A System Theoretic Approach to Synthesis and Classification of Lip Articulation

Hasan Ertan Çetingül, Rizwan Ahmed Chaudhry, and René Vidal

Center for Imaging Science, Johns Hopkins University, Baltimore MD 21218, USA

Abstract. We present a system for synthesizing lip movements and recognizing speakers/phrases from visual lip sequences. Low-dimensional geometrical lip features, such as trajectories of landmarks on the outer lip contour and vertical distances for the mouth opening are first extracted from the images. The temporal evolution of these features is modeled with linear dynamical systems, whose parameters are learned using system identification techniques. By carefully exploiting physical constraints of lip movement both in the learning and synthesis stages, realistic synthesis of novel sequences is achieved. Recognition is performed using classification methods, such as nearest neighbors and support vector machines, combined with various metrics based on subspace angles and kernels, such as the Binet-Cauchy, Martin, and Kullback-Leibler kernels. Experiments are designed to find the combination of features, identification method, kernel and classification method that is most appropriate for synthesis and classification of lip articulation.

1 Introduction

Recently, the analysis of lip articulation has attracted widespread interest from the computer vision community due to several applications in biometrics, audio-visual speech synthesis, rehabilitation engineering and virtual reality. Motion capture technology plus complex 3D facial models are key ingredients in animations [15], whereas visual text-to-speech (VTTS) and viseme-based HMMs have been extensively used in audio-visual speech synthesis (see references in [9]).

In speaker/speech recognition, state-of-the art systems employ both lip movement and audio information in a unified framework [7]. However, most audio-visual biometric systems combine a simple visual modality with a sophisticated audio modality. Systems employing enhanced visual information are quite limited. This is due to several reasons: (i) Lip feature extraction and tracking are complex tasks (see [4] and references therein). Indeed, the performance of existing lip feature extraction techniques depends on acquisition specifics such as image quality, resolution, head pose and illumination conditions. (ii) Current methods are limited to the use of geometric, parametric and motion features for speaker/speech recognition with well-known clustering methodologies such as PCA, LDA, and HMMs (see references in [7, 4]). These methods do not use lip articulation judiciously, although it contains useful information for classification.

On a completely parallel track, several system-theoretic techniques have recently been used in the computer vision community for modeling dynamic visual processes. For instance, [10, 23] 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; [3] uses ARMA models to represent human gaits, such as walking, running, jumping, etc.; [1] uses ARMA models to describe the appearance of moving faces; and [19] uses ARMA models to represent audio-visual lip articulation. Given a video sequence, one can use standard system identification techniques, e.g., subspace identification [18], to learn the parameters of the ARMA model. Given a model, novel sequences can be synthesized by simulating the model forward. Impressive synthesis of dynamic textures has been demonstrated by a number of papers [10, 11, 23, 21]. The same ideas have also been used for synthesis of lip articulation using speech as the driving input [19].

As it turns out, the identified parameters can also be used for recognition. For instance, if one defines a distance on the space of linear dynamical systems (LDSs), then recognition of novel sequences can be performed using k-nearest neighbor (k-NN) classification. This approach has been tested in [20] for recognition of dynamic textures from intensity trajectories, in [3] for recognition of human gaits from motion capture trajectories, and in [1] for recognition of faces from intensity, color and landmark trajectories. As the space of LDSs is not Euclidean, a key challenge to this recognition approach is the definition of a distance between two LDSs. The works of [3, 20, 1] use distances built from the subspace angles among the observability spaces associated with the LDSs [16, 8], while the work of [10] uses the Kullback-Leibler divergence between the probability distributions of the associated output processes. Another approach to comparing LDSs is to use kernels, which can be combined not only with k-NN, but also with support vector machines (SVM). In [5], a probabilistic kernel based on the Kullback-Leibler divergence is introduced and used for classification of dynamic textures. In [22], an entire family of kernels based on the Binet-Cauchy theorem is proposed to compare LDSs. However, classification results have not yet been reported.

Our main goal is to develop a system for synthesis and classification of lip articulation in video sequences, including the comparison of several choices for features, identification methods, synthesis methods, distances and kernels for comparing LDSs, and classification methods. Although we will follow the framework of [3, 20, 1, 5, 22], there are several challenges that need to be considered:

1. *Feature selection*: Unlike previous work, we cannot rely on intensity or color trajectories. Also, we will not use motion capture data. Therefore, good feature selection and tracking methods for lip articulation are needed.
2. *Identification method*: As the data dimension for learning dynamic textures is the number of pixels, prior work had to resort to the PCA-based identification method [10], rather than N4SID [18]. For lip articulation, one can use both N4SID and the PCA-based method. Nevertheless, our experiments will show that, contrary to intuition, the PCA-based method performs better.
3. *Synthesis method*: While synthesis of dynamic textures with LDSs has been quite a success, realistic synthesis of other visual processes is not straightforward, as it depends on the choice of the features, the stability of the

dynamical models, etc. Our experiments will show that good synthesis can only be achieved when physical constraints of articulation are incorporated.

4. *Distances, kernels, and classification methods*: Most prior work used k-NN classification with a small number of distances based on subspace angles. Our work includes both k-NN and SVM classification combined with a wide variety of distances and kernels. In particular, this is the first time SVMs are combined with Binet-Cauchy kernels for the classification of lip articulation.

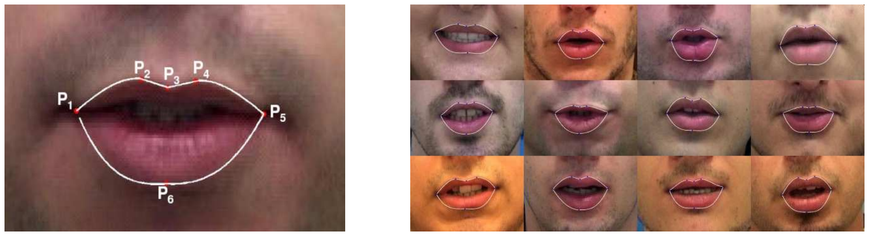
This paper is organized as follows: §2 shows how to extract features for representing lip articulation; §3 describes the identification of LDSs and their use for synthesis; §4 presents several distances and kernels for LDSs, and their use for recognition; §5 presents experimental results; and §6 gives the conclusions.

2 Representation of lip articulation

We first describe the extraction of geometric features representing lip articulation. For this purpose, we employ a 3-stage processing system proposed in [4].

The first stage eliminates natural head motion during the speaking act to obtain pure lip movement. Each frame of each talking face sequence is aligned with the first frame using a 2D parametric motion estimator. For each pair of consecutive frames, global head motion parameters are calculated using hierarchical Gaussian image pyramids and the 12-parameter quadratic motion model. This model is then used to back-warp the frames and extract corrected lip frames.

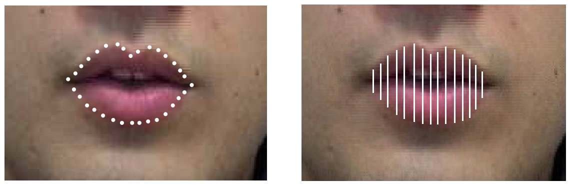
In the second stage, the quasi-automatic technique proposed in [13] is used to extract the outer lip contour. The manual selection of a point above the mouth serves as the initialization of a snake-like algorithm that finds several points in the upper-lip boundary. From these points, the three key points p_2, p_3 and p_4 forming the cupids bow are automatically extracted (see Fig. 1(a)) and the two line segments joining them are computed. Pseudo-hue gradient information is then used to locate the key point in the lower lip boundary, p_6, and the lip corners p_1 and p_5. Least-squares optimization is used to fit four cubic polynomials using five of the key points as junctions. All the key points are then tracked in consecutive frames and the curve fitting steps are repeated. Fig. 1(b) displays examples of lip contours extracted from the images in the database.



(a) The 6 key points on the lip contour (b) Extracted lip contours

Fig. 1. Lip contour extraction.

Once the six key points have been detected and tracked, the number of landmarks is increased to 32 per frame by carefully selecting contour points at regular intervals on the fitted curves. The coordinates of these points $(x^t_{p_i}, y^t_{p_i})$ are then used to create three sets of lip features, denoted by $\{\mathcal{L}_{xy}, \mathcal{L}, \mathcal{D}\}$. The first set, $\mathcal{L}_{xy}$, is formed by stacking the landmark coordinates into two 32-dimensional feature vectors at time t, $\boldsymbol{y}^{\mathcal{L}_x}_t = \left[x^t_{p_1}\ x^t_{p_2} \cdots x^t_{p_{32}}\right]^\top$ and $\boldsymbol{y}^{\mathcal{L}_y}_t = \left[y^t_{p_1}\ y^t_{p_2} \cdots y^t_{p_{32}}\right]^\top$. The second set of features, $\mathcal{L}$, is generated by simply concatenating $\boldsymbol{y}^{\mathcal{L}_x}_t$ and $\boldsymbol{y}^{\mathcal{L}_y}_t$, i.e. $\boldsymbol{y}^{\mathcal{L}}_t = \begin{bmatrix}\boldsymbol{y}^{\mathcal{L}_x}_t \\ \boldsymbol{y}^{\mathcal{L}_y}_t\end{bmatrix}$. The third set of features, $\mathcal{D}$, corresponds to the 15 vertical distances l_i from the equidistant upper lip landmarks to the lower lip boundary. The resulting feature vector is $\boldsymbol{y}^{\mathcal{D}}_t = \left[l_1\ l_2 \ldots l_{15}\right]^\top$. Fig. 2(a) shows the 32 landmark points whereas Fig. 2(b) illustrates the 15 lip distance features.



(a) 32 landmark features (b) 15 distance features

Fig. 2. Landmark and distance based lip features.

3 Synthesis of lip articulation

We model the temporal evolution of different lip features using a dynamical system framework. More specifically, a sequence of feature trajectories $\{\boldsymbol{y}_t\}$ is assumed to be a realization from a second-order stationary stochastic process and hence it can be modeled with a state space model

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{v}_t; \quad \boldsymbol{B}\boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}), \\ \boldsymbol{y}_t &= \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{\mu} + \boldsymbol{w}_t; \qquad \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}).\end{aligned} \tag{1}$$

$\boldsymbol{x}_t \in \mathbb{R}^n$ and $\boldsymbol{y}_t \in \mathbb{R}^m$ are respectively the state and output variables at time t. $\boldsymbol{v}_t$ and $\boldsymbol{w}_t$ are respectively the state and measurement noise processes, which are assumed to be stationary and zero mean i.i.d. Gaussian processes with covariances $\Sigma_{\boldsymbol{v}_t} = \boldsymbol{I}_p$ and $\boldsymbol{R}$, where $\boldsymbol{I}_p$ is the identity matrix of dimensions $p \times p$. $(\boldsymbol{x}_0, \boldsymbol{\mu}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{R})$ are the parameters of the LDS with $\boldsymbol{x}_0 \in \mathbb{R}^n$ being the initial state, $\boldsymbol{\mu} \in \mathbb{R}^m$ being the mean of $\boldsymbol{y}_t$, $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times p}$, $\boldsymbol{C} \in \mathbb{R}^{m\times n}$, and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$. We assume that $\boldsymbol{B}\boldsymbol{B}^\top = \boldsymbol{Q} \in \mathbb{R}^{n\times n}$. It is also typical to take the output dimension to be greater than the system order, i.e. $m > n$.

3.1 ARMA system identification

Given a sequence of measurements $\{\boldsymbol{y}_t\}^{\tau-1}_{t=0}$ generated by a linear dynamical system, the task is to identify the system parameters $(\boldsymbol{x}_0, \boldsymbol{\mu}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{R})$. Subspace identification methods, such as N4SID [18], provide asymptotically optimal estimates of the model parameters in the maximum likelihood sense if the measurements are generated by stochastic linear systems [2]. We do not provide mathematical details of N4SID in this paper and refer the reader to [18].

However, N4SID becomes impractical when the output dimension is high, which is usually the case when intensity values of an image sequence are used as measurements, e.g., in dynamic textures. The PCA-based approach introduced in [10] identifies the system parameters $(\boldsymbol{x}_0, \boldsymbol{\mu}, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \sigma^2)$ assuming $\boldsymbol{R} = \sigma^2 \boldsymbol{I}_m$. Briefly, this method first estimates $(\boldsymbol{\mu}, \boldsymbol{x}_t, \boldsymbol{C}, \sigma^2)$ by neglecting the relationship between $\boldsymbol{x}_{t+1}$ and $\boldsymbol{x}_t$. Therefore, the equation $\boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{\mu} + \boldsymbol{w}_t$ becomes a PCA model, where $\boldsymbol{\mu}$ is the mean of $\boldsymbol{y}_t$, $\boldsymbol{C}$ is a basis for the subspace spanned by $\{\boldsymbol{y}_t - \boldsymbol{\mu}\}$ with coefficients $\boldsymbol{x}_t$. By assuming a canonical model where $\boldsymbol{C}^\top \boldsymbol{C} = \boldsymbol{I}_n$, one obtains the estimates $\hat{\boldsymbol{\mu}} = \bar{\boldsymbol{y}} \doteq \frac{1}{\tau}\sum_{i=0}^{\tau-1} \boldsymbol{y}_i$, $\hat{\boldsymbol{C}} = \boldsymbol{U}$ and $\hat{\boldsymbol{X}}_0^{\tau-1} = \boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the singular value decomposition (SVD) of the mean-subtracted output matrix $\boldsymbol{Y}_0^{\tau-1} \doteq \left[\boldsymbol{y}_0 - \bar{\boldsymbol{y}}, \ldots, \boldsymbol{y}_{\tau-1} - \bar{\boldsymbol{y}}\right]$ and $\hat{\boldsymbol{X}}_{t_0}^{t_f} = \left[\hat{\boldsymbol{x}}_{t_0}, \ldots, \hat{\boldsymbol{x}}_{t_f}\right]$ is the matrix of state estimates for $t = t_0, \ldots, t_f$. Note that $\hat{\sigma}^2 = var(\boldsymbol{Y}_0^{\tau-1} - \hat{\boldsymbol{C}}\hat{\boldsymbol{X}}_0^{\tau-1})$ and $\hat{\boldsymbol{x}}_0 = \hat{\boldsymbol{X}}_0^0$. Then, the transition matrix can be computed as $\hat{\boldsymbol{A}} = \hat{\boldsymbol{X}}_1^{\tau-1}(\hat{\boldsymbol{X}}_0^{\tau-2})^\dagger$, where $\boldsymbol{Z}^\dagger$ denotes the pseudo-inverse of $\boldsymbol{Z}$. The estimate of $\boldsymbol{Q}$ is given by $\hat{\boldsymbol{Q}} = \frac{1}{\tau-1}\sum_{i=0}^{\tau-2} \hat{\boldsymbol{u}}_i \hat{\boldsymbol{u}}_i^\top$, where $\hat{\boldsymbol{u}}_t := \hat{\boldsymbol{x}}_{t+1} - \hat{\boldsymbol{A}}\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{B}}\hat{\boldsymbol{B}}^\top = \hat{\boldsymbol{Q}}$. See [3, 10] for a detailed explanation of the algorithm along with MATLAB implementations.

3.2 Synthesis with ARMA models

Given the identified model parameters $(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}, \hat{\boldsymbol{C}}, \hat{\boldsymbol{R}})$, one can generate new lip articulation data $\tilde{\boldsymbol{Y}}_0^{\tau-1} = \left[\tilde{\boldsymbol{y}}_0, \tilde{\boldsymbol{y}}_1, \ldots, \tilde{\boldsymbol{y}}_{\tau-1}\right]$ based on the learned dynamical model. However, the estimated parameters, and hence the corresponding synthesis procedures, may differ depending on the identification approach used.

It is worth noticing that N4SID determines state sequences that are outputs of non-steady state Kalman filter banks [18]. Hence it estimates the so-called *Kalman gain*, $\boldsymbol{K}$, instead of $\boldsymbol{B}$. When the parameters $(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{K}}, \hat{\boldsymbol{C}}, \hat{\boldsymbol{Q}})$ are identified using N4SID, for accurate synthesis, the model needs to be driven by the noise process $\boldsymbol{e}_t \sim \mathcal{N}(\boldsymbol{0}, \hat{\boldsymbol{Q}})$,

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1} &= \hat{\boldsymbol{A}}\hat{\boldsymbol{x}}_t + \hat{\boldsymbol{K}}\boldsymbol{e}_t, \\ \tilde{\boldsymbol{y}}_t &= \hat{\boldsymbol{C}}\hat{\boldsymbol{x}}_t + \hat{\boldsymbol{\mu}}. \end{aligned} \tag{2}$$

On the other hand, when $(\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}, \hat{\boldsymbol{C}}, \hat{\boldsymbol{R}})$ are identified using the PCA-based method, lip movements can be synthesized by driving the system with the noise process $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_p)$, and $p = n$,

$$\begin{aligned} \hat{\boldsymbol{x}}_{t+1} &= \hat{\boldsymbol{A}}\hat{\boldsymbol{x}}_t + \hat{\boldsymbol{B}}\boldsymbol{z}_t, \\ \tilde{\boldsymbol{y}}_t &= \hat{\boldsymbol{C}}\hat{\boldsymbol{x}}_t + \hat{\boldsymbol{\mu}}. \end{aligned} \tag{3}$$

However, when applying these procedures to landmarks, it is important to take the physics of the process into account. Since the vertical movement has larger variance than the horizontal movement during a natural speaking act, it is useful to separately identify the $x-$ and $y-$ trajectories, i.e. $\{\mathcal{L}_x, \mathcal{L}_y\}$, as $(\hat{\boldsymbol{x}}_0{}^{\mathcal{L}_x}, \hat{\boldsymbol{\mu}}^{\mathcal{L}_x}, \hat{\boldsymbol{A}}^{\mathcal{L}_x}, \hat{\boldsymbol{B}}^{\mathcal{L}_x}, \hat{\boldsymbol{C}}^{\mathcal{L}_x}, \hat{\boldsymbol{R}}^{\mathcal{L}_x})$ and $(\hat{\boldsymbol{x}}_0{}^{\mathcal{L}_y}, \hat{\boldsymbol{\mu}}^{\mathcal{L}_y}, \hat{\boldsymbol{A}}^{\mathcal{L}_y}, \hat{\boldsymbol{B}}^{\mathcal{L}_y}, \hat{\boldsymbol{C}}^{\mathcal{L}_y}, \hat{\boldsymbol{R}}^{\mathcal{L}_y})$, and synthesize new trajectories with parallel addition of them:

$$\begin{aligned}
\begin{bmatrix} \hat{\boldsymbol{x}}_{t+1}^{\mathcal{L}_x} \\ \hat{\boldsymbol{x}}_{t+1}^{\mathcal{L}_y} \end{bmatrix} &= \begin{bmatrix} \hat{\boldsymbol{A}}^{\mathcal{L}_x} & \boldsymbol{0} \\ \boldsymbol{0} & \hat{\boldsymbol{A}}^{\mathcal{L}_y} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_{t}^{\mathcal{L}_x} \\ \hat{\boldsymbol{x}}_{t}^{\mathcal{L}_y} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{B}}^{\mathcal{L}_x} \\ \hat{\boldsymbol{B}}^{\mathcal{L}_y} \end{bmatrix} \boldsymbol{z}'_t, \\
\begin{bmatrix} \tilde{\boldsymbol{y}}_{t}^{\mathcal{L}_x} \\ \tilde{\boldsymbol{y}}_{t}^{\mathcal{L}_y} \end{bmatrix} &= \begin{bmatrix} \hat{\boldsymbol{C}}^{\mathcal{L}_x} \\ \hat{\boldsymbol{C}}^{\mathcal{L}_y} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_{t}^{\mathcal{L}_x} \\ \hat{\boldsymbol{x}}_{t}^{\mathcal{L}_y} \end{bmatrix} + \begin{bmatrix} \hat{\boldsymbol{\mu}}^{\mathcal{L}_x} \\ \hat{\boldsymbol{\mu}}^{\mathcal{L}_y} \end{bmatrix},
\end{aligned} \tag{4}$$

where $\boldsymbol{z}'_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{2p})$, and $p = n$.

4 Classification of lip articulation

Given a training set of models $\{\boldsymbol{M}_i\}_{i=1}^N$ with parameters $(\hat{\boldsymbol{x}}_{0;i}, \hat{\boldsymbol{\mu}}_i, \hat{\boldsymbol{A}}_i, \hat{\boldsymbol{B}}_i, \hat{\boldsymbol{C}}_i, \hat{\boldsymbol{R}}_i)$, and the membership of these models to a number of classes, the goal is to classify new models according to the different classes for the purpose of recognition. To that end, we employ nearest neighbor classifiers or support vector machines together with various metrics on the space of dynamical systems.

4.1 Nearest neighbor classification and subspace angles

Nearest-neighbor classification is a method that assigns each point to the class of its nearest neighbors via a majority vote. The simplest case is the 1-NN method, where given a training set $\{(\boldsymbol{s}_1, g_1), (\boldsymbol{s}_2, g_2), \ldots, (\boldsymbol{s}_N, g_N)\}$, the group membership, g of the input sample vector, $\boldsymbol{s}$ is determined as $g = g_j$, where $j = \mathrm{argmin}_{i=1,\ldots,N} \|\boldsymbol{s} - \boldsymbol{s}_i\|$. Notice that $\boldsymbol{s}_j$ is the nearest neighbor of $\boldsymbol{s}$ according to the metric $\|\cdot\|$.

In order to apply nearest neighbor classification to LDSs, we need a metric in the space of models. A common distance for comparing ARMA models is based on subspace angles between observability subspaces [8]. Given a model $\boldsymbol{M}_i$ specified by $(\boldsymbol{A}_i, \boldsymbol{C}_i)$, one can define its extended observability matrix

$$\mathcal{O}_\infty(\boldsymbol{M}_i) = [\boldsymbol{C}_i^\top, (\boldsymbol{C}_i\boldsymbol{A}_i)^\top, (\boldsymbol{C}_i\boldsymbol{A}_i^2)^\top, \cdots]^\top \in \mathbb{R}^{\infty \times n} \tag{5}$$

and use it to compare two ARMA models $\boldsymbol{M}_i$ and $\boldsymbol{M}_j$ by computing the angles between the range spaces of $\mathcal{O}_\infty(\boldsymbol{M}_i)$ and $\mathcal{O}_\infty(\boldsymbol{M}_j)$. In order to calculate the subspace angles, each model $\boldsymbol{M}$ is converted into the *forward innovation* form:

$$\begin{aligned} \boldsymbol{x}'_{t+1} &= \boldsymbol{A}\boldsymbol{x}'_t + \boldsymbol{K}\boldsymbol{e}_t, \\ \boldsymbol{y}_t &= \boldsymbol{C}\boldsymbol{x}'_t + \boldsymbol{e}_t, \end{aligned} \tag{6}$$

where $\boldsymbol{K}$ is the Kalman gain and $\boldsymbol{x}'_t$ is the transformed state. The Kalman gain can be computed as $\boldsymbol{K} = (\boldsymbol{APC}^\top)(\boldsymbol{CPC}^\top + \boldsymbol{I})^{-1}$, where $\boldsymbol{P}$ is obtained by solving the following algebraic Riccati equation

$$\boldsymbol{P} = \boldsymbol{APA}^\top - (\boldsymbol{APC}^\top)(\boldsymbol{CPC}^\top + \boldsymbol{I})^{-1}(\boldsymbol{APC}^\top)^\top + \boldsymbol{Q}. \tag{7}$$

Then the computation of the subspace angles between the ARMA models $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ boils down to the following steps:

1. Solve the Lyapunov equation $\mathcal{A}^\top \mathcal{Q} \mathcal{A} - \mathcal{Q} = -\mathcal{C}^\top \mathcal{C}$ for $\mathcal{Q} = \begin{bmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} \end{bmatrix}$, where
$$\mathcal{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2 - \boldsymbol{K}_2\boldsymbol{C}_2 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_1 - \boldsymbol{K}_1\boldsymbol{C}_1 \end{bmatrix} \quad \text{and} \quad \mathcal{C} = \begin{bmatrix} \boldsymbol{C}_1 & -\boldsymbol{C}_2 & \boldsymbol{C}_2 & -\boldsymbol{C}_1 \end{bmatrix}.$$
2. Compute the eigenvalues $\{\lambda_i\}_{i=1}^{4n}$ of
$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{Q}_{11}^{-1}\boldsymbol{Q}_{12} \\ \boldsymbol{Q}_{22}^{-1}\boldsymbol{Q}_{21} & \boldsymbol{0} \end{bmatrix}.$$
3. The $2n$ largest eigenvalues are equal to the cosines of the subspace angles $\{\theta_i\}_{i=1}^{2n}$ between $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$.

Given the subspace angles $\{\theta_i\}_{i=1}^{2n}$ between the observability subspaces of two dynamical systems, one can define various distances between the two ARMA models. The *Finsler* distance d_F, the *Martin* distance d_M, and the *Frobenius* distance d_f between $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are defined as:

$$\begin{aligned} d_F(\boldsymbol{M}_1, \boldsymbol{M}_2) &= \theta_{\max}, \\ d_M(\boldsymbol{M}_1, \boldsymbol{M}_2)^2 &= -\ln \prod_{i=1}^{2n} \cos^2 \theta_i, \\ d_f(\boldsymbol{M}_1, \boldsymbol{M}_2)^2 &= 2 \sum_{i=1}^{2n} \sin^2 \theta_i. \end{aligned} \tag{8}$$

The reader is referred to [8] for theoretical and to [3] for implementation details.

4.2 Support vector machines and kernels

A linear SVM classifies the training samples $\{(\boldsymbol{s}_1, g_1), \ldots, (\boldsymbol{s}_N, g_N)\}$ by finding a *hyperplane* $\{\boldsymbol{s} | \boldsymbol{w}^\top \boldsymbol{s} + b = 0\}$ that maximizes the distance to all data points $\{\boldsymbol{s}_i \in \mathcal{S}\}$. When the data are linearly separable, the separating hyperplane can be computed as $\boldsymbol{w} = \sum \alpha_i \boldsymbol{s}_i$, where $\alpha_i \neq 0$ only for a few *support vectors* $\{\boldsymbol{s}_i\}$. Given $(\boldsymbol{w}, b)$, a new point $\boldsymbol{s}$ is classified as $g = \text{sign}(\sum \alpha_i \boldsymbol{s}_i^\top \boldsymbol{s} + b)$.

When the data are not linearly separable, one can use slack variables to achieve separability. Alternatively, one can find an embedding $\varPhi : \mathcal{S} \to \mathcal{F}$ into a *feature space* $\mathcal{F}$ where a separating hyperplane exists. Since $\boldsymbol{w} = \sum \alpha_i \varPhi(\boldsymbol{s}_i)$, a new point $\boldsymbol{s}$ is classified as $g = \text{sign}(\sum \alpha_i k(\boldsymbol{s}, \boldsymbol{s}_i) + b)$, where $k : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the *kernel* $k(\boldsymbol{s}_i, \boldsymbol{s}_j) = \varPhi(\boldsymbol{s}_i)^\top \varPhi(\boldsymbol{s}_j)$. Therefore, classification may be performed by defining a kernel, without having to specifically compute the embedding $\varPhi$.

In order to apply SVM classification to LDSs, we need to define a kernel in the space of models. Over the past few years, different kernels for LDSs have been proposed. The *Martin* kernel [16] is defined from the Martin distance among the observability subspaces between two dynamical systems $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ as

$$k_M(\boldsymbol{M}_1, \boldsymbol{M}_2) = e^{-\gamma(d_M(\boldsymbol{M}_1,\boldsymbol{M}_2))^2}, \tag{9}$$

where γ is a parameter, which is equal to 1 in the original definition in [16].

In [5], Chan and Vasconcelos proposed another metric based on the Kullback-Leibler (KL) divergence $D(P_{\boldsymbol{Y}_1}\|P_{\boldsymbol{Y}_2})$ between the probability distributions $P_{\boldsymbol{Y}_1}$ and $P_{\boldsymbol{Y}_2}$ of the output processes $\boldsymbol{Y}_1 \doteq \{\boldsymbol{y}_{t;1} - \boldsymbol{\mu}_1 \in \mathbb{R}^m\}_{t=1}^{\infty}$ and $\boldsymbol{Y}_2 \doteq \{\boldsymbol{y}_{t;2} - \boldsymbol{\mu}_2 \in \mathbb{R}^m\}_{t=1}^{\infty}$. Although these are infinite sequences, one can approximate the KL divergence using sequences of length $\tau - 1$, $\boldsymbol{Y}_1^{1:\tau-1} \in \mathbb{R}^{m(\tau-1)}$ and $\boldsymbol{Y}_2^{1:\tau-1} \in \mathbb{R}^{m(\tau-1)}$, because

$$D(P_{\boldsymbol{Y}_1}\|P_{\boldsymbol{Y}_2}) = \lim_{\tau\to\infty} \frac{1}{\tau-1} D(P_{\boldsymbol{Y}_1^{1:\tau-1}}\|P_{\boldsymbol{Y}_2^{1:\tau-1}}). \tag{10}$$

In the case of LDSs, the output sequences are distributed as $\boldsymbol{Y}_i^{1:\tau-1} \sim \mathcal{N}(\boldsymbol{\Upsilon_i}, \boldsymbol{\Phi_i})$, $i = 1, 2$. Omitting the subscript, $\boldsymbol{\Upsilon} = \underline{\boldsymbol{C}}[(\boldsymbol{A}\boldsymbol{x}_0)^\top, \ldots, (\boldsymbol{A}^{\tau-1}\boldsymbol{x}_0)^\top]^\top$ is the mean of $\boldsymbol{Y}^{1:\tau-1}$ and $\boldsymbol{\Phi} = \underline{\boldsymbol{C}}\boldsymbol{\Sigma}\underline{\boldsymbol{C}}^\top + \underline{\boldsymbol{R}}$ is the covariance. Here $\underline{\boldsymbol{C}}$ and $\underline{\boldsymbol{R}}$ are block diagonal matrices formed from $\boldsymbol{C}$ and $\boldsymbol{R}$, respectively, and $\boldsymbol{\Sigma}$ is a matrix with block entries $\boldsymbol{\Sigma}_{(i,j)} = \boldsymbol{A}^{i-j}\sum_{k=0}^{\min(i,j)-1} \boldsymbol{A}^k\boldsymbol{Q}(\boldsymbol{A}^k)^\top$ for $j \le i \le \tau - 1$, and $\boldsymbol{\Sigma}_{(j,i)} = \boldsymbol{\Sigma}_{(i,j)}^\top$ for $i < j$. Thus, the KL divergence between the two sequences becomes

$$D(P_{\boldsymbol{Y}_1^{1:\tau-1}}\|P_{\boldsymbol{Y}_2^{1:\tau-1}}) = \frac{1}{2}[\log\frac{|\boldsymbol{\Phi_2}|}{|\boldsymbol{\Phi_1}|} + \mathrm{trace}(\boldsymbol{\Phi_2}^{-1}\boldsymbol{\Phi_1}) + \|\boldsymbol{\Upsilon_1} - \boldsymbol{\Upsilon_2}\|_{\boldsymbol{\Phi_2}}^2 - m(\tau-1)]. \tag{11}$$

By symmetrizing the KL divergence, we obtain the KL distance d_{KL}

$$d_{KL}(\boldsymbol{M}_1, \boldsymbol{M}_2) = \frac{1}{2}[D(P_{\boldsymbol{Y}_1^{1:\tau-1}}\|P_{\boldsymbol{Y}_2^{1:\tau-1}}) + D(P_{\boldsymbol{Y}_2^{1:\tau-1}}\|P_{\boldsymbol{Y}_1^{1:\tau-1}})]. \tag{12}$$

The KL kernel k_{KL} is then defined by using radial basis functions (RBFs) as

$$k_{KL}(\boldsymbol{M}_1, \boldsymbol{M}_2) = e^{-\gamma(d_{KL}(\boldsymbol{M}_1,\boldsymbol{M}_2))^2}, \tag{13}$$

where γ is a free parameter.

In [22], Vishwanathan et al. introduced the family of Binet-Cauchy kernels for the analysis of dynamical systems. One of such kernels is called the *trace kernel*. In the case of two ARMA models $\boldsymbol{M}_1 = (\boldsymbol{x}_{0;1}, \boldsymbol{A}_1, \boldsymbol{B}_1, \boldsymbol{C}_1)$ and $\boldsymbol{M}_2 = (\boldsymbol{x}_{0;2}, \boldsymbol{A}_2, \boldsymbol{B}_2, \boldsymbol{C}_2)$ driven with the same noise realization $\boldsymbol{v}_t$, the trace kernel is

$$k_T(\boldsymbol{M}_1, \boldsymbol{M}_2) = \mathrm{trace}(\Sigma_{\boldsymbol{x}_0}\boldsymbol{P}) + \frac{\lambda}{1-\lambda}\mathrm{trace}(\boldsymbol{B}_1^\top\boldsymbol{P}\boldsymbol{B}_2\Sigma_{\boldsymbol{v}_t}). \tag{14}$$

Here $\Sigma_{\boldsymbol{x}_0} = \mathbb{E}\left[\boldsymbol{x}_{0;1}\boldsymbol{x}_{0;2}^\top\right]$ is the covariance matrix of the initial conditions (assuming that $\mathbb{E}[\boldsymbol{x}_{0;1}] = \mathbb{E}[\boldsymbol{x}_{0;2}] = \boldsymbol{0}$), $\Sigma_{\boldsymbol{v}_t} = \boldsymbol{I}_p$ with $p = n$, λ is a parameter such that $0 < \lambda < 1$, and $\boldsymbol{P} \in \mathbb{R}^{n\times n}$ is the solution of the Sylvester's equation

$$\boldsymbol{P} = \lambda\boldsymbol{A}_1^\top\boldsymbol{P}\boldsymbol{A}_2 + \boldsymbol{C}_1^\top\boldsymbol{C}_2. \tag{15}$$

5 Experimental evaluation

Experiments are conducted using the audio-visual database MVGL-AVD [12]. The video stream has color video frames of size 720×576 pixels at a rate of 15 fps, each containing the frontal view of a speaker's head. In this work, we randomly select 12 out of 50 subjects from the database and form 3 groups (G_4, G_8, G_{12}) composed of 4, 8 and 12 subjects, respectively. Fig. 1(b) illustrates these groups in such a way that the subjects on the first row form G_4, the subjects on the first two rows form G_8, and the subjects on all rows form G_{12}. There are 10 sequences per subject, making G_4, G_8, G_{12} have 40, 80, and 120 sequences respectively.

From the frontal face images of a sequence, we eliminate the head motion and extract the lip region of size 128×80. By the nature of the speaking, act the length of the lip sequences, even for the same uttering, may be different. Hence for accurate identification, we perform linear interpolation to the $x-$, $y-$, and distance trajectories, i.e. $\{\mathcal{L}_x, \mathcal{L}_y, \mathcal{D}\}$, and extend the length of the sequences to a fixed number by oversampling the fitted curves. Having obtained the equal-length trajectories as measurements, the parameters of the corresponding ARMA models, of order $n = 3$, are identified using both the N4SID and the PCA-based identification methods. We use the implementation of N4SID in the MATLAB `system identification toolbox`. We then compute the (dis)similarity between pairs of models using the distances and kernels for LDSs, and perform classification using both nearest neighbors and support vector machines. To ensure that the identified systems are stable, we scale the eigenvalues of $\boldsymbol{A}$ to lie inside the unit circle. Also, to ensure that the KL divergence converges, in the computation of the KL metrics we regularize $\boldsymbol{Q}$ so that $\boldsymbol{Q}' = \boldsymbol{Q} + \boldsymbol{I}_n$ to prevent singularities. We set the λ parameter in the trace kernel k_T to 0.5 in all our experiments.

5.1 Classification of lip models

We evaluate the classification performance on two different scenarios, namely the *Name* scenario and the *Digit* scenario. In the name scenario each subject utters ten repetitions of her/his name as the secret phrase, whereas in the digit scenario each subject utters ten repetitions of a 6-digit number as the password. It is worth noting that the latter is more challenging than the former as it reveals discrimination only among speakers.

To evaluate the performance of the different lip features, identification methods, distances and kernels for LDSs, and classification methods, we performed classification of the data sets G_4, G_8, and G_{12} using leave-one-out classification. Table 1 shows the classification errors of all groups in the name and digit scenarios for both the landmark trajectories $\mathcal{L}$ and the distance trajectories $\mathcal{D}$, using both the N4SID and the PCA-based identification methods, and using both 1-NN classification with 5 different distances and SVM classification with 3 different kernels. In SVM classification, a one-against-all scheme is used to learn the multi-class problem. The slack penalty parameter C and the kernel parameter γ are selected using 2-fold cross-validation over the training set. The multi-class SVM training-testing is performed using the `LibSVM` software [6].

Table 1. Classification error (%) using leave-one-out classification.

(a) name scenario

	1-NN					SVM		
N4SID-$\mathcal{D}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	40	50	40	50	55	95	75	47
G_8	79	75	69	79	75	73	92	70
G_{12}	88	83	80	90	78	84	92	81
N4SID-$\mathcal{L}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	33	25	30	58	8	20	5	15
G_8	58	44	45	64	59	46	17	48
G_{12}	53	42	43	66	44	50	20	64
PCA-$\mathcal{D}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	50	45	38	43	30	95	32	52
G_8	69	70	65	61	58	76	52	63
G_{12}	74	70	69	68	71	75	55	66
PCA-$\mathcal{L}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	38	13	10	3	0	17	0	0
G_8	63	41	25	23	33	45	15	33
G_{12}	63	49	34	36	39	47	17	40

(b) digit scenario

	1-NN					SVM		
N4SID-$\mathcal{D}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	70	63	50	55	70	57	50	75
G_8	79	80	80	90	86	85	82	87
G_{12}	83	83	87	93	89	100	95	90
N4SID-$\mathcal{L}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	50	43	28	15	23	32	12	42
G_8	70	55	49	54	50	76	48	68
G_{12}	74	64	55	59	50	65	25	67
PCA-$\mathcal{D}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	55	50	48	33	75	77	40	100
G_8	63	65	64	60	83	65	57	93
G_{12}	78	75	71	73	83	93	78	81
PCA-$\mathcal{L}$	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
G_4	43	28	15	13	23	40	10	10
G_8	55	46	30	34	51	41	13	36
G_{12}	63	48	33	43	64	64	20	43

In order to analyze the effect of the size of the training set on the classification error, we perform another classification experiment for all groups in the name scenario using the landmark trajectories $\mathcal{L}$ as features and the PCA-based identification method. We use 70%, 50%, and 30% of the data set for training the SVMs. The results of these experiments are summarized in Table 2(a).

To evaluate the performance of the different ARMA metrics and kernels as a function of the order of the identified dynamical systems, we perform a third classification experiment for group G_8 in the name scenario using the landmark trajectories $\mathcal{L}$ as features and the PCA-based identification method. Table 2(b) shows the classification errors for this experiment.

By looking at the results from the different experiments, we can draw the following conclusions.

1. *Feature selection*: Tables 1(a) and 1(b) clearly show that the landmark features $\mathcal{L}$ outperform the distance features $\mathcal{D}$ in terms of classification error. The reason is that, by construction, the lip landmarks contain more local information than the lip distances. Furthermore, this information is coded in a feature of much larger dimension. Hence, the more local information a lip feature vector has, the more powerful it is in terms of discrimination. In fact, the classification errors obtained using the distance features $\mathcal{D}$ are so high that they cannot be used for classification. From now on we will only use the landmark features to draw conclusions about the classification performance.
2. *Identification method*: As shown in Tables 1(a) and 1(b), the PCA-based method outperforms N4SID in term of classification error for most distances and kernels. At a first sight, this may come as a surprise, because N4SID is asymptotically consistent and under some conditions also asymptotically

Table 2. Classification errors (%) as a function of the size of the training set and the system orders for the name scenario, using $\mathcal{L}$ features and PCA-based identification.

(a) Error versus size of the training set for a system order of $n = 3$.

		SVM		
Group	Training (%)	k_M	k_{KL}	k_T
	70	8	0	0
G_4	50	25	5	5
	30	32	3	3
	70	41	25	41
G_8	50	47	40	47
	30	51	28	48
	70	38	27	52
G_{12}	50	40	26	51
	30	47	30	57

(b) Error versus system order n using leave-one-out classification.

	1-NN					SVM		
n	d_F	d_M	d_f	d_{KL}	k_T	k_M	k_{KL}	k_T
1	35	31	31	39	40	50	36	32
2	56	35	21	28	31	45	15	36
3	63	41	25	23	33	45	15	33
4	68	43	18	29	35	61	10	21
5	66	40	18	25	39	45	11	20
6	61	41	28	38	41	38	11	21
7	66	36	29	48	36	33	12	23
8	63	30	19	48	32	20	8	17
9	70	36	26	59	32	28	15	15
10	73	46	24	74	34	46	13	22

efficient [2], while the PCA-based method is not. Because of this, the PCA-based method is often called the sub-optimal identification method [10]. Why does the PCA-method perform better than N4SID then? First of all, modeling the feature trajectories as the output of a LDS is merely a modeling assumption, and so the data need not comply with the model. Thus, by disregarding the relationship between $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$, the PCA-based approach allows for some level of non-linearity on the evolution of $\boldsymbol{x}_t$, even if it fits a linear model thereafter. Second, notice that our evaluation metric is the classification error, not the identification error. Therefore, a method that is optimal for identification, need not be optimal for classification.

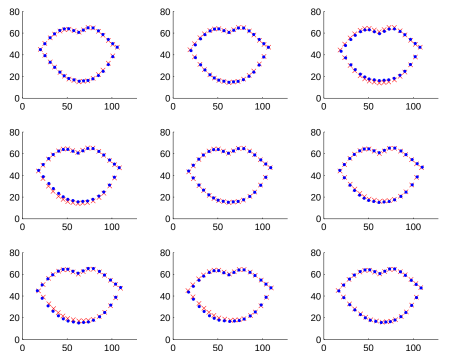
3. *Distances*: Among the distances based on subspace angles, the Frobenius distance d_f outperforms both the Finsler distance d_F and the Martin distance d_M when using 1-NN classification with the landmarks $\mathcal{L}$ as features. In the name scenario (Table 1(a)) d_f achieves an error rate of 10% in G_4, 25% in G_8, and 34% in G_{12}. In the digit scenario (Table 1(b)) d_f achieves an error rate of 15% in G_4, 30% in G_8, and 33% in G_{12}. Among the remaining distances, notice that the performance of the KL divergence d_{KL} and trace kernel k_T based distances is comparable to that of the Frobenius distance d_f. In fact, the relative performance of these distances depends on the number of classes: d_f often performs better for a large number of classes, whereas d_{KL} and k_T often perform better for a small number of classes. Regarding the performance as a function of the system order, for 1-NN classification the best distance is the Frobenius distance d_f, as shown in Table 2(b).
4. *Kernels*: When we look at the classification results in Tables 1(a) and 1(b), we conclude that the KL kernel k_{KL} outperforms the Martin kernel k_M and the trace kernel k_T with error rate of 0% in G_4, 15% in G_8, and 17% in G_{12} when using 1-NN classification with the landmarks $\mathcal{L}$ as features in the name scenario, and with an error rate of 10% in G_4, 13% in G_8, and 20% in G_{12} in the digit scenario. In addition, as shown in Table 2(a), k_{KL} is also the best

kernel in terms of its robustness with respect to the size of the training set. Furthermore, the KL kernel also shows the best performance for different system orders, as shown in Table 2(b). In fact, the KL kernel achieves the best classification result for G_8 with an error of 8% when the order is $n = 8$.

5. *Classification method:* Tables 1(a) and 1(b) show that SVMs yield lower errors than 1-NN with a decrease in the error rate from 34% to 17% for G_{12} in the name scenario, and from 33% to 20% for G_{12} in the digit scenario. In addition, as the system order changes in Table 2(b), the k_{KL} kernel used with SVMs outperform all other metrics with 1-NN classification.

5.2 Synthesis of Lip Dynamics

Figures 3(a) and 3(b) illustrate consecutive frames of a lip sequence with the original features and the synthesized features. It is observed that the synthesized landmarks are close to the original ones and follow them according to the learned dynamical models. In Fig. 3(b), the synthesized distances are placed in such a way that the middle point of each distance is the mean point of 2 vertical landmarks, and still the discrepancy between the landmarks is small. Thus the lip articulation is accurately animated with the landmark $\mathcal{L}_{xy}$ and distance features $\mathcal{D}$. We also show the time evolution of the average error between the true and the synthesized lip articulation for 120 synthesized videos in the name scenario. Fig. 4(a) shows error plots for the 4 most critical landmarks, i.e. $\{p_1, p_3, p_5, p_6\}$, and Fig. 4(b) shows error plots for the 4 distance features with maximum discrepancy, i.e. $\{l_7, l_8, l_9, l_{10}\}$. It can be seen that these errors are within an acceptable range of $[0, 5]$ pixels, with a maximum of 3.75 pixels for the landmark features and 4.4 pixels for the distance features. The average lip size is 50×80 pixels.


(a) The original (red) and synthesized (blue) lip landmarks for the first 9 frames of a sequence

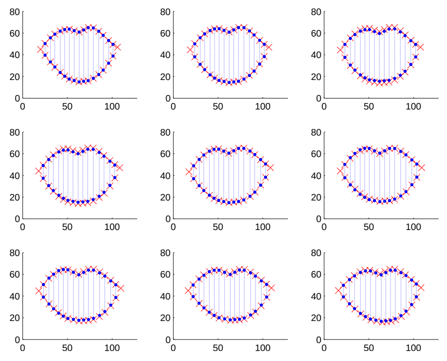
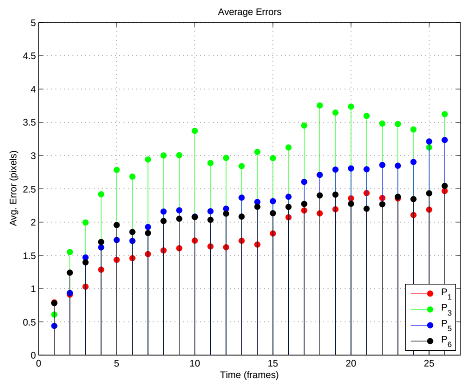
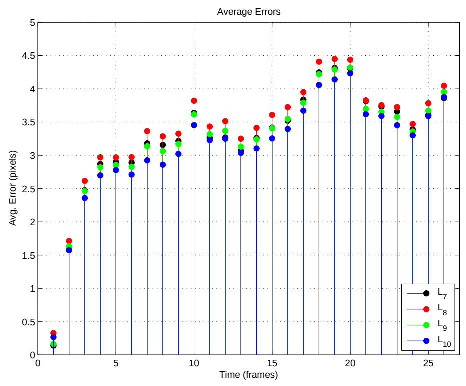

(b) The original (red) and lip landmarks created from synthesized distances (blue) for the first 9 frames of a sequence

Fig. 3. Synthesis results using landmark and distance based features.



(a) Landmark features $\mathcal{L}_{xy}$



(b) Distance features $\mathcal{D}$

Fig. 4. Time evolution of the error between true and synthesized lip articulation.

6 Conclusion

An analysis/synthesis methodology that models lip articulation in the space of dynamical systems has been proposed. In this setting, using intelligent lip features not only gives good classification and accurate animations, but also helps us model highly complex dynamics that are nonlinear and non-stationary in the intensity space. Among the lip features that are fed to the system, landmark trajectories are shown to possess adequate local information, which gives discriminative power to the classifier. Furthermore, it has been demonstrated that the PCA-based approach, which is mostly preferred for dynamic texture recognition, outperforms the N4SID in terms of classification performance. Moreover, the Frobenius distance provides the best classification results among other subspace angles based distances for 1-NN, whereas the Kullback-Leibler kernel is shown to be the best kernel for SVM classification. Lastly, the corresponding synthesis scheme modified for natural lip movements can create accurate and realistic lip sequences.

In the future, not only can kernels derived from the Frobenius distance be employed for classification, but also hybrid systems can be used for system identification. Further research is also needed for determining the best system order. The proposed framework could be integrated with speech modality to obtain more accurate lip dynamics, which makes it possible to use in audio-visual speech synthesis, facial animations, lip reading education of hard of hearing people, and biometric security systems.

Acknowledgements

The authors thank Antoni Chan for providing the KL divergence code, and Dr. Mihály Petreczky and Avinash Ravichandran for useful discussions. This work has been funded by Johns Hopkins WSE startup funds and by grants ONR N00014-05-10836, NSF CAREER IIS-0447739, and NSF EHS-0509101.

References

1. G. Aggarwal, A. Roy-Chowdhury, and R. Chellappa. A system identification approach for video-based face recognition. In *IEEE International Conference on Pattern Recognition*, pages 23–26, 2004.
2. D. Bauer. Asymptotic properties of subspace estimators. *Automatica*, 41(3):359–376, 2005.
3. A. Bissacco, A. Chiuso, Y. Ma, and S. Soatto. Recognition of human gaits. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 52–58, 2001.
4. H. E. Çetingül, E. Erzin, Y. Yemez, and A. M. Tekalp. Discriminative analysis of lip motion features for speaker identification and speech-reading. *IEEE Trans. on Image Processing*, 15(10):2879–2891, 2006.
5. A. B. Chan and N. Vasconcelos. Probabilistic kernels for the classification of auto-regressive visual processes. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 846–851, 2005.
6. C.-C. Chang and C.-J. Lin. *LIBSVM : a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
7. T. Chen. Audiovisual speech processing. *IEEE Signal Processing Magazine*, pages 9–21, 2001.
8. K. D. Cock and B. D. Moor. Subspace angles and distances between ARMA models. *System and Control Letters*, 46(4):265–270, 2002.
9. E. Cosatto, J. Ostermann, H. P. Graf, and J. Schroeter. Lifelike talking faces for interactive services. *Proceedings of the IEEE*, 91(9):1406–1429, 2003.
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 and S. Soatto. Editable dynamic textures. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 137–142, 2003.
12. E. Erzin, Y. Yemez, and A. M. Tekalp. Audio-visual database (MVGL-AVD), Multimedia, Vision and Graphics Laboratoty, Koç University. http://mvgl.ku.edu.tr/.
13. N. Eveno, A. Caplier, and P.-Y. Coulon. Accurate and quasi-automatic lip tracking. *IEEE Trans. on Circuits and Systems For Video Technology*, 14(5):706–715, 2004.
14. T. Katayama. *Subspace Methods for System Identification*. Springer-Verlag, 2005.
15. S. A. King and R. E. Parent. Creating speech-synchronized animation. *IEEE Trans. on Visualization and Computer Graphics*, 11(3):341–352, 2005.
16. R. J. Martin. A metric for ARMA processes. *IEEE Trans. on Signal Processing*, 48(4):1164–1170, 2000.
17. J.-M. Odobez, P. Bouthemy, and F. Spindler. Motion2D: a software to estimate 2D parametric motion models. http://www.irisa.fr/vista/Motion2D/.
18. P. V. Overschee and B. D. Moor. N4SID : Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica, Special Issue in Statistical Signal Processing and Control*, pages 75–93, 1994.
19. P. Saisan, A. Bissacco, A. Chiuso, and S. Soatto. Modeling and synthesis of facial motion driven by speech. In *ECCV*, pages 456–467, 2004.
20. P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 58–63, 2001.
21. M. Szummer and R. W. Picard. Temporal texture modeling. In *IEEE International Conference On Image Processing*, volume 3, pages 823–826, 1996.
22. S. Vishwanathan, A. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73(1), 2007.
23. L. Yuan, F. Wen, C. Liu, and H.-Y. Shum. Synthesizing dynamic texture with closed-loop linear dynamic system. In *European Conference on Computer Vision*, pages 603–616, 2004.