Applications of Hybrid System Identification in Computer Vision

René Vidal
Center for Imaging Science
Johns Hopkins University
rvidal@cis.jhu.edu

Stefano Soatto
Computer Science Department
University of California, Los Angeles
soatto@cs.ucla.edu

Alessandro Chiuso
Dipartimento di Tecnica e Gestione dei
Sistemi Industriali, Universitá di Padova
chiuso@dei.unipd.it

***Abstract*— In this expository paper we illustrate the use of filtering and identification-theoretic techniques in a number of problems in computer vision. We first demonstrate how linear system identification techniques combined with distances for linear systems can be used for modeling, synthesis, classification and recognition of dynamic textures and human gaits. We then show how hybrid system identification techniques can be used for segmentation of dynamic textures. We also highlight some open problems in system identification that are motivated by extensions of the research described in this paper.**

I. Introduction

Recently, several system-theoretic techniques have been used for modeling dynamic visual processes. For instance, [10], [36] model the appearance of *dynamic textures*, such as videos of water, smoke, fire, etc., as the output of an Auto Regressive Moving Average (ARMA) model; [4] uses ARMA models to represent human gaits, such as walking, running, jumping, etc.; and [1], [26] use ARMA models to describe the appearance of moving faces. Given a video sequence, one can use standard system identification techniques, e.g., subspace identification [25], to learn the parameters of the ARMA model. Given a model, one can use them to generate novel synthetic sequences [10], [36], [28], [29], manipulate real ones [12], and recognize one from another [27], [10].

However, most of these methods assume that the scene contains a single dynamic texture, human gait, or moving face, so that the video can be modeled with a single linear dynamical model. This limitation has motivated recent work on modeling scenes consisting of multiple temporal events, e.g., a video sequence consisting of multiple shots. [17] models such videos as the output of a linear hybrid system, where each discrete state corresponds to a different event in the video sequence. The parameters of each linear model and the sequence of discrete events can be identified from the video sequence using existing linear hybrid system identification techniques for switched ARX systems [34], [21], [31] or piece-wise ARX systems [2], [14], [18].

More recent works [6], [8], [11], [33] consider scenes where different regions in the image have different dynamics, e.g., a video sequence of fire in the foreground and water in the background. Such sequences can be modeled as the output of a multidimensional hybrid system where the discrete state depends on a spatial location (the pixel coordinates). Therefore, there is no switching from one dynamical system to another in time, but rather multiple dynamical system co-exist simultaneously in different spatial locations. [6] models such scenes with a mixture of ARMA models and learns the model parameters and the segmentation of the scene using Expectation Maximization (EM) [9]. As EM-like approaches are often sensitive to good initialization, [33] proposes to cluster the trajectories of the image intensities according to their corresponding observability subspaces using an algebraic method for subspace clustering called Generalized PCA (GPCA) [32]. Unfortunately, GPCA does not incorporate spatial regularization, thus the resulting contour is typically non smooth. [11] and [15] incorporate spatial regularization by modeling the boundaries between two regions in the image as the zero level set of a function ϕ. When the parameters of the dynamical systems are known, one can compute ϕ as the stationary solution of a partial differential equation. Conversely, when ϕ is known so are the boundaries, thus one can identify the parameters of each dynamical model from the data within each region. By alternating between these two steps, both the boundaries and the dynamical models can be identified.

However, in real sequences the region boundaries change as a function of time, and also the dynamics of a region may change suddenly due to depths discontinuities, occlusions, objects entering or leaving the field of view, etc. The works of [8], [30], [31] model such video sequences as the output of a spatial-temporal hybrid system consisting of multiple Auto Regressive eXogenous (ARX) models both in space and in time. [8] segments these models using GPCA. [31] proposes a recursive system identification techniques that identifies the coefficients of a polynomial encoding the parameters of all the ARX models associated with all spatial-temporal regions. The derivatives of this polynomial give an estimate of the parameters of the ARX models associated spatial location at each time. The boundaries between spatial-temporal regions are then computed once the ARX parameters are known.

In this paper we seek to infer models of dynamic visual processes for the purpose of classification, segmentation and recognition tasks. For instance, from a number of sequences of images of fire, smoke or steam, we want to identify a model that can be used to recognize, say, fire in a new sequence. Similarly, in human motion analysis, we want to infer a model of various gaits, such as walk, run, jump, limp,

We thank A. Bissacco, D. Cremers, G. Doretto, P. Favaro, A. Ghoreyshi, Y. Ma, M. Petreczky, P. Saisan, A. Smola, S. Vishwanathan and Y. Wu. R. Vidal thanks grants NSF CAREER IIS-04-47739, NSF EHS-05-09101 and ONR N00014-05-1083. S. Soatto thanks grants ONR N00014-03-1-0850:P0001 and AFOSR E-16-V91-G2. A. Chiuso thanks MIUR through project "New techniques and applications of identification and adaptive control"

so that we can for instance detect a limping person from afar. We will first address the simplest possible classes of models, working under the assumption that the underlying data exhibit some form of stationarity. Here the current body of knowledge in system identification has already played a key role in a number of applications, as we describe in Section II. Even for such simple models, however, recognition and classification tasks remain largely an open problem, which we discuss in Section III. Imposing simpler models and inferring the model parameters as well as the domain where they are satisfied within a prescribed accuracy leads to a segmentation problem, which is described in Section IV. We highlight open problems in system identification motivated by extensions to more complex models in Section V.

II. Modeling, identification and synthesis of dynamic visual processes

The general goal of computer vision is to infer properties of a "scene" from measurement of "images" obtained under varying conditions. For the purposes of this paper, a *scene* is simply a finite collection of *objects*, where each object is a volume in $\mathbb{R}^3$ bounded by closed piecewise smooth surfaces embedded in $\mathbb{R}^3$. On the other hand, an *image* is a function,

$$I : \mathbb{R}^2 \ni \boldsymbol{x} \mapsto I(\boldsymbol{x}) \in \mathbb{R}^+ \tag{1}$$

that assigns to each pixel $\boldsymbol{x}$ a positive number I that measures the intensity (irradiance) of light (electromagnetic radiation) reflected by a point in one of the surfaces in the scene. In general, the energy of the light reflected by different points in the scene varies depending on the scene *photometry* (light sources, reflectance properties of the objects), *geometry* (position of the objects relative to the light sources, position of the objects relative to the camera), and *dynamics* (temporal evolution of object surfaces and of the camera pose).

Unfortunately, one cannot recover a unique model of the photometry, geometry and dynamics of a scene from visual information alone. For instance, a video of water may be generated by an outdoor water scene with simple photometry (ambient light) and very complex geometry and dynamics (the moving surface of the water), or it may be rendered by a projector with very simple geometry (a flat screen) and very complicated photometry. Therefore, rather than trying to recover a "correct" model of the scene, we seek to recover a *statistical* model of visual data directly at the outset.

The next subsections illustrate the simplest possible model (linear systems) applied to pixel intensities, as a model for so-called "dynamic textures". This simple model is indeed useful for recognition and classification tasks, as we will see in Section III. More complex hybrid dynamical models for segmentation purposes will be discussed in Section IV.

A. Dynamic textures

Let $\{I(t) \in \mathbb{R}^{k\times l}\}_{t=1\ldots\tau}$ be a sequence of images. Suppose that at each instant of time t we can measure a noisy version of the image, $y(t) = I(t) + w(t)$ where $w(t)$ is an independent and identically distributed (IID) sequence drawn from a distribution $p_w(\cdot)$ resulting in a positive measured sequence $y(t) \in \mathbb{R}^m,\ t = 1\ldots\tau$, where $m = k \times l$. The sequence $\{I(t)\}$ is *a (linear) dynamic texture* [10] if there exists a set of n spatial filters ϕ_α, $\alpha = 1\ldots n$, and a stationary distribution $q(\cdot)$ such that, calling $z(t) \doteq \phi(I(t))$, $z(t)$ can be modeled as an ARMA process excited by the white noise $v(t)$, distributed according to $q(\cdot)$. Therefore, a dynamic texture is associated to (a second-order stationary process and, therefore) a state space model which, without loss of generality, can be written in forward innovation form

$$\begin{cases} x(t+1) = Ax(t) + Kv(t) \\ z(t) = Cx(t) + v(t) \\ y(t) = \psi(z(t)) + w(t) \end{cases} \tag{2}$$

where $x(0) = x_0$, $v(t) \overset{IID}{\sim} q(\cdot)$ unknown, $w(t) \overset{IID}{\sim} p_w(\cdot)$ given, and $I(t) = \psi(z(t))$ where $\psi(\phi(I)) = I$. One can obviously extend the definition to an arbitrary non-linear model of the form $x(t+1) = f(x(t), v(t))$, leading to the concept of a *non-linear dynamic texture*.

The definition of dynamic texture above, which was proposed in [10], entails a choice of filters $\phi_\alpha,\ \alpha = 1\ldots n$. These filters could also be inferred as part of the identification process for a given dynamic texture. There are several criteria for choosing a suitable class of filters, ranging from biological motivations to computational efficiency. In the trivial case, one can take ϕ to be the identity, and therefore look at the dynamics of individual pixels $z(t) = I(t)$ in (2). However, in texture analysis the dimension of the signal is huge (tens of thousands components) and there is a lot of redundancy. Hence, one can view the choice of filters as a dimensionality reduction step, and seek a decomposition of the image in the simple (linear) form

$$I(t) = \sum_{i=1}^{n} x_i(t)\theta_i \doteq Cx(t), \tag{3}$$

where $C = [\theta_1, \ldots, \theta_n]$ and $\{\theta_i\}$ can be an orthonormal or overcomplete basis of $\mathcal{L}^2$, e.g., a set of principal components, or a wavelet filter bank. The advantage of using principal component analysis (PCA), besides simplicity, is that it reduces complexity via a data-tailored construction of basis function. Experimental results show that 20 to 30 principal components yield synthesized textures which are practically indistinguishable from the original ones. Standard approaches based on filter banks require more coefficients to obtain comparable results; for instance, dynamic textures based on Fourier and Gabor filters were presented in [37].

B. Inference

The problem of going from data to models is the usual system identification problem. Several approaches have been proposed in the literature ranging from standard prediction error methods [20], to iterative solutions based on EM [9], to the more recent subspace methods [25].

For the case of dynamic textures, due to the dimension of the signal (76,800 for video at half-resolution), one cannot apply standard identification algorithms. A dimensionality reduction step is a must as we have discussed at the end of

section II-A. Even after this reduction step, the dimension of $x(t)$ is still high (20 to 50 in typical examples presented in [10]). If we restrict ourselves to first-order AR processes for modeling the evolution of $x(t)$, the following algorithm, proposed in [10], yields a simple and yet effective solution: Let $Y_1^\tau \doteq [y(1), \ldots, y(\tau)] \in \mathbb{R}^{m\times\tau}$ with $\tau > n$, and similarly for X_1^τ and W_1^τ, and notice that

$$Y_1^\tau = CX_1^\tau + W_1^\tau; \quad C \in \mathbb{R}^{m\times n}; \quad C^TC = I \tag{4}$$

by our assumptions. Now let $Y_1^\tau = U\Sigma V^T$; $U \in \mathbb{R}^{m\times n}$, $U^TU = I$, $V \in \mathbb{R}^{\tau\times n}$, $V^TV = I$, be the singular value decomposition (SVD) with $\Sigma = \text{diag}\{\sigma_1, \ldots, \sigma_n\}$, and consider the problem of finding the best estimate of C in the sense of Frobenius: $\hat{C}(\tau), \hat{X}(\tau) = \arg\min_{C,X_1^\tau} \|W_1^\tau\|_F^2$ subject to (4). It follows immediately from the fixed-rank approximation property of the SVD that the unique solution (modulo a change of sign) is given by

$$\hat{C}(\tau) = U \quad \hat{X}(\tau) = \Sigma V^T. \tag{5}$$

Given $\hat{X}(\tau)$, $\hat{A}$ can be determined uniquely, again in the sense of Frobenius, by solving the following linear problem: $\hat{A}(\tau) = \arg\min_A \|X_1^\tau - AX_0^{\tau-1}\|_F^2$ which is trivially done in closed form using an estimate of X from (5):

$$\hat{A}(\tau) = \Sigma V^T D_1 V (V^T D_2 V)^{-1}\Sigma^{-1} \tag{6}$$

where $D_1 = \begin{bmatrix} 0 & 0 \\ I_{\tau-1} & 0 \end{bmatrix}$ and $D_2 = \begin{bmatrix} I_{\tau-1} & 0 \\ 0 & 0 \end{bmatrix}$. Notice that $\hat{C}(\tau)$ is uniquely determined up to a change of sign of the components of C and x. Also note that

$$E[\hat{x}(t)\hat{x}^T(t)] \equiv \lim_{\tau\to\infty} \frac{1}{\tau}\sum_{k=1}^{\tau} \hat{x}(t+k)\hat{x}^T(t+k) \simeq \Sigma^2 \tag{7}$$

which is diagonal. Finally, the sample input noise covariance Q can be estimated from

$$\hat{Q}(\tau) = \frac{1}{\tau}\sum_{i=1}^{\tau} \hat{v}(i)\hat{v}^T(i), \tag{8}$$

where $\hat{v}(t) \doteq \hat{x}(t+1) - \hat{A}(\tau)\hat{x}(t)$. This algorithm requires the number of principal components n to be given. In practice, n needs to be inferred from data. This can be done from the singular values $\{\sigma_i\}$, by choosing n such that $\sigma_{n+1}/\sigma_1 < \epsilon$, or $\sigma_{n+1}/\sigma_n < \epsilon$, where $\epsilon > 0$ is a user defined threshold.

C. Synthesis

As reported in [10], identifying a model from a sequence of 100 frames takes about 5 minutes in MATLAB on a 1GHz Pentium III personal computer. Synthesis can be performed in real time. The implementation in [10] used τ between 50 and 150 and n between 20 and 50. Figures 1-2 show the behavior of the algorithm on a representative set of experiments. Figure 1 shows the overall compression error as a function of the dimension of the state space (top row) as well as the prediction error as a function of the length of the learning set (bottom row). The prediction error is computed by using the first τ images to identify a model, then using the model to predict the image at $\tau + 1$, finally comparing the result with the actual image measured at $\tau + 1$. The plot shows the error between the predicted and measured images at $\tau+1$ as a function of τ. Figure 2 shows results on synthesis of novel sequences. For each sequence, the first row shows a few images from the original dataset, and the second row shows a few extrapolated samples.

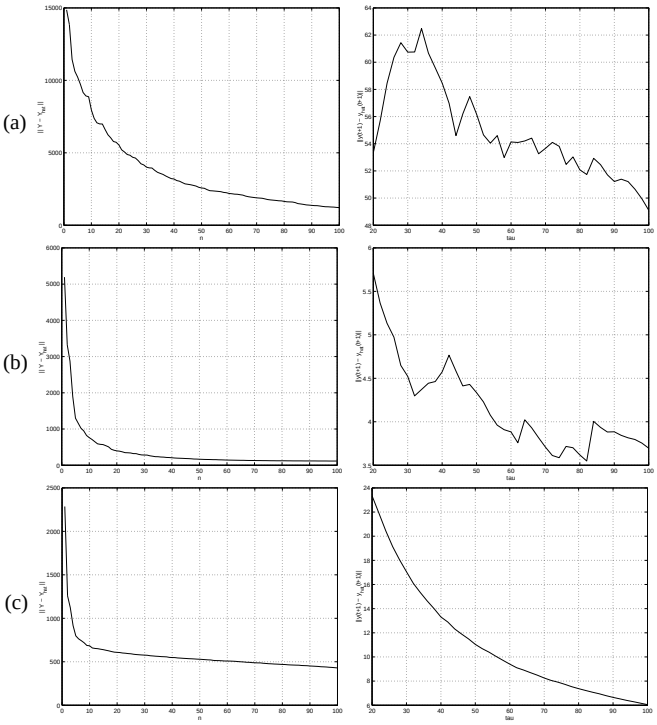


Fig. 1. Compression error as a function of the dimension of the state space n (left column), and extrapolation error as a function of the length of the training set τ (right column). Row (a) *river* sequence, row (b) *smoke* sequence, column (c) *toilet* sequence. (Figure courtesy of [10])

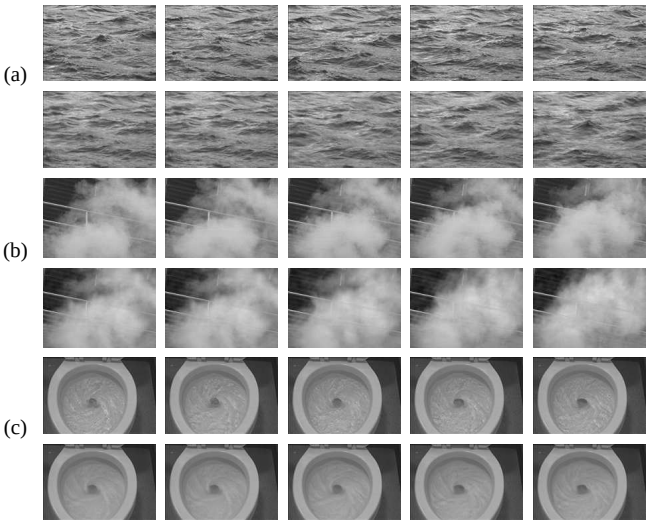


Fig. 2. Synthesis of novel dynamic textures from previously learned ARMA models: (a) *river* sequence, (b) *smoke* sequence, (c) *toilet* sequence. For each of them the top row are samples of the original sequence, the bottom row shows samples of the extrapolated sequence. All the data are available on-line at *http://www.vision.ucla.edu/projects/dynamic-textures.html* (Figure courtesy of [10])

III. Classification and recognition of dynamic visual processes

One approach to classification and recognition is to look directly at the data sequences. These can be regarded as samples from some distribution, thus classical concepts of discrepancy measures such as Kullback-Leibler divergence [19], or Bhattacharyya-Hellinger metric [3], could be employed. The space of probability distributions, suitably restricted, can be given the structure of a differentiable manifold endowed with a Riemannian structure. Thus, one can define distances and probability distributions, which are the basis of standard techniques such as Bayes classification or likelihood ratios. This, however, has several drawbacks. First, one has to take into account that different datasets may have different length. In fact, the probability distribution of an n-vector lives on a different space than that of an m-vector if $m \neq n$. Therefore, it seems more appropriate to look at invariants of the process itself. If we restrict to second-order stochastic processes, these are the spectrum, the covariance function or a spectral factor or, in other words, an ARMA model.

ARMA models, learned from data as described in the previous section, do not live on a linear space. The space of linear systems can be given a structure of a differentiable manifold (see [16]) and is certainly not flat. Recall that in order for the models to be stable and invertible, the matrices A and $A - KC$ are constrained to be stable. Furthermore, all the triples (TAT^{-1}, TK, CT^{-1}) are equivalent. While a simple probabilistic approach to classification based on likelihood ratios with respect to a simple probability distribution on Stiefel manifolds has been proposed in [27], in general it is not straightforward to introduce manageable probability distribution on the space of dynamical models.

In the next subsections, we restrict ourselves to describing an approach to classification and recognition based on distances defined on the space of models, which could be used for nearest-neighbor classification or k-means clustering [13].

A. Distances between dynamical models

We first review the distance for scalar ARMA models introduced by Martin [22] and elaborated upon by [7]. We then review the Binet-Cauchy kernels introduced by [35].

The cepstrum of an ARMA model with transfer function $H(z)$ is defined as $\log(H(z)H^*(1/z)) = \sum_{k\in\mathbb{Z}} c_k z^{-k}$. In [22], Martin defines the distance between two scalar ARMA models M_1 and M_2 with transfer functions $H_1(z)$ and $H_2(z)$ as the following distance between its cepstrum coefficients

$$d_M(M_1, M_2)^2 = \sum_{k=0}^{\infty} k(c_{1k} - c_{2k})^2. \tag{9}$$

As shown in [7], the Martin distance can also be expressed in terms the (extended) observability spaces. More specifically, let $M_1 \doteq (A_1, K_1, C_1)$ and $M_2 \doteq (A_2, K_2, C_2)$ be two ARMA models and define the extended observability matrix

$$\mathcal{O}_\infty(M_i) \doteq \begin{bmatrix} C_i^T & A_i^T C_i^T & \dots & (A_i^T)^n C_i^T & \dots \end{bmatrix}^T.$$

Then, the distance between M_1 and M_2 can be expressed in terms of the subspace angles between the column spaces of $\begin{bmatrix}\mathcal{O}_\infty(M_1) & \mathcal{O}_\infty(M_2^{-1})\end{bmatrix}$ and $\begin{bmatrix}\mathcal{O}_\infty(M_2) & \mathcal{O}_\infty(M_1^{-1})\end{bmatrix}$, where $M_i^{-1} = (A_i - K_i C_i, K_i, -C_i)$. If we denote by θ_i the i^{th} canonical angle between these spaces, the distance defined by Martin can be shown to be equivalent to [7]

$$d_M(M_1, M_2)^2 = -\ln \prod_{i=1}^{2n} \cos^2 \theta_i, \tag{10}$$

where n is the model order. Although the equivalence between (9) and (10) holds for scalar ARMA processes, one can also use (10) to measure the distance between multivariable ARMA models, such as linear dynamic textures. Clearly, in the multivariate case the link with the cepstrum is lost.

In order to compute subspace angles between models, we proceed as follows

- Compute the solution $Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}$ of the Lyapunov equation

$$\mathcal{A}^T Q \mathcal{A} - Q = -\mathcal{C}^T \mathcal{C} \tag{11}$$

where

$$\mathcal{A} \doteq \begin{pmatrix} A_1 & 0 & 0 & 0 \\ 0 & A_2 - K_2 C_2 & 0 & 0 \\ 0 & 0 & A_2 & 0 \\ 0 & 0 & 0 & A_1 - K_1 C_1 \end{pmatrix} \tag{12}$$

$$\mathcal{C} \doteq \begin{pmatrix} C_1 & -C_2 & C_2 & -C_1 \end{pmatrix}. \tag{13}$$

- Compute the $2n$ largest eigenvalues $\{\lambda_i = \cos\theta_i\}_{i=1}^{2n}$ of

$$\begin{pmatrix} 0 & Q_{11}^{-1} Q_{12} \\ Q_{22}^{-1} Q_{21} & 0 \end{pmatrix}.$$

An alternative approach to computing distances between dynamical models is based on the so-called Binet-Cauchy kernels [35]. These kernels are obtained by computing the trace of a compound matrix of order q built from the output trajectories. More specifically, let $Y_i = [y_i(0), y_i(1), \cdots]$. The Binet-Cauchy kernel of order q is defined as

$$k_q(M_1, M_2) = \text{trace}\, \mathcal{C}_q(Y_1 Y_2^T), \tag{14}$$

where $\mathcal{C}_q(X)$ is a matrix whose entries are given by the determinants of all the minors of X of size q. As shown in [35], when $q = 1$, one can compute this kernel explicitly in terms of the ARMA models $M_i = (x_i(0), A_i, K_i, C_i)$ as

$$k_1(M_1, M_2) = x_1^T(0) Q x_2(0), \tag{15}$$

where Q is the solution to the Sylverster's equation

$$A_1^T Q A_2 - Q = -C_1^T C_2. \tag{16}$$

From the Binet-Cauchy kernel, one may define a distance as $d_1(M_1, M_2)^2 = k_1(M_1, M_1) - 2k_1(M_1, M_2) + k_1(M_2, M_2)$. Notice that, unlike the Martin distance, the distance d_1 depends explicitly on the initial conditions. While this might be undesirable in some applications, it might be quite useful in other cases. For instance, not every initial configuration of the joints of the human body is appropriate for modeling human gaits. Thus incorporating the initial condition when comparing dynamical systems may be useful for recognition.

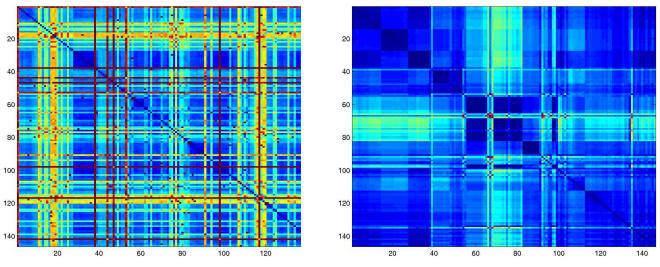
Figure 3 shows an experiment taken from [35] in which the pairwise distances d_M and d_1 are computed on a database of 144 video clips of sequences of dynamic textures of natural and artificial objects, such as trees, water bodies, water flow in kitchen sinks, etc. In general, clips that are close to each other on an axis in the figure are closely related (they are either from similar natural phenomenon or are extracted from the same master clip). However, clips 65–80 are random clips of people and objects taken by a shaky camera, which cannot be modeled as dynamic textures. Notice from the plots that the Binet-Cauchy distance is small (darker colors) for similar clips, and large (lighter colors) for random clips. The Martin distance does not seem to capture these differences very well.

B. Nearest neighbor classification

Suppose we are given a (training) set of dynamical systems $\{M_i\}$ drawn independently from an unknown probability distribution. Suppose also we are given a label $\lambda_i \in \{1, \dots, c\}$ denoting the membership of M_i to one out of c classes. Given a new system M, we want to predict its label λ. As we already have a distance in the space of dynamical models, we can use any distance-based classification technique. One of the simplest methods is nearest neighbor classification [13], where λ is chosen by taking a vote among the k nearest neighbors of M. That is, λ_m is selected if the majority of the k nearest neighbors of M have label λ_m.

In [4], nearest neighbor classification together with the Martin distance were used for recognizing different types of human gaits. The training set consists of several sequences of humans walking, running and going up and down a staircase, as shown in Figure 4. For each gait, 10 sequences of different subjects were acquired with varying viewing position and distance from the camera to the subject. From each sequence, a reduced kinematical model consisting of the projection of 4 joints onto the image plane (shoulder, elbow, hip and knee) was extracted. For each sequence, the vector consisting of the location of all these joints was consider as the output $y(t)$ of an ARMA model. Each model was identified using the implementation of the N4SID algorithm in the Matlab System Identification Toolbox. Note that for human gaits the dimension of $y(t)$ is much smaller than the number of pixels, thus one does not need to resort to the sub-optimal method described in Section II-B for identifying dynamic textures.

Figure 5 shows the pairwise distance between each model in the dataset. Notice that these three gaits are quite similar to each other (as opposed, say, to dancing or jumping), and yet similar gaits result in smaller distances, with a few outliers. A few sample sequences from each class were also chosen as test sequences for classification purposes. For each sequence in the test set, a model was estimated by first pre-processing the sequence (after manual initialization) using the ideas described in [5] to extract joint coordinates. Sample frames from different test sequences are shown in the first and third columns of Figure 6, while the first two corresponding nearest neighbors are shown to the right. Although this dataset is quite small, the discriminating power of the model as a representation of the dynamic sequence is visible.



(a) Martin distance (b) Binet-Cauchy distance

Fig. 3. Pairwise distances among 144 video clips of dynamic textures. Each row/column of a matrix represents a clip, and clips corresponding to similar dynamic textures are grouped in block rows/columns. Dark indicates a small distance, light a large distance. (Figure courtesy of [35])

Fig. 4. Sample frames from the dataset of the gaits: waking, running and walking a staircase. (Figures 4-6 courtesy of [4])

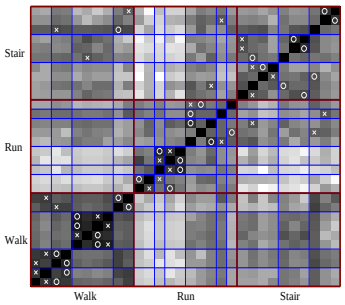


Fig. 5. Pairwise distance between video clips of human gaits. Each row/column of a matrix represents a sequence, and sequences correspondence to similar gaits are grouped in block rows/columns. Dark indicates a small distance, light a large distance. The minimum distance is of course along the diagonal, and for each row the next closest sequence is indicated by a circle, while the second nearest is indicated by a cross.

Fig. 6. *For each gait we have chosen a few sample sequences (left) and computed the distance to every other sequence in the dataset. The closest sequence is shown in the central column, while the second nearest is shown in the right column. With a few exceptions, the nearest neighbor belongs to the same gait as the test sequence. Notice that all gaits are quite similar.*

IV. Segmentation of dynamic visual processes

Consider now the problem of modeling and segmenting video sequences where different spatial-temporal regions exhibit different spatial-temporal statistics. We assume that the image at time t can be divided into $N(t)$ (unknown) regions[1] $\{\Omega_i(t) \subset \mathbb{R}^2\}_{i=1}^{N(t)}$. We model the intensity of each pixel in $\Omega_i(t)$ as the output of an ARMA model such as (2) with state $x_i(t) \in \mathbb{R}^n$ and parameters $A_i(t) \in \mathbb{R}^{n\times n}$, $K_i(t) \in \mathbb{R}^{n\times p}$, and $C_i(t) \in \mathbb{R}^{m_i(t)\times n}$, $\sum_i m_i(t) = m$. For the sake of simplicity, we assume that as time proceeds these parameters take on only finitely many values. Therefore, we can effectively model the entire video sequence as the output of $N \geq N(t)$ ARMA models $\{(A_j, K_j, C_j)\}_{j=1}^{N}$.

Given a sequence of images $\{y(t) \in \mathbb{R}^m,\ t = 1, \ldots, T\}$ with two or more distinct regions $\Omega_i(t)$ that satisfy the model (2), we want to estimate both the regions $\Omega_i(t)$, as well as the states $\{x_i(t)\}$ and parameters $A_i(t)$, $K_i(t)$ and $C_i(\boldsymbol{x}, t)$ in each region. Notice that if the boundaries of each region were known and constant in time, one could easily estimate an ARMA model of the spatial-temporal statistics within each region. Unfortunately, in general one does not know the boundaries of each region, and this is one of the goals of the inference process. On the other hand, if the state and parameters associated with each pixel were known, then one could easily determine the regions by thresholding or by other grouping or segmentation technique. Unfortunately, the model we wish to infer is not a point process, and therefore one cannot pre-compute the ARMA model at each pixel and convert the problem into a static segmentation at the outset. Therefore, we have a classic "chicken-and-egg" problem: If we knew the regions, we could easily identify the dynamical models, and if we knew the dynamical models we could easily segment the regions. Unfortunately, we know neither.

A. Spatial segmentation

In this section, which follows [11] and [15], we discuss the spatial-temporal segmentation problem under the assumption that the boundaries of the regions do not change with time, i.e. $\Omega_i(t) = \Omega_i$ and so $N(t) = N$. Therefore, the problem reduces to partitioning an image sequence into N *spatial* regions, each of which can be described by an ARMA model.

One can address this problem by setting up an alternating minimization procedure: starting with an initial guess of the regions, $\hat{\Omega}_i(0)$, estimate the models within each region, $\hat{A}_i(0)$, $\hat{K}_i(0)$, $\hat{C}_i(0)$, $\hat{x}_i(0)$, and then at any given time t seek for the modification of the regions $\hat{\Omega}_i(t)$, and the update of the models $\hat{A}_i(t)$, $\hat{K}_i(t)$, $\hat{C}_i(t)$, $\hat{x}_i(t)$ so as to minimize a chosen cost functional. For instance, one can minimize the norm of the innovation, integrated in space and time[2]

$$\sum_{i=1}^{N} \int_{\Omega_i} \sum_{t=n+1}^{T} (y(\boldsymbol{x}, t) - \hat{y}_i(\boldsymbol{x}, t|t-1))^2 d\boldsymbol{x}, \qquad (17)$$

where $y_i(\boldsymbol{x}, t|t-1)$ is the predictor of $y(\boldsymbol{x}, t)$ from the measurements up to time $t-1$ based on model i.

[1] That is, $\Omega = \cup_{i=1}^{N(t)} \Omega_i(t)$ and $\Omega_i(t) \cap \Omega_j(t) = \emptyset, i \neq j$.

[2] We use $y(\boldsymbol{x}, t)$ to indicate the intensity of pixel $\boldsymbol{x} \in \Omega$ at time t.

To minimize (17), one needs to choose a representation for the region boundaries $\delta\Omega_i$. In the level-set representation [24], $\Omega_i = \{\boldsymbol{x} : \phi_i(\boldsymbol{x}) > 0\}$ and $\delta\Omega_i = \{\boldsymbol{x} : \phi_i(\boldsymbol{x}) = 0\}$ for some function $\phi_i : \Omega \to \mathbb{R}$. In addition, one also needs to choose a tractable representation for the dynamical model M_i within each region Ω_i.[3] In [15], the dynamics are modeled with an AR model $y(\boldsymbol{x}, t) = \sum_{j=1}^{n} a_j^i y(\boldsymbol{x}, t-j)$, $\boldsymbol{x} \in \Omega_i$, rather than with an ARMA model as in (2).

With this representation, the spatial segmentation problem can be formulated as one of grouping regions of similar AR parameters. In [15], the grouping is obtained by minimizing a temporal generalization of the Mumford-Shah functional [23]

$$\begin{aligned} E(\{\phi_i\}, \{a_j^i\}) = &\sum_{i=1}^{N} \int_{\Omega} \lambda_i |\nabla H(\phi_i(\boldsymbol{x})| d\boldsymbol{x} + \\ &\sum_{i=1}^{N} \int_{\Omega} \sum_{t=n+1}^{T} \big(y(\boldsymbol{x}, t) - \sum_{j=1}^{n} a_j^i y(\boldsymbol{x}, t-j)\big)^2 H(\phi_i(\boldsymbol{x})) d\boldsymbol{x}, \end{aligned} \qquad (18)$$

where the Heaviside function $H(\phi)$ is defined as $H(\phi) = 1$ if $\phi \geq 0$ and $H(\phi) = 0$ otherwise. The first term in (18) aims at minimizing the length of the separating boundaries, weighted by a factor $\lambda_i \in \mathbb{R}^+$, whereas the second term is the innovation error in (17), specialized for AR models. The minimization of E proceeds by alternating between the following two steps until convergence:

1) For fixed $\{\phi_i\}$, minimization with respect to $\{a_j^i\}$ amounts to solving a linear least squares problem.

2) For fixed $\{a_j^i\}$, minimization with respect to $\{\phi_i\}$ can be implemented by gradient descent, which leads to the PDE[4]

$$\frac{\partial \phi_i(\boldsymbol{x})}{\partial t} = \lambda_i \nabla \cdot \left(\frac{\nabla \phi_i(\boldsymbol{x})}{\|\nabla \phi_i(\boldsymbol{x})\|} \right) + \sum_{t=n+1}^{T} (y(\boldsymbol{x}, t) - \sum_{j=1}^{n} a_j^i y(\boldsymbol{x}, t-j))^2. \qquad (19)$$

The right hand-side is obtained by taking the first variation of equation (18). This PDE is simulated numerically using level set methods [24], until convergence to a local minimum.

In an alternative approach [11], the dynamics within each region are described with a spatial-temporal signature

$$s(\boldsymbol{x}) = \big(\cos\theta_1(\boldsymbol{x}), \ldots, \cos\theta_n(\boldsymbol{x})\big) \qquad (20)$$

around each pixel $\boldsymbol{x}$, which depends on the subspace angles $\{\theta_j(\boldsymbol{x})\}$ between a locally computed model $M(\boldsymbol{x})$ and a reference model M_0. The functional (18) is replaced by

$$\begin{aligned} E(\{\phi_i\}, \{s_i\}) = \quad &\textstyle\sum_{i=1}^{N} \int_{\Omega} \lambda_i |\nabla H(\phi_i(\boldsymbol{x}))| d\boldsymbol{x} + \\ &\textstyle\sum_{i=1}^{N} \int_{\Omega} (s(\boldsymbol{x}) - s_i)^2 H(\phi_i(\boldsymbol{x})) d\boldsymbol{x} \end{aligned} \qquad (21)$$

and is also minimized by alternating between two steps:

1) For fixed $\{\phi_i\}$, minimization with respect to $\{s_i\}$ amounts to averaging the signatures over each region.

2) For fixed $\{s_i\}$, minimization with respect to $\{\phi_i\}$ can be implemented by a gradient descent given by:[4]

$$\frac{\partial \phi_i(\boldsymbol{x})}{\partial t} = \lambda_i \nabla \cdot \left(\frac{\nabla \phi_i(\boldsymbol{x})}{\|\nabla \phi_i(\boldsymbol{x})\|} \right) + (s(\boldsymbol{x}) - s_i)^2. \qquad (22)$$

[3] Recall that the number of pixels in each region may be large, which may prevent one from obtaining the optimal parameters within each region.

[4] For $N = 2$ regions it is better to use a single level-set function ϕ to represent both regions. This leads to a slightly different evolution equation.

Figure 7 shows segmentation results on three sequences of size $220 \times 220 \times 120$ containing two dynamic textures. For each sequence, the snapshots show the contour evolution according to the method in [11] based on signatures, starting from a circle. The segmentation is nearly perfect for the *ocean-steam* and *ocean-appearance* sequences, which have constant boundary. For the *ocean-fire* sequence, however, the method converges to the contour of an "average" region, because it computes a single segmentation from all the frames, although the true contour changes with time. The method in [15] yields similar results in all sequences, however with a much smaller order for the ARX models ($n = 2$ versus $n = 10$). Therefore, one may apply the method in [15] to a moving window of frames, and obtain one segmentation per frame. As can be seen from Figure 8, this give a much more accurate segmentation of the moving boundary. Therefore, the approach shows robustness also to changes in the original hypotheses that dynamic textures were spatially stationary.

B. Temporal segmentation

Instead of partitioning the spatial domain into regions of constant statistics, one can partition the temporal domain. For instance, in a news video sequence, the host could be interviewing a guest and the camera may be switching between the host, the guest and both of them, as shown in Figure 9. Given the frames $\{I(t) \in \mathbb{R}^m\}_{t=1}^T$, we would like to cluster them according to the different events. This problem is a particular case of the spatial-temporal segmentation problem in which there is a single region at each time instant ($N(t) = 1$ and $\Omega_i(t) = \Omega$). As such, this problem is very much related to filtering and identification of hybrid systems [2], [14], [18], [34], [21], [31], where each discrete state corresponds to a different event in the video.

Here, we adapt the algebraic approach for identification of switched ARX (SARX) models in [21] and apply it to temporal video segmentation. We assume that pixel intensities obey a SAR model $y(\boldsymbol{x}, t) = \sum_{j=1}^n a_j(q_t) y(\boldsymbol{x}, t-j)$ with discrete state $q_t = 1, \ldots, N$. If we let $b_i = (a_n(i), \ldots, a_1(i), 1)$ and $z_t(\boldsymbol{x}) = (y(\boldsymbol{x}, t-n), \cdots, y(\boldsymbol{x}, t-1), -y(\boldsymbol{x}, t))$, then for each $\boldsymbol{x}$ and t there is a j such that $b_i^T z_t(\boldsymbol{x}) = 0$. Therefore,

$$\forall\, \boldsymbol{x}, t \quad p_N(z_t(\boldsymbol{x})) = \prod_{i=1}^N (b_i^T z_t(\boldsymbol{x})) = 0. \tag{23}$$

This N-degree polynomial can be written linearly in terms of a vector of coefficients $\boldsymbol{h}$ and a vector of monomials $\nu_N(z_t(\boldsymbol{x}))$ as $\boldsymbol{h}^T \nu_N(z_t(\boldsymbol{x})) = 0$. Therefore, one may compute $\boldsymbol{h}$ linearly from the image data. As shown in [21], the parameters of the AR model active at time t can be computed from the gradient of p_N at any pixel $\boldsymbol{x}$ as

$$b_i = \nabla p_N(z_t(\boldsymbol{x})). \tag{24}$$

Figure 9 shows segmentation results reported in [32] on the temporal segmentation of two video sequences with three temporal events. Since the number of pixels is large, the images were projected onto the first three principal components, and the algorithm was applied to the projected data. A perfect segmentation was obtained for both sequences.

C. Spatial-temporal segmentation

In this section, which follows [30], [31], we consider segmentation both in space and in time under the assumption that the model within each spatial-temporal region is an AR model. This gives rise to a spatial-temporal SAR model for the image intensities $y(\boldsymbol{x}, t) = \sum_{j=1}^n a_j(q_t(\boldsymbol{x})) y(\boldsymbol{x}, t-j)$. Notice that the main difference with the previous section is that the discrete state $q_t(\boldsymbol{x})$ depends on the pixel coordinates. Therefore, we may apply the same algorithms as in the previous section. However, rather than obtaining a single model for each time instant, we will obtain several models. By clustering all the models at time t into $N(t) \leq N$ groups using k-means, one obtains the segmentation in space of the frame at time t. This batch method can incorporate temporal coherence by computing an estimate $\hat{\boldsymbol{h}}(t)$ of $\boldsymbol{h}$ from the measurements up to time t. This estimate is updated using a normalized gradient recursive identification algorithm [31].

Figure 10 compares the batch method applied to a moving window of 5 frames and the recursive method on a video sequence with a variable number of models. The video displays a bird swimming on water. The camera moves up at 1 pixel/frame until the bird disappears at $t = 51$. Then, the camera stays stationary from $t = 56$ to $t = 66$. Finally, the camera moves down at 1 pixel/frame, and the bird reappears at $t = 76$. Note that both methods give good results for the first frames containing both the bird and the water. Nevertheless, the batch method performs poorly when the bird disappears, because it overestimates the number of models, whereas the recursive method is robust. When the bird reappears, the recursive method detects the bird correctly, while the batch method produces a wrong segmentation for the first few frames after the bird reappears.

V. Conclusions

We have presented several examples where current algorithms for system identification can be successfully employed to address modeling, synthesis and recognition problems in computer vision. These include modeling dynamic textures and human gaits for the purpose of synthesis and classification or recognition. In addition, we have indicated several directions where further work in system identification is needed in order to address difficult tasks of modeling non-Gaussian, non-linear, non-stationary processes for detection, classification, recognition and segmentation.

References

[1] G. Aggarwal, A. Roy-Chowdhury, and R. Chellappa. A system identification approach for video-based face recognition. In *International Conference on Pattern Recognition*, pages 23-26, 2004.

[2] A. Bemporad, A. Garulli, S. Paoletti, A. Vicino. A greedy approach to identification of piecewise affine models. In *HSCC*, pages 97–112, 2003.

[3] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of Calcutta Mathematical Society*, 35:99–109, 1943.

[4] A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *CVPR*, volume 2, pages 52–58, 2001.

[5] C. Bregler. Learning and recognizing human dynamics in video sequences. In *CVPR*, pages 568–574, 1997.

[6] A. Chan and N. Vasconcelos. Mixtures of dynamic textures. In *IEEE International Conference on Computer Vision*, pages 641–647, 2005.

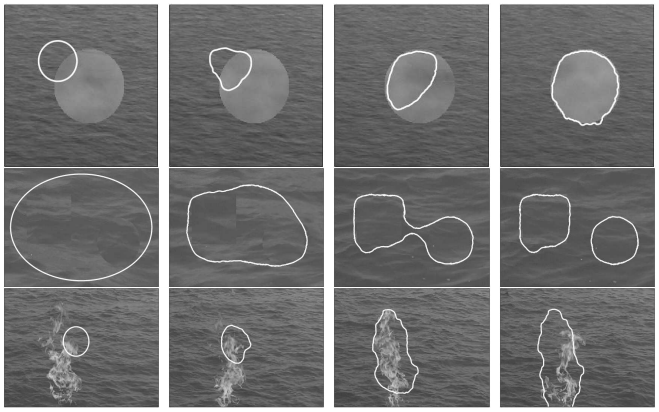


Fig. 7. Video segmentation using spatial-hybrid systems. Top: segmenting steam versus water. Middle: segmenting rotated water versus water. Bottom: segmenting fire versus water. Animation of these results can be downloaded from *http://vision.ucla.edu/projects.html.* (Figure courtesy of [10])

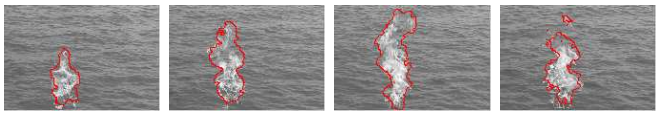


Fig. 8. Segmentation of the *ocean-fire* sequence using spatial hybrid systems. (Figure courtesy of [15])

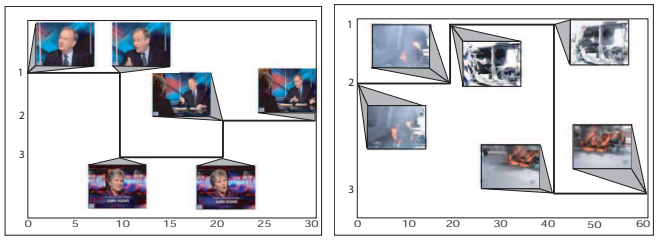
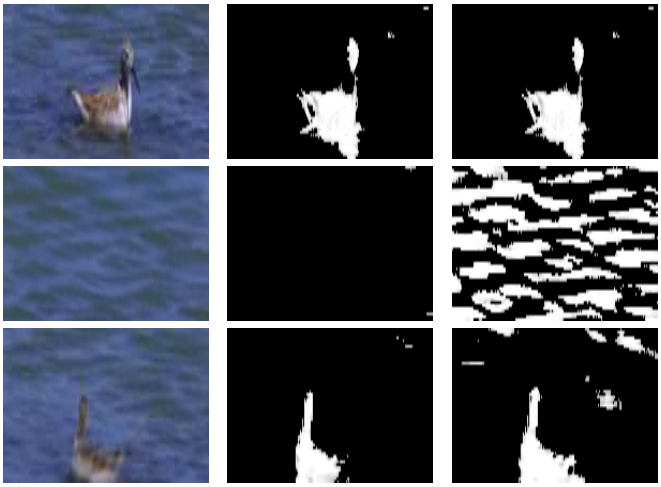


Fig. 9. Video segmentation using temporal hybrid systems. Left: news video segmented into 3 groups: interviewer, interviewee, and both. Right: news video from Iraq segmented into 3 groups: rear of a car with a burning wheel, a burnt car with people, and a burning car. (Figure courtesy of [32])



Sequence Method in [30], [31] Method in [32]

Fig. 10. Video segmentation using spatial-temporal hybrid systems: a bird floating on water (top row) that disappears (middle row) and reappears (bottom row) in the scene. Left: frames of the video sequence. Middle: segmentation using spatial-temporal hybrid ID. Right: segmentation of the bird (white) from the water (black) using GPCA. (Figure courtesy of [30])

[7] K. De Cock and B. De Moor. Subspace angles and distances between ARMA models. *System and Control Letters*, 46(4):265–270, 2002.

[8] L. Cooper, J. Liu, and Kun Huang. Spatial segmentation of temporal texture using mixture linear models. In *Workshop on Dynamical Vision*, LNCS 4358, pages 142–150, 2005.

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

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

[11] G. Doretto, D. Cremers, P. Favaro, and S. Soatto. Dynamic texture segmentation. In *IEEE Conf. on Computer Vision*, pages 44–49, 2003.

[12] G. Doretto and S. Soatto. Editable dynamic textures. In *CVPR*, volume II, pages 137–142, 2003.

[13] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis*. Wiley and Sons, 1973.

[14] G. Ferrari-Trecate, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems. *IEEE Transactions on Automatic Control*, 47(10):1663–1676, 2002.

[15] A. Ghoreyshi and R. Vidal. Segmenting dynamic textures with Ising descriptors, ARX models and level sets. In *Workshop on Dynamic Vision*, LNCS 4358, pages 127–141, 2006.

[16] E. J. Hannan and M. Deistler. *The statistical theory of linear systems*. Wiley and Sons.

[17] K. Huang, A. Wagner, and Y. Ma. Identification of hybrid linear time-invariant systems via subspace embedding and segmentation. In *IEEE Conference on Decision & Control*, pages 3227- 3234, 2004.

[18] A. Juloski, S. Weiland, and M. Heemels. A Bayesian approach to identification of hybrid systems. In *IEEE Conference on Decision & Control*, pages 13- 19, 2004.

[19] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79-86, 1951.

[20] L. Ljung. *System Identification: theory for the user*. Prentice Hall, 1987.

[21] Y. Ma and R. Vidal. Identification of deterministic switched ARX systems via identification of algebraic varieties. In *HSCC*, pages 449–465. 2005.

[22] A. Martin. A metric for ARMA processes. *IEEE Trans. on Signal Processing*, 48(4):1164–1170, 2000.

[23] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1988.

[24] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi equations. *Journal of Computational Physics*, 79:12–49, 1988.

[25] P. Van Overschee and B. De Moor. Subspace algorithms for the stochastic identification problem. *Automatica*, 29(3):649–660, 1993.

[26] P. Saisan, A. Bissacco, A. Chiuso, and S. Soatto. Modeling and synthesis of facial motion driven by speech. In *ECCV*, pages 456–467, 2004.

[27] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *CVPR*, pages 58–63, 2001.

[28] A. Schödl, R. Szeliski, D. Salesin and I. Essa. Video Textures. *SIGGRAPH*, pages 489-498, 2000.

[29] M. Szummer and R. W. Picard. Temporal Texture Modeling. In *IEEE International Conference on Image Processing*, pages 823-826, 1996.

[30] R. Vidal. Online clustering of moving hyperplanes. In *Neural Information Processing Systems, NIPS*, 2006.

[31] R. Vidal. Recursive identification of switched ARX systems. In *Automatica*, 2007.

[32] R. Vidal, Y. Ma, and S. Sastry. Generalized Principal Component Analysis (GPCA). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1–15, 2005.

[33] R. Vidal and A. Ravichandran. Optical flow estimation and segmentation of multiple moving dynamic textures. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 516–521, 2005.

[34] R. Vidal, S. Soatto, Y. Ma, and S. Sastry. An algebraic geometric approach to the identification of a class of linear hybrid systems. In *IEEE Conference on Decision & Control*, pages 167–172, 2003.

[35] S.V.N. Vishwanathan, A. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 2006.

[36] Lu Yuan, Fang Wen, Ce Liu, and Heung-Yeung Shum. Synthesizing dynamic texture with closed-loop linear dynamic system. In *European Conference on Computer Vision*, pages 603–616, 2004.

[37] S. C. Zhu and Y. Z. Wang. A generative method for textured motion: analysis and synthesis. In *ECCV*, pages 583–598, 2002.