the domain $\mathcal{T}$ which ensures convergence. The exponential discounting schemes

$$\begin{aligned}\mu(t) &= \lambda^{-1} e^{-\lambda t} \text{ for } \mathcal{T} = \mathbb{R}_0^+ \\ \mu(t) &= \frac{1}{1 - e^{-\lambda}} e^{-\lambda t} \text{ for } \mathcal{T} = \mathbb{N}_0\end{aligned}$$

are popular choice in reinforcement learning and control theory (Sutton and Barto, 1998; Baxter and Bartlett, 1999). Another possible measure is

$$\mu(t) = \delta_\tau(t) \tag{6}$$

where $\delta_\tau$ corresponds to the Kronecker-$\delta$ for $\mathcal{T} = \mathbb{N}_0$ and to the Dirac's $\delta$-distribution for $\mathcal{T} = \mathbb{R}_0^+$.

The above considerations allow us to extend (2) to obtain the following kernel

$$k((x, \mathbf{A}), (x', \mathbf{A}')) := \mathbb{E}_{t\sim\mu(t)}\left[\kappa\left(x_{\mathbf{A}}(t), x'_{\mathbf{A}'}(t)\right)\right]. \quad (7)$$

Recall that a convex combination of kernels is a kernel. Hence, taking expectations over kernel values is still a valid kernel. We can now specialize the above equation to define kernels on model parameters as well as kernels on initial conditions.

**Kernels on Model Parameters:** We can specialize (7) to kernels on initial conditions or model parameters, simply by taking expectations over a distribution of them. This means that we can define

$$k(\mathbf{A}, \mathbf{A}') := \mathbb{E}_{x,x'}[k((x, \mathbf{A}), (x', \mathbf{A}'))]. \quad (8)$$

However, we need to show that (8) actually is positive semi-definite.

**Theorem 13.** *Assume that* $\{x, x'\}$ *are drawn from an infinitely extendable and exchangeable probability distribution* (de Finetti, 1990). *Then* (8) *is a positive semi-definite kernel.*

**Proof:** By De Finetti's theorem, infinitely exchangeable distributions arise from conditionally independent random variables (de Finetti, 1990). In practice this means that

$$p(x, x') = \int p(x|c)p(x'|c)dp(c)$$

for some $p(x|c)$ and a measure $p(c)$. Hence we can rearrange the integral in (8) to obtain

$$k(\mathbf{A}, \mathbf{A}') = \int [k((x, \mathbf{A}), (x', \mathbf{A}'))dp(x|c)dp(x'|c)]\, dp(c).$$

Here the result of the inner integral is a kernel by the decomposition admitted by positive semi-definite functions. Taking a convex combination of such kernels preserves positive semi-definiteness, hence $k(\mathbf{A}, \mathbf{A}')$ is a kernel. □

In practice, one typically uses either $p(x, x') = p(x)p(x')$ if independent averaging over the initial conditions is desired, or $p(x, x') = \delta(x - x')p(x)$ whenever the averaging is assumed to occur synchronously.

**Kernels on Initial Conditions:** By the same strategy, we can also define kernels exclusively on initial conditions $x, x'$ simply by averaging over the dynamical systems they should be subjected to:

$$k(x, x') := \mathbb{E}_{\mathbf{A},\mathbf{A}'}[k((x, \mathbf{A}), (x', \mathbf{A}'))].$$

As before, whenever $p(\mathbf{A}, \mathbf{A}')$ is infinitely exchangeable in $\mathbf{A}, \mathbf{A}'$, the above equation corresponds to a proper kernel.[3] Note that in this fashion the metric imposed on initial conditions is one that follows naturally from the particular dynamical system under consideration. For instance, differences in directions which are rapidly contracting carry less weight than a discrepancy in a divergent direction of the system.

### *4.4. Determinant Kernels*

Instead of computing traces of $\mathrm{Traj}_{\mathbf{A}}(x)^\top \mathrm{Traj}_{\mathbf{A}'}(x')$ we can follow (3) and compute determinants of such expressions. As before, we need to assume the existence of a suitable measure which ensures the convergence of the sums.

For the sake of computational tractability one typically chooses a measure with finite support. This means that we are computing the determinant of a kernel matrix, which in turn is then treated as a kernel function. This allows us to give an information-theoretic interpretation to the so-defined kernel function. Indeed, Gretton et al. (2003) and Bach and Jordan (2002) show that such determinants can be seen as measures of the independence between sequences. This means that independent sequences can now be viewed as orthogonal in some feature space, whereas a large overlap indicates statistical correlation.

## 5. Kernels on Linear Dynamical Models

In this section we will specialize our kernels on dynamical systems to linear dynamical systems. We will concentrate on both discrete as well as continuous time variants, and derive closed form equations. We also present strategies for efficiently computing these kernels.

### 5.1. ARMA Models

A special yet important class of dynamical systems are ARMA models of the form

$$\begin{aligned} y_t &= Cx_t + w_t \quad \text{where } w_t \sim \mathcal{N}(0, R), \\ x_{t+1} &= Ax_t + v_t \quad \text{where } v_t \sim \mathcal{N}(0, Q), \end{aligned} \tag{9}$$

where the driving noise $w_t$, $v_t$ are assumed to be zero mean iid normal random variables, $y_t \in \mathcal{Y}$ are the observed random variables at time $t$, $x_t \in \mathcal{X}$ are the latent variables, and the matrices $A$, $C$, $Q$, and $R$ are the parameters of the dynamical system. These systems are also commonly known as Kalman Filters. Throughout the rest of the paper we will assume that $\mathcal{Y} = \mathbb{R}^m$, and $\mathcal{X} = \mathbb{R}^n$. In may applications it is typical to assume that $m \gg n$.

We will now show how to efficiently compute an expression for the trace and determinant kernels between two ARMA models $(x_0, A, C)$ and $(x_0', A', C')$, where $x_0$ and $x_0'$ are the initial conditions.

***5.1.1. Trace Kernels on ARMA Models.*** If one assumes an exponential discounting $\mu(t) = e^{-\lambda t}$ with rate $\lambda > 0$, then the trace kernel for ARMA models is

$$\begin{aligned} &k((x_0, A, C), (x_0', A', C')) \\ &\quad := \mathbb{E}_{v,w}\left[\sum_{t=0}^{\infty} e^{-\lambda t} y_t^\top W y_t'\right], \end{aligned} \tag{10}$$

where $W \in \mathbb{R}^{m \times m}$ is a user-defined positive semidefinite matrix specifying the kernel $\kappa(y_t, y_t') := y_t^\top W y_t'$. By default, one may choose $W = \mathbf{1}$, which leads to the standard Euclidean scalar product between $y_t$ and $y_t'$.

A major difficultly in computing (10) is that it involves the computation of an infinite sum. But as the following lemma shows, we can obtain a closed form solution to the above equation.

**Theorem 14.** *If $e^{-\lambda}\|A\|\|A'\| < 1$, and the two ARMA models $(x_0, A, C)$, and $(x_0', A', C')$ evolve with the same noise realization then the kernel of* (10) *is given by*

$$k = x_0^\top M x_0' + \frac{1}{1 - e^{-\lambda}} \operatorname{tr}[QM + WR], \tag{11}$$

*where $M$ satisfies*

$$M = e^{-\lambda} A^\top M A' + C^\top W C'.$$

**Proof:** See Appendix A. □

In the above theorem, we assumed that the two ARMA models evolve with the same noise realization. In many applications this might not be a realistic assumption. The following corollary addresses the issue of computation of the kernel when the noise realizations are assumed to be independent.

**Corollary 15.** *Same setup as above, but the two ARMA models $(x_0, A, C)$, and $(x_0', A', C')$ now evolve with independent noise realizations. Then* (10) *is given by*

$$k = x_0^\top M x_0', \tag{12}$$

*where $M$ satisfies*

$$M = e^{-\lambda} A^\top M A' + C^\top W C'.$$

**Proof** [Sketch]**:** [The second term in the RHS of (11) is contribution due to the covariance of the noise model. If we assume that the noise realizations are independent (uncorrelated) the second term vanishes. □

In case we wish to be independent of the initial conditions, we can simply take the expectation over $x_0, x_0'$. Since the only dependency of (11) on the initial conditions manifests itself in the first term of the sum, the kernel on the parameters of the dynamical systems $(A, C)$ and $(A', C')$ is

$$\begin{aligned} &k((A, C), (A', C')) \\ &\quad := \mathbb{E}_{x_0, x_0'}[k((x_0, A, C), \times (x_0', A', C'))] \\ &\quad = \operatorname{tr} \Sigma M + \frac{1}{1 - e^{-\lambda}} \operatorname{tr}[QM + WR], \end{aligned}$$

where $\Sigma$ is the covariance matrix of the initial conditions $x_0, x_0'$ (assuming that we have zero mean). As before, we can drop the second term if we assume that the noise realizations are independent.

An important special case are fully observable ARMA models, i.e., systems with $C = \mathbf{I}$ and $R = \mathbf{0}$. In this case, the expression for the trace kernel (11) reduces to

$$k = x_0^\top M x_0' + \frac{1}{1 - e^{-\lambda}} \operatorname{tr}(QM), \tag{13}$$

where $M$ satisfies

$$M = e^{-\lambda} A^\top M A' + W. \tag{14}$$

This special kernel was first derived in Smola and Vishwanathan (2003).

***5.1.2. Determinant Kernels on ARMA Models.*** As before, we consider an exponential discounting $\mu(t) = e^{-\lambda t}$, and obtain the following expression for the determinant kernel on ARMA models:

$$k((x_0, A, C), (x_0', A', C')) := \mathbb{E}_{v,w} \det \left[ \sum_{t=0}^{\infty} e^{-\lambda t} y_t y_t'^\top \right]. \tag{15}$$

Notice that in this case the effect of the weight matrix $W$ is just a scaling of the kernel value by $\det(W)$, thus, we assume w.l.o.g. that $W = \mathbf{I}$.

Also, for the sake of simplicity and in order to compare with other kernels we assume an ARMA model with no noise, i.e., $v_t = 0$ and $w_t = 0$. Then, we have the following:

**Theorem 16.** *If $e^{-\lambda}\|A\|\|A'\| < 1$, then the kernel of* (15) *is given by*

$$k((x_0, A, C), (x_0', A', C')) = \det CMC'^\top, \tag{16}$$

*where $M$ satisfies*

$$M = e^{-\lambda} AMA'^\top + x_0 x_0'^\top.$$

**Proof:** We have

$$\begin{aligned} &k((x_0, A, C), (x_0', A', C')) \\ &= \det \sum_{t=0}^{\infty} e^{-\lambda t} C A^t x_0 x_0'^\top (A'^t)^\top C'^\top \\ &= \det CMC'^\top, \end{aligned}$$

where, by using Lemma 18, we can write

$$\begin{aligned} M &= \sum_{t=0}^{\infty} e^{-\lambda t} A^t x_0 x_0'^\top (A'^t)^\top \\ &= e^{-\lambda} AMA'^\top + x_0 x_0'^\top. \end{aligned}$$

□

Note that the kernel is uniformly zero if $C$ or $M$ do not have full rank. In particular, this kernel is zero if the dimension of the latent variable space $n$ is smaller than the dimension of the observed space $m$, i.e., the matrix $C \in \mathbb{R}^{m \times n}$ is rectangular. This indicates that the determinant kernel might have limited applicability in practical situations.

As pointed out in Wolf and Shashua (2003), the determinant is *not* invariant to permutations of the columns of $\text{Traj}_{\mathbf{A}}(x)$ and $\text{Traj}_{\mathbf{A}'}(x')$. Since different columns or linear combinations of them are determined by the choice of the initial conditions $x_0$ and $x_0'$, this means, as is obvious from the formula for $M$, that the determinant kernel does depend on the initial conditions.

In order to make it independent from initial conditions, as before, we can take expectations over $x_0$ and $x_0'$. Unfortunately, although $M$ is linear in on both $x_0$ and $x_0'$, the kernel given by (16) is multi-linear. Therefore, the kernel depends not only on the covariance $\Sigma$ on the initial conditions, but also on higher order statistics. Only in the case of single-output systems, we obtain a simple expression

$$k((A, C), (A', C')) = CMC'^\top,$$

where

$$M = e^{-\lambda} AMA'^\top + \Sigma,$$

for the kernel on the model parameters only.

### 5.2. *Continuous Time Models*

The time evolution of a continuous time LTI system $(x_0, A)$ can be modeled as

$$\frac{d}{dt} x(t) = Ax(t).$$

It is well known (cf. Chapter 2 Luenberger, 1979) that for a continuous LTI system, $x(t)$ is given by

$$x(t) = \exp(At)x(0),$$

where $x(0)$ denotes the initial conditions.

Given two LTI systems $(x_0, A)$ and $(x_0', A')$, a positive semi-definite matrix $W$, and a measure $\mu(t)$ we

can define the trace kernel on the trajectories as

$$k((x_0, A), (x_0', A')) := x_0^\top \left[\int_0^\infty \exp(At)^\top W \exp(A't)\, d\mu(t)\right] x_0'. \tag{17}$$

As before, we need to assume a suitably decaying measure in order to ensure that the above integral converges. The following theorem characterizes the solution when $\mu(t) = e^{-\lambda t}$, i.e., the measure is exponentially decaying.

**Theorem 17.** *Let, $A, A' \in \mathbb{R}^{m\times m}$, and $W \in \mathbb{R}^{m\times m}$ be non-singular. Furthermore, assume that $\|A\|, \|A'\| \leq \Lambda$, $||W||$ is bounded, and $\mu(t) = e^{-\lambda t}$. Then, for all $\lambda > 2\Lambda$ (17) is well defined and can be computed as*

$$k((x_0, A), (x_0', A')) = x_0^\top M x_0,$$

*where*

$$\left(A - \frac{\lambda}{2}\mathbf{I}\right)^\top M + M\left(A' - \frac{\lambda}{2}\mathbf{I}\right) = -W.$$

*Furthermore, if $\mu(t) = \delta_\tau(t)$ we have the somewhat trivial kernel*

$$k((A, x_0), (A', x_0')) = x_0^\top [\exp(A\tau)^\top W \exp(A\tau)] x_0'. \tag{18}$$

**Proof:** See Appendix B. □

### 5.3. *Efficient Computation*

Before we describe strategies for efficiently computing the kernels discussed above we need to introduce some notation. Given a matrix $A \in \mathbb{R}^{n\times m}$, we use $A_{:,i}$ to denote the $i$-th column of $A$, $A_{i,:}$ to denote the $i$-th row and $A_{i,j}$ to denote the $(i, j)$-th element of $A$. The linear operator vec : $\mathbb{R}^{n\times m} \to \mathbb{R}^{nm}$ flattens the matrix (Bernstein, 2005). In other words,

$$\operatorname{vec}(A) := \begin{bmatrix} A_{:,1} \\ A_{:,2} \\ \vdots \\ A_{:,n} \end{bmatrix}.$$

Given two matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{l\times k}$ the Kronecker product $A \otimes B \in \mathbb{R}^{nl\times mk}$ is defined as Bernstein (2005):

$$A \otimes B := \begin{bmatrix} A_{1,1}B & A_{1,2}B & \dots & A_{1,m}B \\ \vdots & \vdots & \vdots & \vdots \\ A_{n,1}B & A_{n,2}B & \dots & A_{n,m}B \end{bmatrix}.$$

Observe that in order to compute the kernels on linear dynamical systems we need to solve for $M \in \mathbb{R}^{n\times n}$ given by

$$M = SMT + U, \tag{19}$$

where $S, T, U \in \mathbb{R}^{n\times n}$ are given. Equations of this form are known as Sylvester equations, and can be readily solved in $O(n^3)$ time with freely available code (Gardiner et al., 1992).

In some cases, for instance when we are working with adjacency matrices of graphs, the matrices $S$ and $T$ might be sparse or rank deficient. In such cases, we might be able to solve the Sylvester equations more efficiently. Towards this end we rewrite (19) as

$$\operatorname{vec}(M) = \operatorname{vec}(SMT) + \operatorname{vec}(U). \tag{20}$$

Using the well known identity see proposition 7.1.9 (Bernstein, 2005)

$$\operatorname{vec}(SMT) = (T^\top \otimes S)\operatorname{vec}(M), \tag{21}$$

we can rewrite (20) as

$$(\mathbf{I} - T^\top \otimes S)\operatorname{vec}(M) = \operatorname{vec}(U), \tag{22}$$

where $\mathbf{I}$ denotes the identity matrix of appropriate dimensions. In general, solving the $n^2 \times n^2$ system

$$\operatorname{vec}(M) = (\mathbf{I} - T^\top \otimes S)^{-1}\operatorname{vec}(U), \tag{23}$$

might be rather expensive. But, if $S$ and $T$ are sparse then by exploiting sparsity, and using (21) we can compute $(\mathbf{I} - T^\top \otimes S)\operatorname{vec} X$ for any matrix $X$ rather cheaply. This naturally suggests two strategies: We can iteratively solve for a fixed point of (23) as is done by Kashima et al. (2004). But the main drawback of this approach is that the fixed point iteration might take a long time to converge. On the other hand, $\operatorname{rank}(T^\top \otimes S) = \operatorname{rank}(T)\operatorname{rank}(S)$, and hence one can

employ a Conjugate Gradient (CG) solver which can exploit both the sparsity of as well as the low rank of $S$ and $T$ and provide guaranteed convergence (Nocedal and Wright, 1999). Let $\{\lambda_i\}$ denote the eigenvalues of $S$ and $\{\mu_j\}$ denote the eigenvalues of $T$. Then the eigenvalues of $S \otimes T$ are given by $\{\lambda_i \mu_j\}$ (Bernstein, 2005). Therefore, if the eigenvalues of $S$ and $T$ are bounded (as they are in the case of a graph) and rapidly decaying then the eigenvalues of $S \otimes T$ are bounded and rapidly decaying. In this case a CG solver will be able to efficiently solve (23). It is worthwhile reiterating that both the above approaches are useful only when $S$ and $T$ are sparse and have low (effective) rank. For the general case, solving the Sylvester equation directly is the recommended approach. More details about these efficient computational schemes can be found in Vishwanathan et al. (2006).

Computing $A^\top C^\top W C' A'$ and $C^\top W C'$ in (12) requires $O(nm^2)$ time where $n$ is the latent variable dimension and $n \ll m$ is the observed variable dimension. Computation of $WR$ requires $O(m^3)$ time, while $Q\tilde{M}$ can be computed in $O(n^3)$ time. Assuming that the Sylvester equation can be solved in $O(n^3)$ time the overall time complexity of computing (12) is $O(nm^2)$. Similarly, the time complexity of computing (11) is $O(m^3)$.

## 6. Connections with Existing Kernels

In this section we will explore connections between Binet-Cauchy kernels and many existing kernels including set kernels based on cepstrum coefficients, marginalized graph kernels, and graph kernels based on diffusion. First we briefly describe set kernels based on subspace angles of observability subspaces, and show that they are related to our determinant kernels. Next we concentrate on labeled graphs and show that the marginalized graph kernels can be interpreted as a special case of our kernels on ARMA models. We then turn our attention to diffusion on graphs and show that our kernels on continuous time models yields many graph kernels as special cases. Our discussion below points to deeper connections between disparate looking kernels, and also demonstrates the utility of our approach.

### 6.1. Kernel via Subspace Angles and Martin Kernel

Given an ARMA model with parameters $(A, C)$ (9), if we neglect the noise terms $w_t$ and $v_t$, then the Hankel matrix $\mathcal{Z}$ of the output satisfies

$$\mathcal{Z} = \begin{bmatrix} y_0 & y_1 & y_2 & \cdots \\ y_1 & y_2 & \cdots & \\ y_2 & \vdots & \ddots & \\ \vdots & & & \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix} \begin{bmatrix} x_0 & x_1 & x_3 & \cdots \end{bmatrix}$$
$$= \mathcal{O} \begin{bmatrix} x_0 & x_1 & x_3 & \cdots \end{bmatrix}.$$

Since $A \in \mathbb{R}^{n \times n}$ and $C \in \mathbb{R}^{m \times n}$, $CA \in \mathbb{R}^{m \times n}$. Therefore, the infinite output sequence $y_t$ lives in an $m$-dimensional *observability subspace* spanned by the $m$ columns of the extended *observability matrix* $\mathcal{O}$. De Cock and De Moor (2002) propose to compare ARMA models by using the subspace angles among their observability subspaces. More specifically, a kernel between ARMA models $(A, C)$ and $(A', C')$ is defined as

$$k((A, C), (A', C')) = \prod_{i=1}^{m} \cos^2(\theta_i), \tag{24}$$

where $\theta_i$ is the $i$-th subspace angle between the column spaces of the extended observability matrices $\mathcal{O} = [C^\top A^\top C^\top \cdots]^\top$ and $\mathcal{O}' = [C'^\top A'^\top C'^\top \cdots]^\top$. Since the subspace angles should be independent of the choice of a particular basis for $\mathcal{O}$ or $\mathcal{O}'$, the calculation is typically done by choosing a canonical basis via the QR decomposition. More specifically, the above kernel is computed as $\det(Q^\top Q')^2$, where $\mathcal{O} = QR$ and $\mathcal{O}' = Q'R'$ are the $QR$ decompositions of $\mathcal{O}$ and $\mathcal{O}'$, respectively. [4] Therefore, the kernel based on subspace angles is essentially the square of a determinant kernel [5] formed from a whitened version of the outputs (via the QR decomposition) rather than from the outputs directly, as with the determinant kernel in (16).

Yet another way of defining a kernel on dynamical systems is via cepstrum coefficients, as proposed by Martin (2000). More specifically, if $H(z)$ is the transfer function of the ARMA model described in (9), then the cepstrum coefficients $c_n$ are defined via the Laplace transformation of the logarithm of the ARMA power spectrum,

$$\log H(z) H^*(z^{-1}) = \sum_{n \in \mathbb{Z}} c_n z^{-n}. \tag{25}$$

The Martin kernel between $(A, C)$ and $(A', C')$ with corresponding cepstrum coefficients $c_n$ and $c'_n$ is

defined as

$$k((A, C), (A', C')) := \sum_{n=1}^{\infty} n c_n^* c_n'. \tag{26}$$

As it turns out, the Martin kernel and the kernel based on subspace angles are the same as shown by De Cock and De Moor (2002).

It is worth noticing that, by definition, the Martin kernel and the kernel based on subspace angles depend exclusively on the model parameters $(A, C)$ and $(A', C')$, and not on the initial conditions. This could be a drawback in practical applications where the choice of the initial condition is important when determining how similar two dynamical systems are, e.g. in modeling human gaits (Saisan et al., 2001).

### 6.2. *ARMA Kernels on Graphs*

We begin with some elementary definitions: We use $\mathbf{e}$ to denote a vector with all entries set to one, $\mathbf{e}_i$ to denote the $i$-th standard basis, i.e., a vector of all zeros with the $i$-th entry set to one, while $\mathbf{E}$ is used to denote a matrix with all entries set to one. When it is clear from context we will not mention the dimensions of these vectors and matrices.

A graph $G$ consists of an ordered and finite set of vertices $V$ denoted by $\{v_1, v_2, \ldots, v_n\}$, and a finite set of edges $E \subset V \times V$. We use $|V|$ to denote the number of vertices, and $|E|$ to denote the number of edges. A vertex $v_i$ is said to be a neighbor of another vertex $v_j$ if they are connected by an edge. $G$ is said to be undirected if $(v_i, v_j) \in E \iff (v_j, v_i) \in E$ for all edges. The unnormalized adjacency matrix of $G$ is an $n \times n$ real matrix $P$ with $P(i, j) = 1$ if $(v_i, v_j) \in E$, and 0 otherwise. If $G$ is weighted then $P$ can contain non-negative entries other than zeros and ones, i.e., $P(i, j) \in (0, \infty)$ if $(v_i, v_j) \in E$ and zero otherwise.

Let $D$ be an $n \times n$ diagonal matrix with entries $D_{ii} = \sum_j P(i, j)$. The matrix $A := PD^{-1}$ is then called the normalized adjacency matrix, or simply adjacency matrix. The adjacency matrix of an undirected graph is symmetric. Below we will always work with undirected graphs. The Laplacian of $G$ is defined as $L := D - P$ and the Normalized Laplacian is $\tilde{L} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. The normalized graph Laplacian often plays an important role in graph segmentation (Shi and Malik, 1997).

A walk $w$ on $G$ is a sequence of indices $w_1, w_2, \ldots w_{t+1}$ where $(v_{w_i}, v_{w_{i+1}}) \in E$ for all $1 \leq i \leq t$. The length of a walk is equal to the number of edges encountered during the walk (here: $t$). A graph is said to be connected if any two pairs of vertices can be connected by a walk; we always work with connected graphs. A random walk is a walk where $\mathbb{P}(w_{i+1}|w_1, \ldots w_i) = A(w_i, w_{i+1})$, i.e., the probability at $w_i$ of picking $w_{i+1}$ next is directly proportional to the weight of the edge $(v_{w_i}, v_{w_{i+1}})$. The $t$-th power of the transition matrix $A$ describes the probability of $t$-length walks. In other words, $A^t(i, j)$ denotes the probability of a transition from vertex $v_i$ to vertex $v_j$ via a walk of length $t$.

Let $x_0$ denote an initial probability distribution over the set of vertices $V$. The probability distribution $x_t$ at time $t$ obtained from a random walk on the graph $G$, can be modeled as $x_t = A^t x_0$. The $j$-th component of $x_t$ denotes the probability of finishing a $t$-length walk at vertex $v_j$. In the case of a labeled graph, each vertex $v \in V$ is associated with an unique label from the set $\{l_1, l_2, \ldots, l_m\}$.[6] The label transformation matrix $C$ is a $m \times n$ matrix with $C_{ij} = 1$ if label $l_i$ is associated with vertex $v_j$, and 0 otherwise. The probability distribution $x_t$ over vertices is transformed into a probability distribution $y_t$ over labels via $y_t = Cx_t$. As before, we assume that there exists a matrix $W \in \mathbb{R}^{m \times m}$ which can be used to define the kernel $\kappa(y_t, y_t') = y_t^\top W y_t'$ on the space of all labels.

***6.2.1. Random Walk Graph Kernels*** The time evolution of the label sequence associated with a random walk on a graph can be expressed as the following ARMA model

$$\begin{aligned} y_t &= Cx_t \\ x_{t+1} &= Ax_t \end{aligned} \tag{27}$$

It is easy to see that (27) is a very special case of (9), i.e., it is an ARMA model with no noise.

Given two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, a decay factor $\lambda$, and initial distributions $x_0 = \frac{1}{|V_1|}\mathbf{e}$ and $x_0' = \frac{1}{|V_2|}\mathbf{e}$, using Corollary 15 we can define the following kernel on labeled graphs which compares random walks:

$$k(G_1, G_2) = \frac{1}{|V_1||V_2|}\mathbf{e}^\top M \mathbf{e}$$

where

$$M = e^{-\lambda} A_1^\top M A_2 + C_1^\top W C_2.$$

We now show that this is essentially the geometric graph kernel of Gärtner et al. (2003). Towards this end, applying vec and using (21), we write

$$\begin{aligned} k(G_1, G_2) &= \frac{1}{|V_1||V_2|} \operatorname{vec}(\mathbf{e}^\top \operatorname{vec}(M)\, \mathbf{e}) \\ &= \frac{1}{|V_1||V_2|} \mathbf{e}^\top \operatorname{vec}(M). \end{aligned} \tag{28}$$

Next, we set $C_1 = C_2 = \mathbf{I}$, $W = \mathbf{e}\mathbf{e}^\top$, and $e^{-\lambda} = \lambda'$. Now we have

$$M = \lambda' A_1^\top M A_2 + \mathbf{e}\mathbf{e}^\top.$$

Applying vec on both sides and using (21) we can rewrite the above equation as

$$\begin{aligned} \operatorname{vec}(M) &= \lambda' \operatorname{vec}(A_1^\top M A_2) + \operatorname{vec}(\mathbf{e}\mathbf{e}^\top) \\ &= \lambda'(A_2^\top \otimes A_1^\top) \operatorname{vec}(M) + \mathbf{e}. \end{aligned}$$

The matrix $A_\times := A_2^\top \otimes A_1$ is the adjacency matrix of the product graph (Gärtner et al., 2003). Recall that we always deal with undirected graphs, i.e., $A_1$ and $A_2$ are symmetric. Therefore, the product graph is also undirected and its adjacency matrix $A_\times$ is symmetric. Using this property, and by rearranging terms, we obtain from the above equation

$$\operatorname{vec}(M) = (\mathbf{I} - \lambda' A_\times)^{-1} \mathbf{e}.$$

Plugging this back into (28) we obtain

$$k(G_1, G_2) = \frac{1}{|V_1||V_2|} \mathbf{e}^\top (\mathbf{I} - \lambda' A_\times)^{-1} \mathbf{e}.$$

This, modulo the normalization constant, is exactly the geometric kernel defined by Gärtner et al. (2003).

The marginalized graph kernels between labeled graphs (Kashima et al., 2004) is also essentially equivalent to the kernels on ARMA models. Two main changes are required to derive these kernels: A stopping probability, i.e., the probability that a random walk ends at a given vertex, is assigned to each vertex in both the graphs. The adjacency matrix of the product graph is no longer a Kronecker product. It is defined by kernels between vertex labels. This can be viewed as an extension of Kronecker products to incorporate kernels. More details about these extensions, and details about the relation between geometric kernels and marginalized graph kernels can be found in Vishwanathan et al. (2006).

***6.2.2. Diffusion Kernels*** If we assume a continuous diffusion process on graph $G(V, E)$ with graph Laplacian $L$ the evolution of $x_t$ can be modeled as the following continuous LTI system

$$\frac{d}{dt} x(t) = L x(t).$$

We begin by studying a very simple case. Set $A = A' = L$, $W = \mathbf{I}$, $x_0 = e_i$, and $x_0' = e_j$ in (18) to obtain

$$k((L, e_i), (L, e_j)) = [\exp(L\tau)^\top \exp(L\tau)]_{ij}.$$

Observe that this kernel measures the probability that any other node $v_l$ could have been reached jointly from $v_i$ and $v_j$ (Kondor and Lafferty, 2002). In other words, it measures the probability of diffusing from either node $v_i$ to node $v_j$ or vice versa. The $n \times n$ kernel matrix $K$ can be written as

$$K = \exp(L\tau)^\top \exp(L\tau),$$

and viewed as a covariance matrix between the distributions over vertices. In the case of a undirected graph, the graph Laplacian is a symmetric matrix, and hence the above kernel can be written as

$$K = \exp(2L\tau).$$

This is exactly the diffusion kernel on graphs proposed by Kondor and Lafferty (2002).

### *6.3. Inducing Feature Spaces*

Burkhardt (2004) uses features on graphs to derive invariants for the comparison of polygons. More specifically, they pick a set of feature functions, denoted by the vector $\Phi(x, p)$, defined on $x$ with respect to the polygon $p$ and they compute their value on the entire trajectory through the polygon. Here $x$ is an index on the vertex of the polygon and the dynamical system simply performs the operation $x \to (x \bmod n) + 1$, where $n$ is the number of vertices of the polygon.

Essentially, what happens is that one maps the polygon into the trajectory $(\Phi(1, p), \ldots, \Phi(n, p))$. For a

fixed polygon, this already would allow us to compare vertices based on their trajectory. In order to obtain a method for comparing different polygons, one needs to rid oneself of the dependency on initial conditions: the first point is arbitrary. To do so, one simply assumes a uniform distribution over the pairs $(x, x')$ of initial conditions. This distribution satisfies the conditions of de Finetti's theorem and we can therefore compute the kernel between two polygons via

$$k(p, p') = \frac{1}{nn'} \sum_{x,x'=1}^{n,n'} \langle \Phi(x, p), \Phi(x', p') \rangle. \quad (29)$$

Note that in this context the assumption of a uniform distribution amounts to computing the Haar integral over the cyclical group defined by the vertices of the polygon.

The key difference to Burkhardt (2004) and (29) is that in our framework one is not limited to a small set of polynomials which need to be constructed explicitly. Instead, one can use any kernel function $k((x, p), (x', p'))$ for this purpose.

This is but a small example of how rich features on the states of a dynamical system can be used in the comparison of trajectories. What should be clear, though, is that a large number of the efficient computations has to be sacrificed in this case. Nonetheless, for discrete measures $\mu$ with a finite number of nonzero steps this can be an attractive alternative to a manual search for useful features.

## 7. Extension to Nonlinear Dynamical Systems

In this section we concentrate on non-linear dynamical systems whose time evolution is governed by

$$x_{t+1} = f(x_t). \quad (30)$$

The essential idea here is to map the state space of the dynamical system into a suitable higher dimensional *feature* space such that we obtain a linear system in this new space. We can now apply the results we obtained in Section 5 for linear models. Two technical difficulties arise. First, one needs efficient algorithms for computing the mapping. Second, we have a limited number of state space observations at our disposal, while the mapped dynamical system evolves in a rather high dimensional space. Therefore, to obtain meaningful results, one needs to restrict the solution further. We discuss methods to overcome both these difficulties below.

### 7.1. Linear in Feature Space

Given a non-linear dynamical system $f$, whose state space is $\mathcal{X}$, we seek a bijective transformation $\Phi : \mathcal{X} \mapsto \mathcal{H}$ and a linear operator $A$ such that in the new coordinates $z_t := \Phi(x_t)$ we obtain a linear system

$$z_{t+1} = A z_t. \quad (31)$$

Furthermore, we assume that $\mathcal{H}$ is a RKHS endowed with a kernel $\kappa(\cdot, \cdot)$. If such a $\phi$ and $A$ exist then we can define a kernel on the nonlinear models $f$ and $f'$ as

$$k_{\text{nonlinear}}((x_0, f), (x_0', f')) = k_{\text{linear}}((z_0, A), (z_0', A'))$$

where $k_{\text{linear}}$ is any of the kernels for linear models defined in the previous sections.

The above construction can be immediately applied whenever $f$ and $f'$ are feedback-linearizable. Conditions for $f$ to be feedback-linearizable as well as an algorithm for computing $\Phi$ and $A$ from $f$ can be found in Isidori (1989).

However, a given $f$ is in general not feedback-linearizable. In such cases, we propose to find a ARMA model whose time evolution is described by

$$\Phi'(y_t) = C\Phi(x_t) + w_t$$
$$\Phi(x_{t+1}) = A\Phi(x_t) + v_t.$$

Here $\Phi$ and $\Phi'$ are appropriate non-linear transforms, and as before $w_t$ and $v_t$ are IID zero mean Gaussian random variables. In case the model is fully observable the time evolution is governed by

$$\Phi(x_{t+1}) = A\Phi(x_t) + v_t. \quad (32)$$

In the following, we only deal with the fully observable case. We define kernels on the original dynamical system by simply studying the time evolution of $\Phi(x_t)$ instead of $x_t$.

Unfortunately, such a transformation need not always exist. Moreover, for the purpose of comparing trajectories it is not necessary that the map $\Phi$ be bijective. In fact, injectivity is all we need. This means that as long as we can find a linear system such that (32) holds we can extend the tools of the previous sections

to nonlinear models. In essence this is what was proposed in Ralaivola and d'Alché Buc (2003) and Bakir, et al. (2003) in the context of time-series prediction.

The problem with (32) is that once we spell it out in feature space using $\Phi$ the matrix $A$ turns into an operator. However, we have only a finite number of observations at our disposition. Therefore, we need to impose further restrictions on the operators for a useful result.

### 7.2. Solution by Restriction

We impose the restriction that the image of $A$ be contained in the span of the $\Phi(x_i)$. This means that we can find an equivalent condition to (32) via

$$\begin{aligned}\kappa(x_i, x_{t+1}) &:= \langle \Phi(x_i), \Phi(x_{t+1})\rangle_{\mathcal{H}} \\ &= \underbrace{\langle \Phi(x_i), A\Phi(x_t)\rangle_{\mathcal{H}}}_{:=\tilde{A}_{it}} + \langle \Phi(x_i), v_t\rangle_{\mathcal{H}}. \quad (33)\end{aligned}$$

For a perfect solution we "only" need to find an operator $A$ for which

$$KT_+ = \tilde{A}, \quad (34)$$

where $T_+ \in \mathbb{R}^{m\times m-1}$ is the shift operator, that is, $(T_+)_{ij} = \delta_{i,j-1}$. Moreover $K$ is the kernel matrix $\kappa(x_i, x_j)$. For a large number of kernel matrices $K$ with full rank, such a solution always exists regardless of the dynamics, which leads to overfitting problems. Consequently we need to restrict $A$ further.

A computationally attractive option is to restrict the rank of $A$ further so that

$$A := \sum_{i,j\in S} \alpha_{ij}\Phi(x_i)\Phi(x_j)^\top \quad (35)$$

for some subset $S$. We choose the same basis in both terms to ensure that the image of $A$ and of its adjoint operator $A^\top$ lie in the same subspace. We define the matrix $\tilde{K} \in \mathbb{R}^{m\times|S|}$ as $\tilde{K}_{ij} = \kappa(x_i, x_j)$ (where $j \in S$). Now we can rewrite (34) in order to obtain the following equivalent condition:

$$KT_+ = \tilde{K}\alpha\tilde{K}T_-. \quad (36)$$

Here $T_- \in \mathbb{R}^{m\times m-1}$ is the inverse shift operator, that is $(T_-)_{ij} = \delta_{ij}$. One option to solve (36) is to use pseudo-inverses. This yields

$$\alpha = \tilde{K}^\dagger(KT_+)(\tilde{K}T_-)^\dagger. \quad (37)$$

### 7.3. Sylvester Equations in Feature Space

Finally, one needs to solve the Sylvester equations, or QR factorizations of determinants in feature space. We will only deal with the Sylvester equation below. Using derivations in Wolf and Shashua (2003) it is easy to obtain analogous expressions for the latter cases.

We begin with (14). Without loss of generality we assume that $W = \mathbf{I}$ (other cases can be easily incorporated into the scalar product directly, hence we omit them). Moreover we assume that $A$, $A'$ have been expanded using the same basis $\Phi(x_j)$ with $j \in S$.

The RHS of (14) has the same expansion as $A$, hence we can identify $M$ uniquely by observing the action of $M$ on the subspace spanned by $\Phi(x_i)$. Using $M = \sum_{i,j\in S}\eta_{ij}\Phi(x_i)\Phi(x_j)^\top$ and $\bar{K}_{ij} := k(x_i, x_j)$ we obtain

$$\begin{aligned}\Phi(x_i)^\top M\Phi(x_j) &= [\bar{K}\eta\bar{K}]_{ij} \\ &= e^{-\lambda}[\bar{K}\alpha^\top\bar{K}\alpha'\bar{K}]_{ij} \\ &\quad + e^{-\lambda}[\bar{K}\alpha^\top\bar{K}\eta\bar{K}\alpha'\bar{K}]_{ij} \quad (38)\end{aligned}$$

Assuming that $\tilde{K}$ has full rank (which is a reasonable assumption for a set of vectors used to span the image of an operator), we can eliminate $\bar{K}$ on both sides of the equation and we have the following new Sylvester equation:

$$\eta = e^{-\lambda}\alpha^\top\bar{K}\alpha' + e^{-\lambda}\alpha^\top\bar{K}\eta\bar{K}\alpha'. \quad (39)$$

Finally, computing traces over finite rank operators can be done simply by noting that

$$\begin{aligned}\operatorname{tr}\sum_{i,j}\alpha_{ij}\Phi(x_i)\Phi(x_j)^\top &= \operatorname{tr}\sum_{i,j}\alpha_{ij}\Phi(x_j)^\top\Phi(x_i) \\ &= \operatorname{tr}\tilde{K}\alpha. \quad (40)\end{aligned}$$

## 8. Application to the Analysis of Dynamic Scenes

Applications of kernel methods to computer vision have so far been largely restricted to methods which analyze a single image at a time, possibly with some

post-processing to take advantage of the fact that images are ordered. Essentially there are two main approaches (Heisele et al., 2001): kernels which operate directly on the raw pixel data (Schölkopf, 1997; Blanz et al., 1996), such as the Haussdorff kernel which matches similar pixels in a neighborhood (Barla et al., 2002), and kernels which exploit the statistics of a small set of features on the image, such as intensity, texture, and color histograms (Chapelle et al., 1999). However, when dealing with video sequences, such similarity measures are highly inadequate as they do not take the temporal structure of the data into account.

The work of Doretto et al. (2003) and Soatto et al. (2001) points out a means of comparing dynamic textures by first approximating them with an ARMA model, and subsequently computing the subspace angles between the observability subspaces of such models. In this section, we present experiments showing that the proposed Binet-Cauchy kernels outperform the subspace angle based approach when used for comparing dynamic textures. In addition, we show that our kernels can be effectively used for clustering video shots in dynamic scenes.

### 8.1. Setup

We restrict our attention to video sequence of dynamic scenes whose temporal evolution can be modeled as the output of a linear dynamical model. This assumption has proven to be very reasonable when applied to a large number of common phenomena such as smoke, fire, water flow, foliage, wind etc., as corroborated by the impressive results of Doretto et al. (2003).

More specifically, let $I_t \in \mathbb{R}^m$ for $t = 1, \ldots, \tau$ be a sequence of $\tau$ images, and assume that at every time step we measure a noisy version $y_t = I_t + w_t$ where $w_t \in \mathbb{R}^m$ is Gaussian noise distributed as $\mathcal{N}(0, R)$. To model the sequences of observed images as a Gaussian ARMA model, we will assume that there exists a positive integer $n \ll m$, a process $x_t \in \mathbb{R}^n$ and symmetric positive definite matrices $Q \in \mathbb{R}^{n \times n}$ and $R \in \mathbb{R}^{m \times m}$ such that (9) holds. Without loss of generality, the scaling of the model is fixed by requiring that

$$C^\top C = \mathbf{I}.$$

Given a sequence of $\tau$ noisy images $\{y_t\}$, the identification problem is to estimate $A, C, Q$ and $R$. Optimal solutions in the maximum likelihood sense exist, e.g. the `n4sid` method in systems identification toolbox of MATLAB, but they are very expensive to compute for large images. Instead, following Doretto et al. (2003), we use a sub-optimal closed form solution. The advantage of this approach is twofold: (a) It is computationally efficient and (b) it allows us to compare our results with previous work.

Set $Y := [y_1, \ldots, y_\tau]$, $X := [x_1, \ldots, x_\tau]$, and $W := [w_1, \ldots, w_\tau]$, and solve

$$\min ||Y - CX||_F = \min ||W||_F,$$

where $C \in \mathbb{R}^{m \times n}$, $C^\top C = \mathbf{I}$, and $||\cdot||_F$ is used to denote the Frobenius norm. The unique solution to the above problem is given by $C = U_n$ and $X = \Sigma_n V_n^\top$ where $U_n \Sigma_n V_n^\top$ is the best rank $n$ approximation of $Y$. $U_n$, $\Sigma_n$ and $V_n$ can be estimated in a straightforward manner from $Y = U \Sigma V^\top$, the Singular Value Decomposition (SVD) of $Y$ (Golub and Van Loan, 1996).

In order to estimate $A$ we solve $\min ||AX - X'||_F$ where $X' = [x_2, \ldots, x_\tau]$, which again has a closed form solution using SVD. Now, we can compute $v_i = x_i - Ax_{i-1}$ and set

$$Q = \frac{1}{\tau - 1} \sum_{i=1}^{\tau} v_i v_i^\top.$$

The covariance matrix $R$ can also be computed from the columns of $W$ in a similar manner. For more information including details of efficient implementation we refer the interested reader to Doretto et al. (2003).

### 8.2. Parameters

The parameters of the ARMA model deserve some consideration. The initial conditions of a video clip are given by the vector $x_0$. This vector helps in distinguishing scenes which have a different stationary background, but the same dynamical texture in the foreground.[7] In this case, if we use identical initial conditions for both systems, the similarity measure will focus on the dynamical part (the foreground). On the other hand, if we make use of $x_0$, we will be able to distinguish between scenes which only differ in their background.

Another important factor which influences the value of the kernel is the value of $\lambda$. If we want to provide more weightage to short range interactions due to differences in initial conditions it might be desirable to use a large values of $\lambda$ since it results in heavy attenuation of contributions as time $t$ increases. On the other

hand, when we want to identify samples of the same dynamical system it might be desirable to use small values of $\lambda$.

Finally, $A, C$ determine the dynamics. Note that there is no requirement that $A$, $A'$ share the same dimensionality. Indeed, the only condition on the two systems is that the spaces of observations $y_t$, $y'_t$ agree. This allows us, for instance, to determine the approximation qualities of systems with various levels of detail in the parametrization.

### *8.3. Comparing Video Sequences of Dynamic Textures*

In our experiments we used some sequences from the *MIT temporal texture database*. We enhanced this dataset by capturing video clips of dynamic textures of natural and artificial objects, such as trees, water bodies, water flow in kitchen sinks, etc. Each sequence in this combined database consists of 120 frames. For compatibility with the MIT temporal texture database we downsample the images to use a grayscale colormap (256 levels) with each frame of size $115 \times 170$. Also, to study the effect of temporal structure on our kernel, we number each clip based on the sequence in which it was framed. In other words, our numbering scheme implies that clip $i$ was filmed before clip $i+1$. This also ensures that clips from the same scene get labeled with consecutive numbers. Figure 1 shows some samples from our dataset.

We used the procedure outlined in Section 8.1 for estimating the model parameters $A$, $C$, $Q$, and $R$. Once the system parameters are estimated we compute distances between models using our trace kernel (11) as well as the Martin distance (26). We varied the value of the down-weighting parameter $\lambda$ and report results for two values $\lambda = 0.1$ and $\lambda = 0.9$. The distance matrix obtained using each of the above methods is shown in Fig. 2. Darker colors imply smaller distance value, i.e., two clips are more related according to the value of the metric induced by the kernel.

As mentioned before, clips that are closer to each other on an axis are closely related, that is, they are either from similar natural phenomena or are extracted from the same master clip. Hence a perfect distance measure will produce a block diagonal matrix with a high degree of correlation between neighboring clips. As can be seen from our plots, the kernel using trajectories assigns a high similarity to clips extracted from the same master clip, while the Martin distance fails to do so. Another interesting feature of our approach is that the value of the kernel seems to be fairly independent

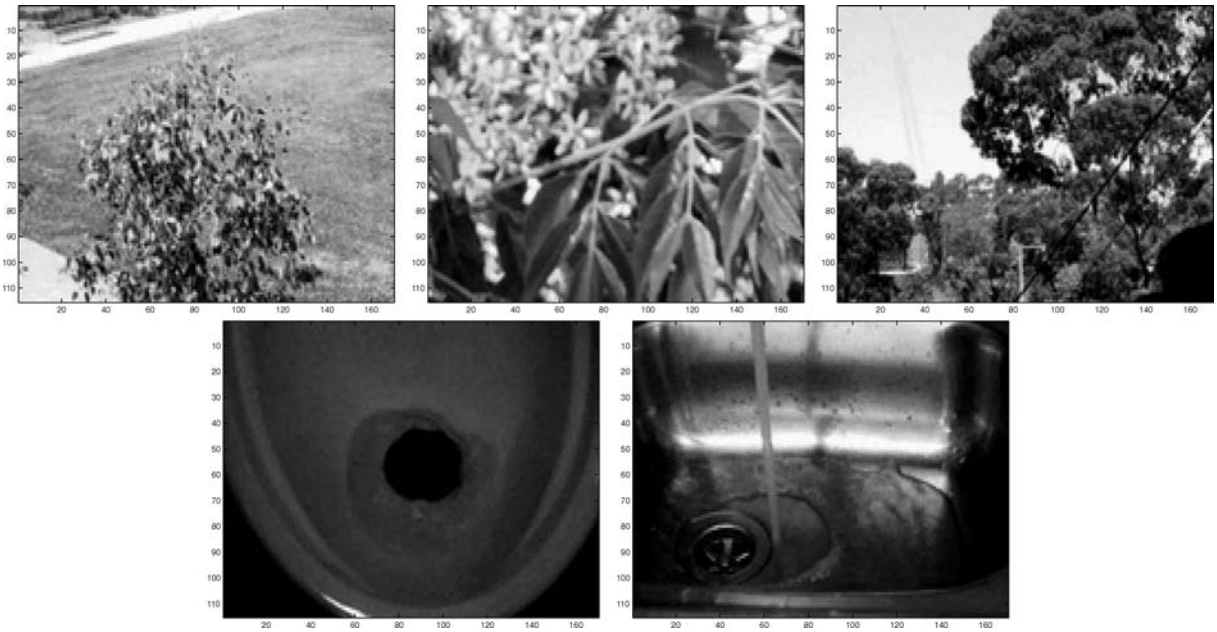

*Figure 1.* Some sample textures from our dataset.

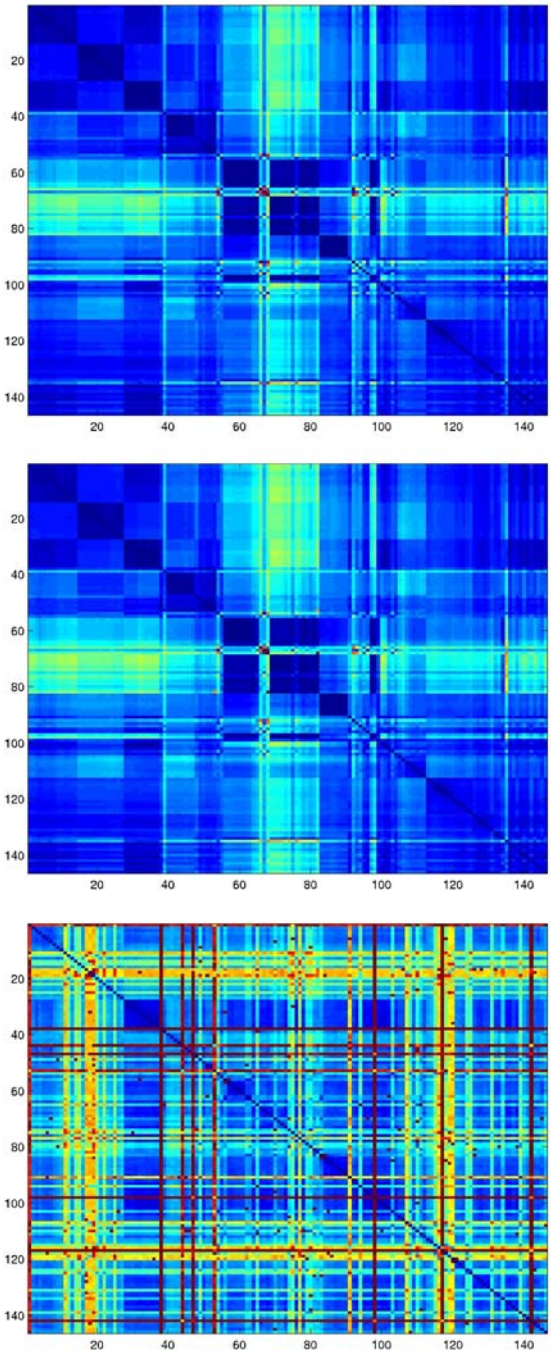


*Figure 2.* Distance matrices obtained using the trace kernel (top two plots) and the Martin kernel (bottom plot). We used a value of $\lambda = 0.9$ for the first plot and a value of $\lambda = 0.1$ for the second plot. The matrix $W = \mathbf{1}$ was used for both plots. Clips which are closer to each other on an axis are closely related.

of $\lambda$. The reason for this might be because we consider long range interactions averaged out over infinite time steps. Two dynamic textures derived from the same source might exhibit very different short term behavior due to the differences in the initial conditions. But once these short range interactions are attenuated we expect the two systems to behave in a more or less similar fashion. Hence, an approach which uses only short range interactions might not be able to correctly identify these clips.

To further test the effectiveness of our method, we introduced some "corrupted" clips (clip numbers 65–80). These are random clips which clearly cannot be modeled as dynamic textures. For instance, we used shots of people and objects taken with a very shaky camera. From Figs. 2 and 3 it is clear that our kernel is able to pick up such random clips as novel. This is because our kernel compares the similarity between each frame during each time step. Hence if two clips are very dissimilar our kernel can immediately flag this as novel.

### *8.4. Clustering Short Video Clips*

We also apply our kernels to the task of clustering short video clips. Our aim here is to show that our kernels are able to capture semantic information contained in scene dynamics which are difficult to model using traditional measures of similarity.

In our experiment we randomly sample 480 short clips (120 frames each) from the movie Kill Bill Vol. 1, and model each clip as the output of a linear ARMA model. As before, we use the procedure outlined in Doretto et al. (2003) for estimating the model parameters $A$, $C$, $Q$, and $R$. The trace kernel described in Section 4 was applied to the estimated models, and the metric defined by the kernel was used to compute the $k$-nearest neighbors of a clip. Locally Linear Embedding (LLE) (Roweis and Saul, 2000) was used to cluster, and embed the clips in two dimensions using the $k$-nearest neighbor information obtained above. The two dimensional embedding obtained by LLE is depicted in Fig. 4. Observe the linear cluster (with a projecting arm) in Fig. 4. This corresponds to clips which are temporally close to each other and hence have similar dynamics. For instance, clips in the far right depict a person rolling in the snow while those in the far left corner depict a sword fight while clips in the center involve conversations between two characters. To visualize this, we randomly select a few data points from

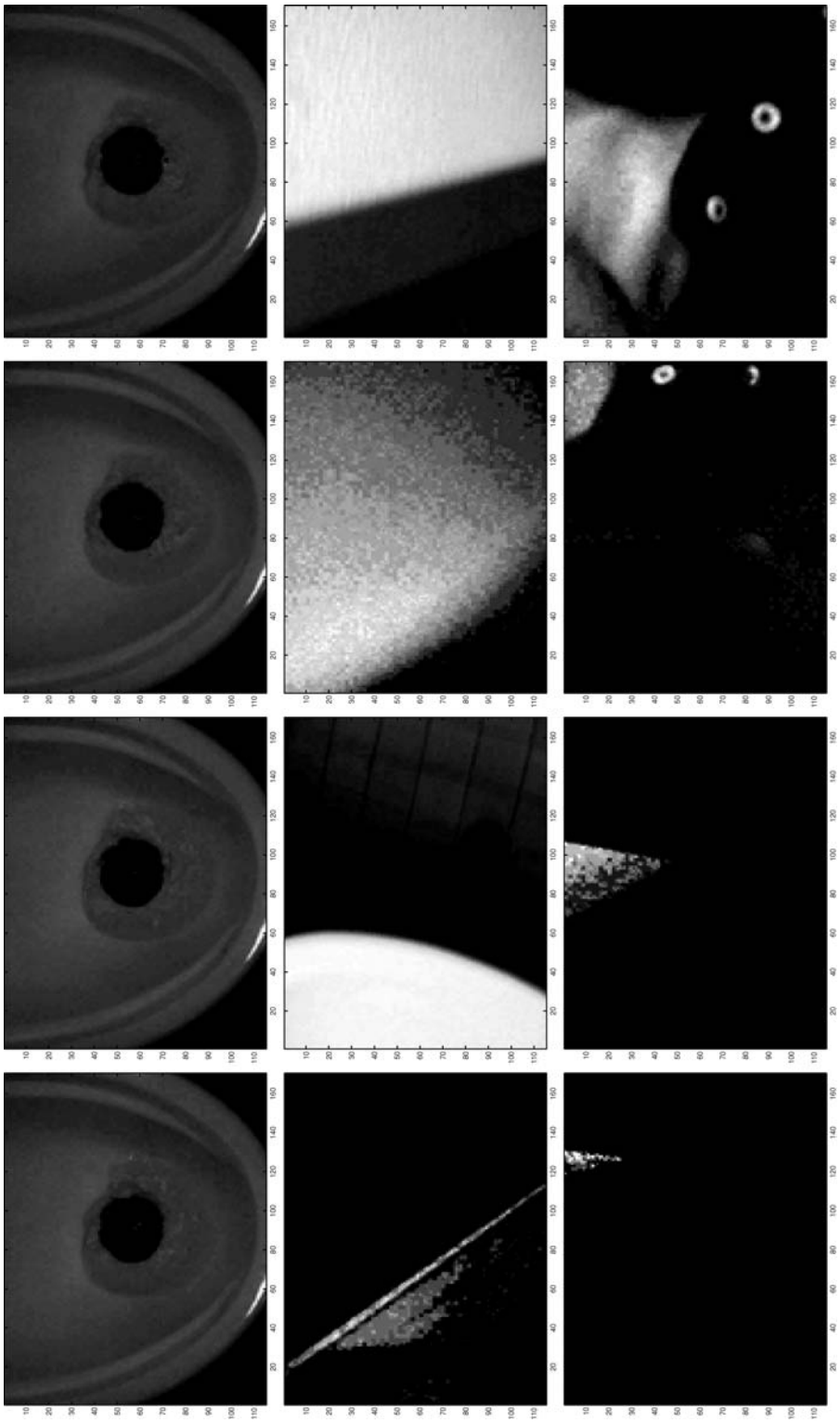

*Figure 3.* Typical frames from a few samples are shown. The first column shows four frames of a flushing toilet, while the last two columns show corrupted sequences that do not correspond to a dynamic texture. The distance matrix shows that our kernel is able to pick out this anomaly.

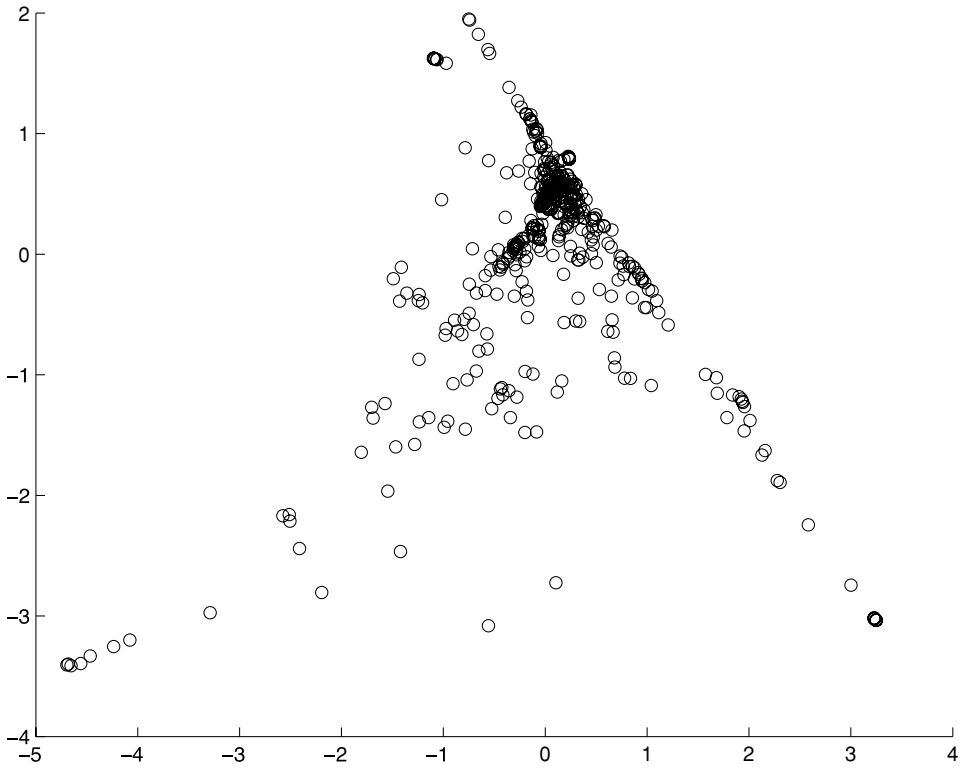


*Figure 4.* LLE Embeddings of random clips from Kill Bill.

Fig. 4 and depict the first frame of the corresponding clip in Fig. 5.

A naive comparison of the intensity values or a dot product of the actual clips would not be able to extract such semantic information. Even though the camera angle varies with time our kernel is able to successfully pick out the underlying dynamics of the scene. These experiments are encouraging and future work will concentrate on applying this to video sequence querying.

## 9. Summary and Outlook

The current paper sets the stage for kernels on dynamical systems as they occur frequently in linear and affine systems. By using correlations between trajectories we were able to compare various systems on a behavioral level rather than a mere functional description. This allowed us to define similarity measures in a natural way.

In particular, for a linear first-order ARMA process we showed that our kernels can be computed in closed form. Furthermore, we showed that the Martin distance used for dynamic texture recognition is a kernel. The main drawback of the Martin kernel is that it does not take into account the initial conditions and might be very expensive to compute. Our proposed kernels overcome these drawbacks and are simple to compute. Another advantage of our kernels is that by appropriately choosing downweighting parameters we can concentrate on either short term or long term interactions.

While the larger domain of kernels on dynamical systems is still untested, special instances of the theory have proven to be useful in areas as varied as classification with categorical data (Kondor and Lafferty, 2002; Gärtner et al., 2003) and speech processing (Cortes et al., 2002). This gives reason to believe that further useful applications will be found shortly. For instance, we could use kernels in combination with novelty detection to determine unusual initial conditions, or likewise, to find unusual dynamics. In addition, we can use the kernels to find a metric between objects such as Hidden Markov Models (HMMs), e.g. to compare various estimation methods. Preliminary work in this direction can be found in Vishwanathan (2002).

*Figure 5.* LLE embeddings of a subset of our dataset.

Future work will focus on applications of our kernels to system identification, and computation of closed form solutions for higher order ARMA processes.

## Appendix A. Proof of Theorem 14

We need the following technical lemma in order to prove Theorem 14.

**Lemma 18.** *Let $S, T \in \mathbb{R}^{n\times n}$. Then, for all $\lambda$ such that $e^{-\lambda}\|S\|\|T\| < 1$ and for all $\bar{\mathrm{W}} \in \mathbb{R}^{n\times n}$ the series*

$$M := \sum_{t=0}^{\infty} e^{-\lambda t} S^t \,\bar{\mathrm{W}}\, T^t$$

*converges, and $M$ can be computed by solving the Sylvester equation $e^{-\lambda} SMT + \bar{\mathrm{W}} = M$.*

**Proof:** To show that $M$ is well defined we use the triangle inequality, which leads to

$$\begin{aligned}\|M\| &= \left\| \sum_{t=0}^{\infty} e^{-\lambda t} S^t \,\bar{\mathrm{W}}\, T^t \right\| \leq \sum_{t=0}^{\infty} \left\| e^{-\lambda t} S^t \,\bar{\mathrm{W}}\, T^t \right\| \\ &\leq \|\,\bar{\mathrm{W}}\,\| \sum_{t=0}^{\infty} \left(e^{-\lambda}\|S\|\|T\|\right)^t \\ &= \frac{\|\,\bar{\mathrm{W}}\,\|}{1 - e^{-\lambda}\|S\|\|T\|}.\end{aligned}$$

The last equality follows because $e^{-\lambda}\|S\|\|T\| < 1$. Next, we decompose the sum in $M$ and write

$$\begin{aligned}M &= \sum_{t=0}^{\infty} e^{-\lambda t} S^t \,\bar{\mathrm{W}}\, T^t = \sum_{t=1}^{\infty} e^{-\lambda t} S^t \,\bar{\mathrm{W}}\, T^t + \bar{\mathrm{W}} \\ &= e^{-\lambda} S \left[ \sum_{t=0}^{\infty} e^{-\lambda t} S^t \,\bar{\mathrm{W}}\, T^t \right] T + \bar{\mathrm{W}} \\ &= e^{-\lambda} SMT + \bar{\mathrm{W}}.\end{aligned}$$

□

We are now ready to prove the main theorem.

**Proof:** By repeated substitution of (9) we see that

$$y_t = C\left[A^t x_0 + \sum_{i=0}^{t-1} A^i v_{t-1-i}\right] + w_t.$$

Hence, in order to compute (10) we need to take expectations and sums over 9 different terms for every $y_t^\top W y_t'$. Fortunately, terms involving $v_i$ alone, $w_i$ alone, and the mixed terms involving $v_i$, $w_j$ for any $i$, $j$, and the mixed terms involving $v_i$, $v_j$ for $i \neq j$ vanish since we assumed that all the random variables are zero mean and independent. Next note that

$$\mathbb{E}_{w_t}\left[w_t^\top W w_t\right] = \text{tr } WR,$$

where $R$ is the covariance matrix of $w_t$, as specified in (9). Taking sums over $t$ yields

$$\sum_{t=0}^{\infty} e^{-\lambda t} \text{ tr } WR = \frac{1}{1-e^{-\lambda}} \text{ tr } WR. \tag{41}$$

Next, in order to address the terms depending only on $x_0, x_0'$ we define

$$\bar{\mathrm{W}} := C^\top W C',$$

and write

$$\begin{aligned}\sum_{t=0}^{\infty} e^{-\lambda t}(CA^t x_0)^\top W (C' A'^t x_0') &= x_0^\top \left[\sum_{t=0}^{\infty} e^{-\lambda t}(A^t)^\top \bar{\mathrm{W}} A'^t\right] x_0' \\ &= x_0^\top M x_0', \end{aligned} \tag{42}$$

where, by Lemma 18, $M$ is the solution of

$$M = e^{-\lambda} A^\top M A' + C^\top W C'.$$

The last terms to be considered are those depending on $v_i$. Recall two facts: First, for $i \neq j$ the random variables $v_i$, $v_j$ are independent. Second, for all $i$ the random variable $v_i$ is normally distributed with covariance $Q$. Using these facts we have

$$\begin{aligned}&\mathbb{E}_{v_t}\left[\sum_{t=0}^{\infty}\sum_{j=0}^{t-1} e^{-\lambda t}(CA^j v_{t-1-j})^\top W (C' A'^j v_{t-1-j})\right] \\ &= \text{tr } Q\left[\sum_{t=0}^{\infty} e^{-\lambda t}\sum_{j=0}^{t-1}(A^j)^\top \bar{\mathrm{W}} A'^j\right] \\ &= \text{tr } Q\left[\sum_{j=0}^{\infty}\frac{e^{-\lambda j}}{1-e^{-\lambda}}(A^j)^\top \bar{\mathrm{W}} A'^j\right] \qquad (43) \\ &= \frac{1}{1-e^{-\lambda}} \text{tr } QM. \qquad (44)\end{aligned}$$

Here (43) follows from rearranging the sums, which is permissible because the series is absolutely convergent as $e^{-\lambda}\|A\|\|A'\| < 1$. Combining (41), (42), and (44) yields the result. □

## Appendix B. Proof of Theorem 17

**Proof:** We begin by setting

$$M = \int_0^\infty e^{-\lambda t} \exp(At)^\top W \exp(A't)\, dt.$$

To show that $M$ is well defined we use the triangle inequality, the fact that $\lambda > 2\Lambda$, and $||W|| < \infty$ to write

$$\begin{aligned}\|M\| &\leq \int_0^\infty e^{-\lambda t}\|\exp(At)^\top W \exp(A't)\|\, dt \\ &\leq \int_0^\infty \exp((-\lambda + 2\Lambda)t)\|W\|\, dt < \infty.\end{aligned}$$

Using integration by parts we can write

$$\begin{aligned}M =\ & (A^\top)^{-1} e^{-\lambda t}(\exp(At))^\top W \exp(A't)\Big|_0^\infty \\ & - \int_0^\infty (A^\top)^{-1} e^{-\lambda t}(\exp(At))^\top W \\ & \times \exp(A't)(A' - \mathbf{I}\lambda)\, dt \\ =\ & -(A^\top)^{-1} W - (A^\top)^{-1} M(A' - \lambda\, \mathbf{I}).\end{aligned}$$

Here we obtained the last line by realizing that the integrand vanishes for $t \to \infty$ (for suitable $\lambda$) in order to make the integral convergent. Multiplication by $A^\top$ shows that $M$ satisfies

$$A^\top M + MA' - \lambda M = -W.$$

Rearranging terms, and the fact that multiples of the identity matrix commute with $A, A'$ proves the first part of the theorem.

To obtain (18) we simply plug in Dirac's $\delta$-distribution into (17), and observe that all terms for $t \neq \tau$ vanish. □

## Acknowledgments

We thank Karsten Borgwardt, Stéphane Canu, Laurent El Ghaoui, Patrick Haffner, Daniela Pucci de Farias, Frederik Schaffalitzky, Nic Schraudolph, and Bob Williamson for useful discussions. We also thank Gianfranco Doretto for providing code for the estimation of linear models, and the anonymous referees for providing constructive feedback. National ICT Australia is supported by the Australian Government's Program Backing Australia's Ability. SVNV and AJS were partly supported by the IST Programme of the European Community, under the Pascal Network of Excellence, IST-2002-506778. RV was supported by a by Hopkins WSE startup funds, and by grants NSF-CAREER ISS-0447739 and ONR N000140510836.

## Notes

1. For instance, the model parameters of a linear dynamical model are determined only up to a change of basis, hence the space of models has the structure of a Stiefel manifold.
2. If further generality is required $\mathbf{A}(\cdot, t)$ can be assumed to be non-linear, albeit at the cost of significant technical difficulties.
3. Note that we made no specific requirements on the parameterization of $\mathbf{A}$, $\mathbf{A}'$. For instance, for certain ARMA models the space of parameters has the structure of a manifold. The joint probability distribution, by its definition, has to take such facts into account. Often the averaging simply takes additive noise of the dynamical system into account.
4. As shown in De Cock and De Moor (2002), the squared cosines of the principal angles between the column spaces of $\mathcal{O}$ and $\mathcal{O}'$ can be computed as the $n$ eigenvalues of $(\mathcal{O}^\top\mathcal{O})^{-1}\mathcal{O}^\top\mathcal{O}'(\mathcal{O}'^\top\mathcal{O}')^{-1}\mathcal{O}'^\top\mathcal{O}$. From the QR decomposition $Q^\top Q = Q'^\top Q' = \mathcal{I}$, hence $\prod_{i=1}^n \cos^2(\theta_i) = \det(\mathcal{O}^\top\mathcal{O}')^2/(\det(\mathcal{O}^\top\mathcal{O})\det(\mathcal{O}^\top\mathcal{O}))$.
5. Recall that the square of a kernel is a kernel thanks to the product property.
6. For ease of exposition we will ignore labels on edges. If desired, they can be incorporated into our discussion in a straightforward manner.
7. For example, consider sequence with a tree in the foreground with lawn in the background versus a sequence with a tree in the foreground and a building in the background.

## References

Aggarwal, G., Roy-Chowdhury,A., and Chellappa, R. 2004. A system identification approach for video-based face recognition. In *Proc. Intl. Conf. Pattern Recognition*, Cambridge, UK.

Aitken, A.C. 1946. *Determinants and Matrices*, 4th edition. Interscience Publishers.

Bach, F.R. and Jordan, M.I. 2002. Kernel independent component analysis. *Journal of Machine Learning Research*, 3:1–48.

Bakir, G., Weston, J., and Schölkopf, B. 2003. Learning to find pre-images. In *Advances in Neural Information Processing Systems 16*. MIT Press.

Barla, A., Odone, F., and Verri, A. 2002. Hausdorff kernel for 3D object acquisition and detection. In *European Conference on Computer Vision '02*, number 2353 in LNCS, pp. 20. Springer.

Baxter, J. and Bartlett, P.L. 1999. Direct gradient-based reinforcement learning: Gradient estimation algorithms. Technical report, Research School of Information, ANU Canberra.

Bernstein D.S. 2005. *Matrix Mathematics*. Princeton University Press.

Blanz, V., Schölkopf, B., Bülthoff H., Burges, C., Vapnik, V., and Vetter, T. 1996. Comparison of view-based object recognition algorithms using realistic 3D models. In C. von der Malsburg, W. von Seelen, J.C. Vorbrüggen, and B. Sendhoff (eds.), *Artificial Neural Networks ICANN'96*, vol. 1112 of *Lecture Notes in Computer Science*, pp. 251–256. Berlin: Springer-Verlag.

Burges, C.J.C. and Vapnik, V. 1995. A new method for constructing artificial neural networks. Interim technical report, ONR contract N00014-94-c-0186, AT&T Bell Laboratories.

Burkhardt, H. 2004. Invariants on skeletons and polyhedrals. Technical report, Universität Freiburg, in preparation.

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

Chapelle, O., Haffner, P., and Vapnik, V. 1999. SVMs for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5).

De Cock, K. and De Moor, B. 2002. Subspace angles between ARMA models. *Systems and Control Letter*, 46:265–270.

Cortes, C., Haffner, P., and Mohri, M. 2002. Rational kernels. In *Advances in Neural Information Processing Systems 15*, vols. 14. Cambridge, MA: MIT Press.

Cristianini, N. and Shawe-Taylor, J. 2000. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK.

de Finetti, B. 1990. *Theory of probability*, vol. 1–2. John Wiley and Sons, 1990. reprint of the 1975 translation.

Doretto, G., Chiuso, A., Wu, Y.N., and Soatto, S. 2003. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109.

Gardiner, J.D., Laub, A.L., Amato, J.J., and Moler, C.B. 1992. Solution of the Sylvester matrix equation $AXB^\top + CXD^\top = E$. *ACM Transactions on Mathematical Software*, 18(2):223–231.

Gärtner, T., Flach, P.A., and Wrobel, S. 2003. On graph kernels: Hardness results and efficient alternatives. In B. Schölkopf and M.K. Warmuth, (eds.) *Sixteenth Annual Conference on Computational Learning Theory and Seventh Kernel Workshop, COLT*. Springer.

Golub, G.H. and Van Loan, C.F. 1996. *Matrix Computations*, 3rd edition. Baltimore, MD: John Hopkins University Press.

Gretton, A., Herbrich, R., and Smola, A.J. 2003. The kernel mutual information. In *Proceedings of ICASSP*.

Gröbner, W. 1965. *Matrizenrechnung*. BI Hochschultaschenbücher.

Heisele, B., Ho, P., and Poggio, T. 2001. Face recognition with support vector machines: Global versus component-based approach. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pp. 688–694. Los Alamitos, CA: IEEE Computer Society.

Herbrich, R. 2002. *Learning Kernel Classifiers: Theory and Algorithms*. MIT Press.

Isidori, A. 1989. *Nonlinear Control Systems*. 2nd edition. Springer.

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pp. 137–142. Berlin: Springer.

Kashima, H., Tsuda, K., and Inokuchi, A. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, Washington, DC, United States.

Kashima, H., Tsuda, K., and Inokuchi, A. 2004. Kernels on graphs. In K. Tsuda, B. Schölkopf, and J.P. Vert (Eds.), *Kernels and Bioinformatics*, Cambridge, MA: MIT Press.

Kondor, I.R. and Lafferty, J.D. 2002. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the ICML*.

Luenberger, D.G. 1979. *Introduction to Dynamic Systems: Theory, Models, and Applications*. New York, USA:John Wiley and Sons, Inc., ISBN 0–471 - 02594 - 1.

Martin, R.J. 2000. A metric for ARMA processes. *IEEE Transactions on Signal Processing*, 48(4):1164–1170.

Nocedal, J. and Wright, S.J. 1999. *Numerical Optimization*. Springer Series in Operations Research. Springer.

Pinkus, A. 1996. Spectral properties of totally positive kernels and matrices. In M. Gasca and C.A. Micchelli (eds.), *Total Positivity and its Applications*, vol. 359 of *Mathematics and its Applications*, pp. 1–35. Kluwer.

Platt, J. 1999. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C.J.C. Burges, and A.J. Smola (eds.), *Advances in Kernel Methods—Support Vector Learning*, pp. 185–208, Cambridge, MA: MIT Press.

Ralaivola, L. and d'Alché Buc, F. 2003. Dynamical modeling with kernels for nonlinear time series prediction. In Se. Thrun, La. Saul, and Be. Schölkopf, (eds.) *Advances in Neural Information Processing Systems 16*. MIT Press.

Roweis, S. and Saul, L.K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326.

Saisan, P., Doretto, G., Wu, Y.N., and Soatto, S. 2001. Dynamic texture recognition. In *Proceedings of CVPR*, vol. 2, pp. 58–63.

Schölkopf, B. 1997. *Support Vector Learning*. R. Oldenbourg Verlag, Munich, Download: http://www.kernel-machines.org.

Schölkopf, B. and Smola, A. 2002. *Learning with Kernels*. Cambridge, MA: MIT Press.

Shashua, A., and Hazan, T. 2005. Algebraic set kernels with applications to inference over local image representations. In L.K. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 17*, pp. 1257–1264, Cambridge, MA: MIT Press.

Shi, J. and Malik, J. 1997. Normalized cuts and image segmentation. *IEEE Conf. Computer Vision and Pattern Recognition*.

Smola, A.J. and Kondor, I.R. 2003. Kernels and regularization on graphs. In B. Schölkopf and M.K. Warmuth (eds.), *Proceedings of the Annual Conference on Computational Learning Theory*, Lecture Notes in Computer Science, pp. 144–158. Springer.

Smola, A.J. and Vishwanathan, S.V.N. 2003. Hilbert space embeddings in dynamical systems. In *Proceedings of the 13th IFAC symposium on system identification*. Rotterdam, Netherlands.

Soatto, S. Doretto, G., and Wu: Y.N. 2001. Dynamic textures. In *Proceedings of the Eighth International Conference On Computer Vision (ICCV-01)*, pp. 439–446. Los Alamitos, CA: IEEE Computer Society.

Sutton, R.S. and Barto, A.G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York: Springer .

Vapnik, V. 1998. *Statistical Learning Theory*. New York: John Wiley and Sons.

Vidal, R., Ma, Y., and Sastry, S. 2005. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, In press.

Vishwanathan, S.V.N. 2002. *Kernel Methods: Fast Algorithms and Real Life Applications*. PhD thesis, Indian Institute of Science, Bangalore, India.

Vishwanathan, S.V.N., Borgwardt, K., and Schraudolph, Nicol N. 2006. Faster graph kernels. In *International Conference on Machine Learning*, submitted.

Vishwanathan, S.V.N., Smola, A.J., and Murty, M.N. 2003. SimpleSVM. In T. Fawcett and N. Mishra (eds.), *Proceedings of the 20th International Conference on Machine Learning (ICML)*, Washington DC, AAAI press.

Willems, J.C. 1986b. From time series to linear system. I. Finite-dimensional linear time invariant systems. *Automatica J. IFAC*, 22(5):561–580.

Willems, J.C. 1986a. From time series to linear system. II. Exact modelling. *Automatica J. IFAC*, 22(6):675–694.

Willems, J.C. 1987. From time series to linear system. III. Approximate modelling. *Automatica J. IFAC*, 23(1):87– 115.

Wolf, L. and Shashua, A. 2003. Learning over sets using kernel principal angles. *Jounal of Machine Learning Research*, 4:913–931.