Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 2423–2428

Identification of dynamic textures using Dynamic Mode Decomposition

D. Previtali *, **N. Valceschini** *, **M. Mazzoleni** *, **F. Previdi** *

* *Department of Management, Information and Production engineering University of Bergamo, via Galvani 2, 24044 Dalmine (BG), Italy (e-mail: davide.previtali@unibg.it).*

Abstract: Dynamic Textures (DTs) are image sequences of moving scenes that present stationary properties in time. In this paper, we apply Dynamic Mode Decomposition (DMD) and Dynamic Mode Decomposition with Control (DMDc) to identify a parametric model of dynamic textures. The identification results are compared with a benchmark method from the dynamic texture literature, both from a mathematical and from a computational complexity point of view. Extensive simulations are carried out to assess the performance of the proposed algorithms with regards to synthesis and denoising purposes, with different types of dynamic textures. Results show that DMD and DMDc present lower error, lower residual noise and lower variance compared to the benchmark approach.

Keywords: Dynamic textures, System Identification, Texture Synthesis, Dynamic Mode Decomposition

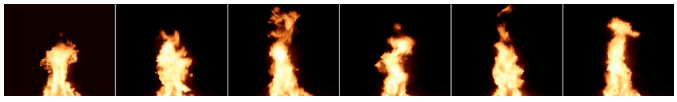
1. INTRODUCTION

Dynamic Textures (DTs) are sequences of images that exhibit some form of temporal stationarity as well as a statistical regularity in the spatial domain, see Doretto et al. (2003). Examples of DT are depicted in Fig. 1, where several frames of the sequence (from left to right) are reported. The study of the DTs is well motivated by a wide number of applications, see Zhao et al. (2019); Saisan et al. (2001); Doretto and Soatto (2003):

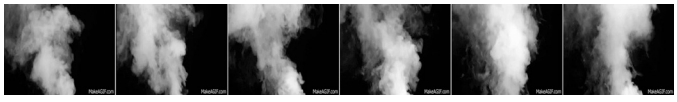
- **Synthesis:** generation of new sequences similar to the original one.
- **Denoising:** reduce the amount of noise present in the original video.
- **Classification/recognition:** assess if a dynamic texture belongs to a particular category.
- **Compression:** reduce the storage requirement needed to represent a dynamic texture.
- **Editing:** modify the temporal behavior of the dynamic texture.

There are different approaches that can be used to solve these problems. Two macro-categories of algorithms exist: (i) *physics-based*, which derive the model of the scene from first principles (Barzel and Barr (2013)); (ii) *image-based*, where texture movies are obtained using images without building a physical model of the process that generated the scene (Wei and Levoy (2000)). Image-based methods can be further subdivided in: (a) procedural techniques, which don't use a model and perform synthesis by concatenation or repetition of the image data, and (b) model-based techniques that indeed use a model, although not a physical one.

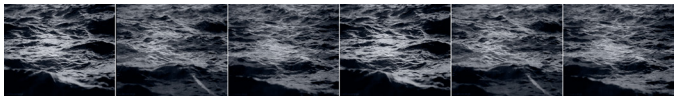
Compared to physical and procedural approaches, model-based techniques are the most cost-effective solutions, in terms of complexity and flexibility for the aforementioned purposes, see Doretto et al. (2003). Several methods have been devised relying on the model-based rationale, see Bar-Joseph et al. (2001). Pivotal to this modeling approach is the concept of DT *representation*. In this work, we follow the interpretation of Doretto et al. (2003), where individual images are seen as *realizations of the output of a (linear) dynamical system* driven by an independent and identically distributed (i.i.d.) process. The problem is now to estimate a parametric model of the dynamical system that describes the DT. Throughout this paper, we will refer to the system identification algorithm proposed in Doretto et al. (2003) as the "Classic approach".



(a) The `Fire` dynamic texture



(b) The `Smoke` dynamic texture



(c) The `Waves` dynamic texture

(d) The `Waves and ship` dynamic texture

Fig. 1. Frames extracted from different dynamic textures.

10.1016/j.ifacol.2020.12.045

An immediate obstacle in developing a model of images for the DT is given by the *high-dimensionality of the system*. In fact, in its most intuitive formulation, each pixel of the images in the DT can be considered as a state of the system that evolves in time. In order to cope with this, the authors in Doretto et al. (2003) represented the system in a state-space formulation where the full image pixels are projected onto a lower number of states.

Dynamic Mode Decomposition (DMD), Schmid (2010), and DMD with control (DMDc), Proctor et al. (2016) recently emerged as powerful data-driven techniques to represent and model high-dimensional dynamical systems. The aim of DMD-like methods is to estimate the most important modes and eigenvalues of a dynamical system using matrix decomposition methods such as Singular Value Decomposition (SVD), starting from its measurements. This linear approximation has been found useful also for representing *nonlinear* systems, see Rowley et al. (2009). DMD has been already applied for video processing applications, for instance for the separation of the background from the foreground (in Grosek and Kutz (2014)) and in shot detection (in Bi et al. (2018)). To the best of our knowledge, these algorithms have *never been employed for the identification of dynamic textures*.

The contributions of this work are: (i) use DMD and DMDc algorithms for the parametric identification of dynamic textures; (ii) reformulate the DMD-like estimation problems in a way that is akin to the Classic approach; (iii) compare the DMD, DMDc and the Classic approach algorithms, both from a mathematical and a computational efficiency standpoint; (iv) propose and perform a set of experiments to verify the goodness of the synthesis, as well as the denoising performance, of dynamic textures.

2. PROBLEM STATEMENT

Let $\{I(t)\}_{t=1,\dots,\tau}$, $I(t) \in \mathbb{R}^m$, be a sequence of τ images, which represents the analyzed video. Suppose that it is possible to measure a *noisy version* of the image at each time instant t,

$$y(t) = I(t) + w(t) \qquad w(t) \overset{\text{i.i.d.}}{\sim} p_w(\cdot), \tag{1}$$

with $y(t) \in \mathbb{R}^m$ and $w(t) \in \mathbb{R}^m$ is a realization drawn from a known distribution $p_w(\cdot)$ [1]. We say that the sequence $\{I(t)\}$ is a *(linear) dynamic texture* (Doretto et al. (2003)) if there exists a stationary distribution $q(\cdot)$ and three matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times n_e}$, $C \in \mathbb{R}^{m\times n}$ such that each image $I(t)$ can be thought as a linear combination of states $x(t)$ of the following dynamical system:

$$\begin{cases} x(t+1) = Ax(t) + Be(t), & e(t) \overset{\text{i.i.d.}}{\sim} q(\cdot), \\ y(t) = Cx(t) + w(t), & w(t) \overset{\text{i.i.d.}}{\sim} p_w(\cdot), \end{cases} \tag{2}$$

with $x(t) \in \mathbb{R}^n$, $x(0) = x_0$, $e(t) \in \mathbb{R}^{n_e}$ and $q(\cdot)$ unknown. Therefore, in order to identify a dynamic texture, we need to estimate the matrices A, B, C and the distribution of the input $q(\cdot)$ in the model. To simplify the problem and avoid estimating an unknown distribution, we assume that the input sequence can be modeled as a *second-order stationary process* with arbitrary covariance (Doretto et al. (2003)). Under this hypothesis, we can model $\{y(t)\}$ as the output of a linear dynamical system driven by white zero-mean Gaussian noise (Ljung (1986)):

$$\begin{cases} x(t+1) = Ax(t) + e(t), & e(t) \sim \mathcal{N}(0, Q), \qquad x(0) = x_0, \\ y(t) = Cx(t) + w(t), & w(t) \sim \mathcal{N}(0, R), \end{cases} \tag{3}$$

with $Q \in \mathbb{R}^{n\times n}$ and $R \in \mathbb{R}^{m\times m}$ symmetric positive-definite matrices. In this form, the system identification problem amounts to: *estimate the model parameters A, C, Q, R from the measurements $y(1), ..., y(\tau)$.*

Remark. Under the given hypothesis, $Be(t)$ in the model (2), with $e(t) \sim \mathcal{N}(0, I_{n_e})$ where I_{n_e} is the $n_e \times n_e$ identity matrix, is equivalent to the term $e(t)$ in the model (3) if Q is such that $Q = BB^T$.

Remark. Since we are interested in data dimensionality reduction, we make the following assumptions about the model (3):

$$m \gg n, \qquad \text{rank}(C) = n. \tag{4}$$

3. IDENTIFICATION OF DYNAMIC TEXTURES

3.1 Identification with the Classic approach

In the Classic approach of Doretto et al. (2003), the authors obtained a (although suboptimal) *closed form solution* to the identification problem posed in Section 2.

Let $Y_1^\tau = [y(1)\ y(2)\ ...\ y(\tau)] \in \mathbb{R}^{m\times\tau}$ be the input matrix that represents the analyzed video. Each column $y(t)$ contains every pixel of the frame at the time t. The same matrix notation can be used for the state of the dynamical system $x(t)$ as well as $w(t)$ and $e(t)$. Let n be the system order (it can be estimated by analyzing the singular values of Y_1^τ). Then, let $Y_1^\tau = U_Y \Sigma_Y V_Y^\top$, where $U_Y \in \mathbb{R}^{m\times n}$, $\Sigma_Y \in \mathbb{R}^{n\times n}$ and $V_Y \in \mathbb{R}^{\tau\times n}$, be the low-rank approximation obtained from the SVD of Y_1^τ.

We can write the output equation of the system (3) as

$$Y_1^\tau = CX_1^\tau + W_1^\tau, \tag{5}$$

with $X_1^\tau \in \mathbb{R}^{n\times\tau}$ and $W_1^\tau \in \mathbb{R}^{m\times\tau}$. In order to identify a unique model, the authors of Doretto et al. (2003) chose the canonical model that makes the columns of C orthonormal:

$$C^\top C = I_n. \tag{6}$$

Under this assumption, the matrices C and X_1^τ can be estimated by minimizing the following cost function:

$$J(C, X_1^\tau) = ||W_1^\tau||_F = \sqrt{\text{tr}\left((Y_1^\tau - CX_1^\tau)(Y_1^\tau - CX_1^\tau)^\top\right)}$$

obtaining:

$$\hat{C}(\tau) = U_Y, \tag{7a}$$

$$\hat{X}_1^\tau(\tau) = \Sigma_Y V_Y^\top. \tag{7b}$$

Similarly, an estimate of the matrix A can be achieved by solving the following linear problem:

$$\hat{A}(\tau) = \underset{A}{\text{argmin}} ||X_2^\tau - AX_1^{\tau-1}||_F = \hat{X}_2^\tau(\tau)\left[\hat{X}_1^{\tau-1}(\tau)\right]^\dagger, \tag{8}$$

[1] As stated by the authors in Doretto et al. (2003), $p_w(\cdot)$ can be inferred from the physics of the imaging device. For example, for CCD sensors, a good approximation is a Poisson distribution with intensity related to the average photon count.

where the symbol † denotes the pseudoinverse of the matrix, $X_2^\tau = [x(2)\ x(3)\ ...\ x(\tau)] \in \mathbb{R}^{n\times(\tau-1)}$ and $X_1^{\tau-1} = [x(1)\ x(2)\ ...\ x(\tau-1)] \in \mathbb{R}^{n\times(\tau-1)}$ are partitions of the state matrix X_1^τ. Given the state matrix X_1^τ and A, we can estimate the input matrix $\hat{E}_1^{\tau-1}(\tau) \in \mathbb{R}^{n\times(\tau-1)}$ as

$$\hat{E}_1^{\tau-1}(\tau) = \hat{X}_2^\tau(\tau) - \hat{A}(\tau)\hat{X}_1^{\tau-1}(\tau), \tag{9}$$

and estimate its covariance matrix Q using the unbiased estimator:

$$\hat{Q}(\tau) = \frac{1}{\tau-2}\hat{E}_1^{\tau-1}(\tau)\cdot\left[\hat{E}_1^{\tau-1}(\tau)\right]^\top. \tag{10}$$

The matrix B can be estimated by recognizing that $Q = BB^\top$. In order to do so, it is possible to get the input dimension as

$$n_e = \operatorname{rank}\left(\hat{Q}(\tau)\right), \tag{11}$$

and then perform the low-rank approximation of the matrix $\hat{Q}(\tau) = U_Q\Sigma_Q V_Q^\top$ by SVD, with $U_Q \in \mathbb{R}^{n\times n_e}$, $\Sigma_Q \in \mathbb{R}^{n_e\times n_e}$ and $V_Q \in \mathbb{R}^{n\times n_e}$. Since Q is a positive semi-definite symmetric matrix and Σ_Q is a diagonal matrix, it holds that:

$$\hat{Q}(\tau) = U_Q\Sigma_Q U_Q^\top = U_Q\Sigma_Q^{\frac{1}{2}}\Sigma_Q^{\frac{1}{2}}U_Q^\top = U_Q\Sigma_Q^{\frac{1}{2}}\left(\Sigma_Q^{\frac{1}{2}}\right)^\top U_Q^\top,$$

and therefore B can be estimated as:

$$\hat{B}(\tau) = U_Q\Sigma_Q^{\frac{1}{2}}. \tag{12}$$

The estimated dynamical system can be represented as:

$$\begin{cases} x(t+1) = \hat{A}(\tau)x(t) + \hat{B}(\tau)e(t), & e(t)\sim\mathcal{N}(0, I_{n_e}),\\ y(t) = \hat{C}(\tau)x(t). \end{cases} \tag{13}$$

3.2 Identification with DMD and DMDc

Dynamic Mode Decomposition The DMD approach assumes that the data can be described by the linear mapping:

$$y(t+1) \approx Ay(t), \tag{14}$$

where $A \in \mathbb{R}^{m\times m}$. Equation (14) can be seen in matrix form considering two partitions of the input matrix $Y_1^{\tau-1}$, i.e. $Y_1^{\tau-1} = [y(1)\ y(2)\ ...\ y(\tau-1)] \in \mathbb{R}^{m\times(\tau-1)}$ and $Y_2^\tau = [y(2)\ y(3)\ ...\ y(\tau)] \in \mathbb{R}^{m\times(\tau-1)}$, such that:

$$Y_2^\tau \approx AY_1^{\tau-1}. \tag{15}$$

In order to find the best-fit solution of the operator A, we minimize the following cost function:

$$J(A) = ||Y_2^\tau - AY_1^{\tau-1}||_F, \tag{16}$$

which results in

$$A = Y_2^\tau\left(Y_1^{\tau-1}\right)^\dagger. \tag{17}$$

In practice, this solution *can not be used* since the pseudoinverse of $Y_1^{\tau-1}$ is not computable, because it involves the inverse of a high dimensional $m\times m$ matrix. Therefore, it is necessary to find a *computationally efficient way* to compute the pseudoinverse of $Y_1^{\tau-1}$. In order to do so, the system order $n \ll m$ is obtained similarly to Section 3.1, but the considered matrix is now $Y_1^{\tau-1}$ instead of Y_1^τ. Then, the low-rank approximation of $Y_1^{\tau-1}$ is computed with the SVD, obtaining $Y_1^{\tau-1} = U_Y\Sigma_Y V_Y^\top$, where $U_Y \in \mathbb{R}^{m\times n}$, $\Sigma_Y \in \mathbb{R}^{n\times n}$ and $V_Y \in \mathbb{R}^{(\tau-1)\times n}$. By substituting this expression in

$$\left(Y_1^{\tau-1}\right)^\dagger = \left(Y_1^{\tau-1}\right)^\top\left[Y_1^{\tau-1}\left(Y_1^{\tau-1}\right)^\top\right]^{-1},$$

we obtain a computationally efficient way to compute the pseudoinverse of a matrix:

$$\left(Y_1^{\tau-1}\right)^\dagger = V_Y\Sigma_Y^{-1}U_Y^\top. \tag{18}$$

By substituting (18) in (17), we have that:

$$\tilde{A} = Y_2^\tau V_Y\Sigma_Y^{-1}U_Y^\top, \tag{19}$$

where $\tilde{A} \in \mathbb{R}^{m\times m}$ is the matrix approximate of A but, even though it is actually computable, *it is still a high dimensional matrix*. To obtain a tractable matrix, notice that it is possible to project $y(t)$ onto a linear subspace of dimension n. In order to do so, we can use the transpose of the previously computed left singular vectors matrix U_Y in the following fashion:

$$x(t) = U_Y^\top y(t), \tag{20a}$$

$$\hat{A}(\tau) = U_Y^\top\tilde{A}\left(U_Y^\top\right)^\dagger = U_Y^\top Y_2^\tau V_Y\Sigma_Y^{-1}. \tag{20b}$$

Therefore, we have obtained an estimate $\hat{A}(\tau) \in \mathbb{R}^{n\times n}$ of the matrix A in a lower-dimensional subspace, and $x(t)$ *can be seen as the state of the following dynamical system*:

$$\begin{cases} x(t+1) = \hat{A}(\tau)x(t),\\ y(t) = U_Y x(t), \end{cases} \tag{21}$$

where the output $y(t)$ of (21) is simply the projection of the state onto the original m-dimensional space.

Dynamic Mode Decomposition with Control The DMDc method is an extension to the DMD, which takes into consideration an external input $z(t) \in \mathbb{R}^{n_z}$. DMDc assumes that the data can be described by the following linear mapping:

$$y(t+1) \approx Ay(t) + Bz(t), \tag{22}$$

where $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times n_z}$. Equation (22) can be written in matrix form as:

$$Y_2^\tau \approx AY_1^{\tau-1} + BZ_1^{\tau-1}. \tag{23}$$

It is possible to group together the parameters and the data matrices as:

$$Y_2^\tau \approx [A\ B]\begin{bmatrix} Y_1^{\tau-1}\\ Z_1^{\tau-1}\end{bmatrix} = G\Omega_1^{\tau-1}, \tag{24}$$

where $G \in \mathbb{R}^{m\times(m+n_z)}$ and $\Omega_1^{\tau-1} \in \mathbb{R}^{(m+n_z)\times(\tau-1)}$. Thus, equation (24) is equivalent to equation (15) and we can proceed in the same way as with DMD. However, this time we need to compute the low-rank approximation of the matrix $\Omega_1^{\tau-1}$ instead of $Y_1^{\tau-1}$, $\Omega_1^{\tau-1} = U_\Omega\Sigma_\Omega V_\Omega^\top$ with $U_\Omega \in \mathbb{R}^{(m+n_z)\times r}$, $\Sigma_\Omega \in \mathbb{R}^{r\times r}$ and $V_\Omega \in \mathbb{R}^{(\tau-1)\times r}$. The truncation value r should be larger than n used for DMD because the matrix $\Omega_1^{\tau-1}$ also includes information about the input dynamics. Similarly to DMD, we obtain:

$$\tilde{G} = Y_2^\tau V_\Omega\Sigma_\Omega^{-1}U_\Omega^\top \tag{25}$$

where $\tilde{G} \in \mathbb{R}^{m\times(m+n_z)}$. We can find the matrices $\tilde{A} \in \mathbb{R}^{m\times m}$ and $\tilde{B} \in \mathbb{R}^{m\times n_z}$ by properly partitioning the matrix $U_\Omega^\top$:

$$U_\Omega^\top = \begin{bmatrix} U_{\Omega_1}^\top & U_{\Omega_2}^\top \end{bmatrix}, \quad U_{\Omega_1}^\top \in \mathbb{R}^{r\times m},\, U_{\Omega_2}^\top \in \mathbb{R}^{r\times n_z},$$

$$\tilde{A} = Y_2^\tau V_\Omega \Sigma_\Omega^{-1} U_{\Omega_1}^\top, \tag{26a}$$

$$\tilde{B} = Y_2^\tau V_\Omega \Sigma_\Omega^{-1} U_{\Omega_2}^\top. \tag{26b}$$

As done in the previous section, we need to project $y(t)$ onto a subspace with dimension $n \ll m$. Note that we can not simply use $U_\Omega^\top$ because it includes information about the input $z(t)$ of the system. Therefore, we need to find a $n \times m$ projection matrix which only carries information about $y(t)$. As proposed in Proctor et al. (2016), it is possible use the transpose of the left singular vectors matrix, obtained from the n-rank-approximation given by the SVD of Y_2^τ, i.e. $Y_2^\tau = U_Y \Sigma_Y V_Y^\top$. Thus, we can estimate the state of the system, as well as A and B, as:

$$x(t) = U_Y^\top y(t) \tag{27a}$$

$$\hat{A}(\tau) = U_Y^\top \tilde{A} \left(U_Y^\top\right)^\dagger = U_Y^\top Y_2^\tau V_\Omega \Sigma_\Omega^{-1} U_{\Omega_1}^\top U_Y \tag{27b}$$

$$\hat{B}(\tau) = U_Y^\top \tilde{B} = U_Y^\top Y_2^\tau V_\Omega \Sigma_\Omega^{-1} U_{\Omega_2}^\top \tag{27c}$$

Therefore, we obtain the following linear dynamical system:

$$\begin{cases} x(t+1) = \hat{A}(\tau)x(t) + \hat{B}(\tau)z(t) \\ y(t) = U_Y x(t). \end{cases} \tag{28}$$

Similarly to DMD, the output of the dynamical system is the projection of the state onto an m-dimensional space.

4. COMPARISON OF THE APPROACHES

4.1 Comparison of the mathematical formulations

Even if their derivation differs, the Classic approach and the DMD-like approaches share many similarities in their formulations, see Table 1. In Section 3 we showed how the methods estimate a dynamical system in state-space form, see (13), (21) and (28). Other similarities are:

(1) The output is the projection of the system state $x(t)$ onto the original m-dimensional space. All approaches use the left singular vectors matrix (respectively of Y_1^τ, $Y_1^{\tau-1}$ and Y_2^τ), U_Y, as the projection matrix.
(2) The resulting dynamical systems are linear.

Instead, the main differences are:

(1) The system order is extracted from different matrices.
(2) The estimates of $\hat{A}(\tau)$ and $\hat{B}(\tau)$ differ.
(3) The Classic approach estimates the state matrix $\hat{X}_1^\tau(\tau)$ and the input matrix $\hat{E}_1^{\tau-1}(\tau)$, unlike DMD and DMDc.
(4) DMD identifies an autonomous dynamical system.
(5) DMDc does not estimate any input information, instead it assumes that n_z and $Z_1^{\tau-1}$ are known.

4.2 Comparison of the computational cost

In order to obtain an estimate of the computational cost of the considered algorithms, we have analyzed every step shown in Table 1 and calculated the number of operations needed for each one. From Section 3, we can deduce the following relationships between the problem dimensions:

(1) The number of pixels m is always the highest dimension, since we consider *short videos* such that the number of pixels of each frame is always greater than the number of frames τ; in other words, $m \gg \tau$.
(2) The number of singular values is τ for Y_1^τ and $\tau-1$ for $Y_1^{\tau-1}$ and Y_2^τ. Therefore, the order of the dynamical system is surely $n \leq \tau - 1$ (we do not consider the case $n = \tau$, even if it is an admissible order for the Classic approach, because it is not a valid option for DMD and DMDc since it would exceed the matrix dimensions of the SVD). A similar remark can be said for $r = \max(n + n_z, \tau - 1)$, so $n \leq r < \tau$.

Thus, we have $n \leq r < \tau \ll m$. Under this assumption, the most computationally expensive step is the SVD. In our case, we performed the Thin Singular Value Decomposition: given a matrix $A \in \mathbb{R}^{m\times\tau}$ with $\tau \leq m$, we computed the matrices $U \in \mathbb{R}^{m\times\tau}$, $\Sigma \in \mathbb{R}^{\tau\times\tau}$ and $V \in \mathbb{R}^{\tau\times\tau}$ such that $A = U\Sigma V^\top$. This can be done using the Bidiagonalization and QR algorithm whose computational cost is $\mathrm{O}\left(m\tau^2\right)$ (Golub and Van Loan, 2013, Chapter 5). After that, we chose a truncation value n and we extracted only a portion of U, Σ and V, obtaining three matrices with dimensions $m\times n$, $n\times n$ and $\tau\times n$, respectively. In order to do that, we perform $mn+n^2+\tau n$ operations which is negligible compared to $\mathrm{O}(m\tau^2)$. Given these considerations, the approximate computational costs of each algorithm are reported in Table 2. The analysis shows that *the Classic approach and the DMD one have the same asymptotic complexity*, while *the DMDc method is the most complex*.

5. EXPERIMENTAL RESULTS

5.1 Definition of the experiments

We consider the videos reported in Fig. 1. The video `waves and ship` is from Peteri et al. (2008), while the others were found on the Web. All the original videos and presented results are available[2]. To evaluate the identification performance of the methods, we compare them using two indicators (as function of the system order n): (i) the synthesis error (i.e. how much the generated sequence differs from the original); (ii) the denoising error (i.e. how much noise is still present in the synthesized sequence). Notice that, to compute the performance indicators, it is necessary to first perform the identification of a linear system of chosen order n.

Remark. The DMDc method assumes that the input $z(t)$ is known. To compare the DMDc with the Classic approach, we set $z(t) = e(t)$, such that $n_z = n_e$. Furthermore, we chose the order $r = n + n_z$ (in case $n + n_z > \tau-1$, we set $r = \tau-1$ to prevent exceeding the matrix dimensions) to avoid using a heuristic, which means that we consider all inputs relevant to the decomposition.

Synthesis error as a function of the system order Given the input matrix Y_1^τ, we performed system identification using the three algorithms, for $1 \leq n \leq n_{max}$. The value of n_{max} is chosen by observing the singular values of Y_1^τ, i.e. $\{s_1, s_2, ..., s_\tau\}$. For some videos (i.e the `fire` DT), most of the singular values are zero and therefore n_{max} is equal to the index of the last non-zero singular value; for others, the choice is arbitrary and depends on τ as well as on the

[2] `https://cal.unibg.it/publications/identification-of-dynamic-textures-using-dynamic-mode-decomposition/`

Table 1. Mathematical and algorithmic comparison of the three analyzed approaches.

Classic approach	**Dynamic Mode Decomposition**	**Dynamic Mode Decomposition with Control**
1) Extract n from the singular values of Y_1^τ	1) Extract n from the singular values of $Y_1^{\tau-1}$	1) Extract n from the singular values of Y_2^τ
2) n-rank SVD: $Y_1^\tau = U_Y \Sigma_Y V_Y^\top$	2) n-rank SVD: $Y_1^{\tau-1} = U_Y \Sigma_Y V_Y^\top$	2) n-rank SVD: $Y_2^\tau = U_Y \Sigma_Y V_Y^\top$
3) $\hat{X}_1^\tau(\tau) = \Sigma_Y V_Y^\tau$		
		3) Extract r from the singular values of $\Omega_1^{\tau-1}$
		4) r-rank SVD: $\Omega_1^{\tau-1} = U_\Omega \Sigma_\Omega V_\Omega^\top$
4) $\hat{A}(\tau) = \hat{X}_2^\tau(\tau)\left(\hat{X}_1^{\tau-1}(\tau)\right)^\dagger$	3) $\hat{A}(\tau) = U_Y^\top Y_2^\tau V_Y \Sigma_Y^{-1}$	5) $\hat{A}(\tau) = U_Y^\top Y_2^\tau V_\Omega \Sigma_\Omega^{-1} U_{\Omega_1}^\top U_Y$
5) $\hat{E}_1^{\tau-1}(\tau) = \hat{X}_2^\tau(\tau) - \hat{A}(\tau)\hat{X}_1^{\tau-1}(\tau)$		
6) $\hat{Q}(\tau) = \frac{1}{\tau-2}\hat{E}_1^{\tau-1}(\tau) \cdot \left[\hat{E}_1^{\tau-1}(\tau)\right]^\top$		
7) $n_e = \text{rank}\left(\hat{Q}(\tau)\right)$		
8) n_e-rank SVD: $\hat{Q}(\tau) = U_Q \Sigma_Q V_Q^\top$		
9) $\hat{B}(\tau) = U_Q \Sigma_Q^{\frac{1}{2}}$		6) $\hat{B}(\tau) = U_Y^\top Y_2^\tau V_\Omega \Sigma_\Omega^{-1} U_{\Omega_2}^\top$
10) $\hat{C}(\tau) = U_Y$	4) $\hat{C}(\tau) = U_Y$	7) $\hat{C}(\tau) = U_Y$

Table 2. Approximate computational costs of the compared algorithms.

Algorithm	**Computational cost**
Classic approach	$\mathrm{O}\left(m\tau^2\right)$
DMD	$\mathrm{O}\left(m\tau^2\right)$
DMDc	$\mathrm{O}\left((m+n_z)\,\tau^2\right)$

computational time allowed. Then, for each identified DT, we synthesized a video[3] of length τ using the estimated parameters and equations (13), (21) and (28). We can compute the *normalized synthesis error* $e_S(n)$:

$$e_S(n) = \frac{||Y_1^\tau - \hat{Y}_n||_F}{||Y_1^\tau - \bar{Y}||_F}, \tag{29}$$

where $\bar{Y} = [\mathbb{E}[Y_1^\tau]\ \mathbb{E}[Y_1^\tau]\ \ldots\ \mathbb{E}[Y_1^\tau]] \in \mathbb{R}^{m\times\tau}$ is the average video, $\mathbb{E}[Y_1^\tau] = \frac{1}{\tau}\sum_{j=1}^{\tau} y(j)$, $\mathbb{E}[Y_1^\tau] \in \mathbb{R}^m$, and $\hat{Y}_n \in \mathbb{R}^{m\times\tau}$ is the synthesized video matrix for the order n.

Denoising error as a function of the system order As mentioned in Doretto et al. (2003), a possible use of the DT model is to perform denoising. To carry out this experiment, we created a noisy sequence by adding a Gaussian noise matrix $R \in \mathbb{R}^{m\times\tau}$ to the video matrix Y_1^τ, obtaining a noisy video $Y_{noisy} = Y_1^\tau + R$. Each element R_{ij} of R is such that $R_{ij} \sim \mathcal{N}\left(0, 50^2\right)$. Then, we performed identification and synthesis for $1 \le n \le n_{max}$ from both the original video, Y_1^τ, and the noisy one, Y_{noisy}, obtaining respectively $\hat{Y}_n$ and $\hat{Y}_{noisy_n}$. The reason behind this is that we want to obtain an *estimate of the residual noise in the synthesized video* $\hat{Y}_{noisy_n}$. In order to do so, we need to omit the synthesis error; a possibility is to estimate the remaining noise matrix as $\hat{Y}_n - \hat{Y}_{noisy_n}$ and then compute the residual noise percentage $e_R(n)$ as:

$$e_R(n) = \frac{||\hat{Y}_n - \hat{Y}_{noisy_n}||_F}{||R||_F} \cdot 100. \tag{30}$$

5.2 Results and discussion

After analyzing the singular values of each video matrix, we decided which n_{max} to use for the experiments. For the videos `waves` and `fire` we have, respectively, $n_{max} = 10$ and $n_{max} = 20$, with $s_{n_{max}+1} = s_{n_{max}+2} = \ldots = s_\tau = 0$. For the video `smoke` we choose $n_{max} = \tau - 1 = 39$ since it is relatively small and the experiments are not particularly time consuming (this one has the lowest m of all the videos) while for `waves and ship` (which has $\tau = 250$ and a high m) we pick $n_{max} = 50$ as a compromise between accuracy and computational time. The dimensions of all the videos can be seen in Table 3.

Table 3. Dimensions of the analyzed videos.

Video	**Number of frames τ**	**Number of pixels m**
`Fire`	110	428220
`Smoke`	40	172800
`Waves`	210	402000
`Waves and ship`	250	304128

The results of the synthesis error experiment are shown in Fig. 2. We can see that when $n = \tau - 1$ we have an *exact reconstruction* of the original video (as for the video `smoke`), but *no compression*. For the videos `fire` and `waves` we have $e_S(n_{max}) = 0$ without reaching the videos length, because we only need a portion of the singular values to describe Y_1^τ (the low-rank approximation of Y_1^τ is an exact decomposition for $n = n_{max}$). In this case, we have *both compression and perfect reconstruction.*

The performances of all the algorithms on the `waves and ship` DT suffer from two problems: (i) it is not possible to perfectly reconstruct the video with $n \le \tau - 1 = 249$; (ii) the video itself does not fully represent a DT, but it can be divided in two portions. The first one is the motion of the ship on the river (which does not exhibit the properties needed in order to be a DT), and the second one is the dynamics of the waves which is actually a DT.

In general, DMD and DMDc present lower errors and lower sensitivity to n compared to the Classic approach. In particular, this algorithm exhibits some outliers with high error, which represent synthesized videos that greatly differ from the original. The reason behind this is that the stationarity hypothesis does not apply for certain orders and therefore the covariance of the process $x(t)$ diverges, causing the generation of incorrect frames. Keeping this in mind, we can point out that, at least for DMD and DMDc, by *increasing n we increase the quality of the synthesized video.* However, visually, some of the videos synthesized

[3] The initial state of the dynamical system, x_0, can be extracted from the matrix $\hat{X}_1^\tau$ for the Classic approach while it can be estimated as $\hat{x}_0(\tau) = U_Y^T \cdot y(1)$ for DMD and DMDc.

by DMD and DMDc present little motion compared to the Classic approach.

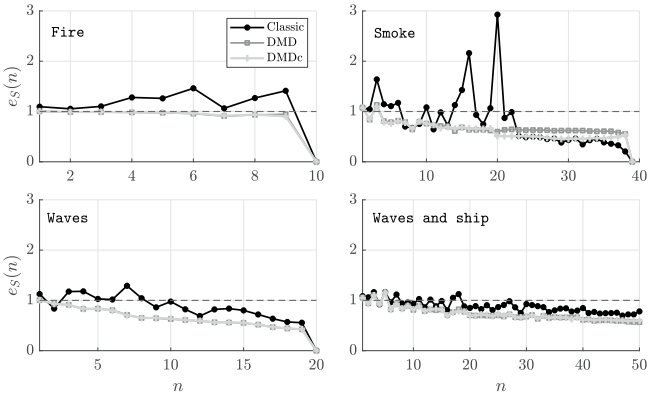


Fig. 2. Synthesis error for different videos.

The denoising performances are reported in Fig. 3. Most of the times, the synthesized video presents a lower noise level compared to Y_{noisy}. The quality of the reconstructed video depends both on the residual noise and the synthesis error. Therefore, choosing a lower n, which usually results in a lower noise, means synthesizing only a portion of the original video dynamics. Thus, there is a trade-off between the quality of the reconstructed dynamics of Y_1^τ and the residual noise. With regards to the videos `fire` and `waves`, for $n = n_{max}$ we obtain a $\hat{Y}_{noisy_n}$ which is similar to Y_1^τ and so we have succeeded both in denoising and in reconstructing the original video. Similarly to what we have seen in the first experiment, the Classic approach exhibits some outliers (for example, for the video `smoke` in Fig. 3) with residual noise higher than the added noise R. In fact, visually, the corresponding synthesized videos show more noise than Y_{noisy}.

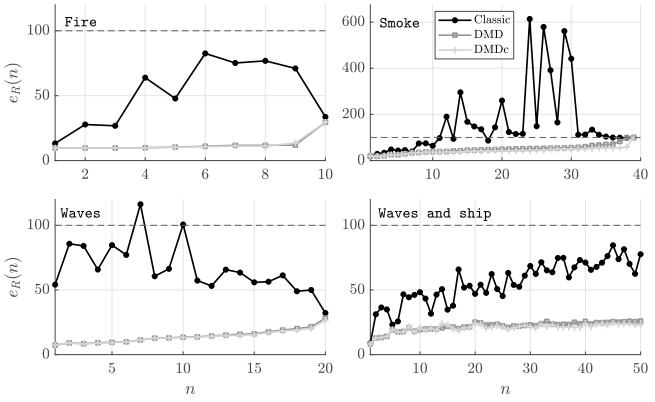


Fig. 3. Denoising error for different videos.

The goodness of the reconstruction from a visual point of view is not easily quantifiable and it is subjective. What we can deduce from these experiments is that *DMD and DMDc aim to minimize the synthesis error at the cost of the videos dynamics*. Instead, the Classic approach, for most orders, synthesizes videos that exhibit motion, but they may differ greatly from the original.

6. CONCLUSIONS

In this work, we presented the application of the DMD and DMDc algorithms to the DT identification problem. We formally compared these techniques to the Classic approach, highlighting the main similarities and differences. From the analysis of the computational complexities, we showed that the Classic approach and DMD have the same cost. From the experiments that compare the Classic approach to the DMD-like algorithms, we can see that the latter usually presents lower error and residual noise as well as lower sensitivity to the chosen system order. In particular, DMD and DMDc tend to minimize the noise and the synthesis error at the cost of the video dynamics, while the Classic approach generates videos that present motion that could be different from the original one.

REFERENCES

Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., and Werman, M. (2001). Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics*, 7(2), 120–135. doi: 10.1109/2945.928165.

Barzel, R. and Barr, A.H. (2013). *Physically-based modeling for computer graphics: a structured approach*. Morgan Kaufmann.

Bi, C., Yuan, Y., Zhang, J., Shi, Y., Xiang, Y., Wang, Y., and Zhang, R. (2018). Dynamic mode decomposition based video shot detection. *IEEE Access*, 6, 21397–21407. doi:10.1109/ACCESS.2018.2825106.

Doretto, G. and Soatto, S. (2003). Editable dynamic textures. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, II–137. doi:10.1109/CVPR.2003.1211463.

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

Golub, G.H. and Van Loan, C.F. (2013). *Matrix Computations*. Johns Hopkins University Press.

Grosek, J. and Kutz, J.N. (2014). Dynamic mode decomposition for real-time background/foreground separation in video. *arXiv preprint arXiv:1404.7592*.

Ljung, L. (1986). *System Identification: Theory for the User*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Peteri, R., Fazekas, S., and Huiskes, M. (2008). Dyntex: A comprehensive database of dynamic textures. *Elsevier*.

Proctor, J.L., Brunton, S.L., and Kutz, J.N. (2016). Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems*, 15(1), 142–161.

Rowley, C.W., Mezić, I., Bagheri, S., Schlatter, P., and Henningson, D.S. (2009). Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641, 115–127.

Saisan, P., Doretto, G., Ying Nian Wu, and Soatto, S. (2001). Dynamic texture recognition. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, II–II. doi:10.1109/CVPR.2001.990925.

Schmid, P.J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656, 5–28.

Wei, L.Y. and Levoy, M. (2000). Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 479–488. ACM Press/Addison-Wesley Publishing Co.

Zhao, X., Lin, Y., Liu, L., Heikkilä, J., and Zheng, W. (2019). Dynamic texture classification using unsupervised 3d filter learning and local binary encoding. *IEEE Transactions on Multimedia*, 21(7), 1694–1708. doi:10.1109/TMM.2018.2890362.