Dynamic Mode Decomposition for Background Modeling

J. N. Kutz, N. B. Erichson, and T. Askham
Applied Mathematics
University of Washington, Seattle, WA 98195
kutz@uw.edu

S. Pendergrass and S. L. Brunton
Mechanical Engineering
University of Washington, Seattle, WA 98195
sbrunton@uw.edu

Abstract

The Dynamic Mode Decomposition (DMD) is a spatio-temporal matrix decomposition method capable of background modeling in video streams. DMD is a regression technique that integrates Fourier transforms and singular value decomposition. Innovations in compressed sensing allow for a scalable and rapid decomposition of video streams that scales with the intrinsic rank of the matrix, rather than the size of the actual video. Our results show that the quality of the resulting background model is competitive, quantified by the F-measure, recall and precision. A GPU (graphics processing unit) accelerated implementation is also possible allowing the algorithm to operate efficiently on streaming data. In addition, it is possible to leverage the native compressed format of many data streams, such as HD video and computational physics codes that are represented sparsely in the Fourier domain, to massively reduce data transfer from CPU to GPU and to enable sparse matrix multiplications.

1. Introduction

Matrix decompositions are of fundamental importance for constructing subspaces for the efficient representation of data. Indeed, matrix decomposition algorithms have dominated both scientific computing and the data sciences by allowing for the construction of low-rank features (e.g. feature engineering) that dominate the sampled data. Levering the low-rank feature space, for instance, is at the core of statistical and machine learning methods. The singular value decomposition (SVD) is the algorithmic workhorse for the most common matrix decomposition methods such as principle component analysis (PCA) and the proper orthogonal decomposition (POD). An alternative to the SVD is the *Dynamic Mode Decomposition* (DMD) [19] which capitalizes on the low-rank feature extraction of SVD while augmenting it with an eigendecomposition in the time variable. Thus DMD provides a decomposition of data into spatio-temporal modes that correlates the data across spatial features (like principal component analysis (PCA)), but also pins the correlated data to unique temporal Fourier modes. For the specific application of video analysis, the time snapshots of video streams are used to compose matrices that are high-dimensional, but which often have a high-degree of correlation between frames. Understanding the correlation structure between time frames is fundamental for accurate and real-time background modeling techniques. Specifically, background variations in a video stream are highly correlated between frames so that the low-level computer vision task of foreground/background separation is typically an integral step in detecting, identifying, tracking, and recognizing objects in video streams.

DMD has its roots in the fluid dynamics community, where it was applied to the analysis of numerical simulations and experimental data of fluid flows [27, 26]. Over the past decade, its popularity has grown and it has been applied as a diagnostic tool, as a means of model order reduction, and as a component of optimal controller design for a variety of dynamical systems. The DMD also has connections to the Koopman spectral analysis of nonlinear dynamical systems [24]. In particular, the DMD shows promise as a data-driven tool for the analysis video streams which we consider as a dynamical systems [13, 11, 10, 21], and for which we use the DMD architecture to approximate its low-rank components. For computer vision applications, algorithms are envisioned to be implemented in real-time on high-definition video streams. Given the importance of this task for surveillance and target tracking/acquisition, a variety of matrix decomposition techniques have already been developed. For instance, a number of iterative (optimization and gradient descent based) techniques have already been developed in order to perform background/foreground separation [22, 32, 23, 15, 7]. We point the reader to several recent reviews [2, 3, 28, 29, 5] and a textbook [31] which highlight many of the methods developed and their performance metrics. In this work, we compare the basic DMD architecture and its various innovations with many of the leading background modeling methods.

2. DMD and the optimized DMD algorithm

The DMD method provides a spatio-temporal decomposition of data into a set of dynamic modes that are derived from snapshots or measurements of a given system in time. In this case, the snapshots are frames of a video sequence. The data collection process involves two parameters:

$n =$ number of spatial points saved per time snapshot
$m =$ number of snapshots taken

2.1. Exact DMD

The leading algorithm for computing the DMD decomposition is the *exact DMD* [33]. *Exact DMD* is defined for pairs of data $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_m, \mathbf{y}_m)\}$ which we assume satisfy $\mathbf{y}_j = \mathbf{A}\mathbf{x}_j$, for some matrix $\mathbf{A}$. Typically, the pairs are assumed to be given by equispaced snapshots of some dynamical system (video stream) $\mathbf{z}(t)$, i.e. $\mathbf{x}_j = \mathbf{z}((j-1)\Delta t)$ and $\mathbf{y}_j = \mathbf{z}(j\Delta t)$, but they are not required to be of this form. For most data sets, the matrix $\mathbf{A}$ is not determined fully by the snapshots. Therefore, we define the matrix $\mathbf{A}$ from the data in a least- squares sense. In particular, we set

$$\mathbf{A} = \mathbf{Y}\mathbf{X}^\dagger \ , \tag{2}$$

where $\mathbf{X}^\dagger$ is the pseudo-inverse of $\mathbf{X}$. The matrix $\mathbf{A}$ above is the minimizer of $\|\mathbf{AX} - \mathbf{Y}\|_F$ in the case that $\mathbf{AX} = \mathbf{Y}$ is over-determined and the minimum norm ($\|\mathbf{A}\|_F$) solution of $\mathbf{AX} = \mathbf{Y}$ in the case that the equation is under-determined [34] ($\| \cdot \|_F$ denotes the standard Frobenius norm). We may say that $\mathbf{A}$ is the best fit linear system mapping $\mathbf{X}$ to $\mathbf{Y}$ or, in the typical application, the best fit linear map which advances $\mathbf{z}(t)$ to $\mathbf{z}(t+\Delta t)$ (this map is sometimes referred to as a forward propagator). The DMD is then defined to be the set of eigenvectors and eigenvalues of $\mathbf{A}$ which can be computed using Algorithm 1.

The low-rank DMD approximation of both the eigenvalues and eigenvectors allows for a construction of the past, current and future state of the system. By first rewriting for convenience $\omega_k = \ln(\lambda_k)/\Delta t$, where Δt is the time between frames, then the approximate solution at all future times, $\tilde{\mathbf{x}}(t)$, is given by

$$\tilde{\mathbf{x}}(t) = \sum_{k=1}^{K} b_k(0)\boldsymbol{\psi}_k(\boldsymbol{\xi})\exp(\omega_k t) = \boldsymbol{\Psi}\mathrm{diag}(\exp(\boldsymbol{\omega} t))\mathbf{b} \tag{8}$$

where $\boldsymbol{\xi}$ are the spatial coordinates, $b_k(0)$ is the initial amplitude of each mode, $\boldsymbol{\Psi}$ is the matrix whose columns are the DMD modes $\boldsymbol{\psi}$, diag(ωt) is a diagonal matrix whose entries are the eigenvalues $\exp(\omega_k t)$, and $\mathbf{b}$ is a vector of the coefficients b_k.

It only remains to compute the initial coefficient values $b_k(0)$. If we consider the initial snapshot ($\mathbf{x}_1$) at time $t_1 =$

Algorithm 1 Exact DMD [33]

1. Define matrices $\mathbf{X}$ and $\mathbf{Y}$ from the data:
$$\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_m) \ , \qquad \mathbf{Y} = (\mathbf{y}_1, \dots, \mathbf{y}_m) \ . \tag{3}$$
2. Take the (reduced) SVD of the matrix $\mathbf{X}$, i.e. compute $\mathbf{U}$, $\boldsymbol{\Sigma}$, and $\mathbf{V}$ such that
$$\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^* \ , \tag{4}$$
where $\mathbf{U} \in \mathbb{C}^{n\times r}$, $\boldsymbol{\Sigma} \in \mathbb{C}^{r\times r}$, and $\mathbf{V} \in \mathbb{C}^{m\times r}$, with r the rank of $\mathbf{X}$.
3. Let $\tilde{\mathbf{A}}$ be defined by
$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{Y}\mathbf{V}\boldsymbol{\Sigma}^{-1} \ . \tag{5}$$
4. Compute the eigendecomposition of $\tilde{\mathbf{A}}$, giving a set of r vectors, $\mathbf{w}$, and eigenvalues, λ, such that
$$\tilde{\mathbf{A}}\mathbf{w} = \lambda\mathbf{w} \ . \tag{6}$$
5. For each pair (w, λ), we have a DMD eigenvalue, λ itself, and a DMD mode defined by
$$\boldsymbol{\psi} = \frac{1}{\lambda}\mathbf{Y}\mathbf{V}\boldsymbol{\Sigma}^{-1}\mathbf{w} \ . \tag{7}$$

0, let's say, then (8) gives $\mathbf{x}_1 = \boldsymbol{\Psi}\mathbf{b}$. This generically is not a square matrix so that its solution

$$\mathbf{b} = \boldsymbol{\Psi}^\dagger\mathbf{x}_1 \tag{9}$$

can be found using a pseudo-inverse. Indeed, $\boldsymbol{\Psi}^\dagger$ denotes the Moore-Penrose pseudo-inverse. The pseudo-inverse is equivalent to finding the best solution $\mathbf{b}$ the in the least-squares (best fit) sense. This is equivalent to how DMD modes were derived originally.

2.2. Optimized DMD

The exact DMD algorithm gives rise to a biased estimate for the solution [16, 9]. Specifically, the bias of the algorithm can be computed analytically [9]. If m is the number of snapshots and n is the dimension of the system, the bias will be the dominant component of the DMD error whenever $\sqrt{m}R_{STN} > \sqrt{n}$, where R_{STN} is the signal-to-noise ratio. The bias can be greatly suppressed using the recently developed optimized DMD algorithm [1], making it a substantial improvement over the exact DMD algorithm.

To this end, we present a simple algorithm for computing an optimized version of the DMD, even when sampled at different time intervals. By making use of the variable projection method for nonlinear least squares problems, the algorithm is capable of solving the underlying nonlinear optimization problem efficiently.

Let $\mathbf{X} = (\mathbf{z}_0, \dots, \mathbf{z}_m)$ be a matrix of snapshots, with $\mathbf{z}_j = \mathbf{z}(t_j) \in \mathbb{C}^n$ for a set of times t_j. For a target rank r, we assume that the data is the solution of a linear system of r differential equations ($d\mathbf{z}/dt = \mathbf{A}\mathbf{z}$) so that

$$\mathbf{z}(t) \approx \mathbf{S} e^{\mathbf{\Lambda} t} \mathbf{S}^\dagger \mathbf{z}_0 \ , \tag{10}$$

where $\mathbf{S} \in \mathbb{C}^{n \times r}$ and $\mathbf{\Lambda} \in \mathbb{C}^{r \times r}$. This can be rewritten as

$$\mathbf{X}^\intercal \approx \mathbf{\Phi}(\boldsymbol{\alpha}) \mathbf{B} \ , \tag{11}$$

where

$$B_{i,j} = S_{j,i} \left(\mathbf{S}^\dagger \mathbf{z}_0 \right)_i \tag{12}$$

and $\mathbf{\Phi}(\boldsymbol{\alpha}) \in \mathbb{C}^{(m+1) \times r}$ with entries defined by $\Phi(\boldsymbol{\alpha})_{i,j} = \exp(\alpha_j t_i)$.

The preceding formulation gives the following definition of the optimized DMD in terms of an exponential fitting problem. Suppose that $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{B}}$ solve

$$\text{minimize} \|\mathbf{X}^\intercal - \mathbf{\Phi}(\boldsymbol{\alpha}) \mathbf{B}\|_F \qquad \text{over } \boldsymbol{\alpha} \in \mathbb{C}^k, \mathbf{B} \in \mathbb{C}^{l \times n} \ . \tag{13}$$

The optimized DMD eigenvalues are then defined by $\lambda_i = \hat{\alpha}_i$ and the eigenmodes are defined by

$$\boldsymbol{\varphi}_i = \frac{1}{\|\hat{\mathbf{B}}^\intercal(:, i)\|_2} \hat{\mathbf{B}}^\intercal(:, i) \ , \tag{14}$$

where $\hat{\mathbf{B}}^\intercal(:, i)$ is the i-th column of $\hat{\mathbf{B}}^\intercal$. Execution of this architecture is demonstrated in Algorithm 2, which is essentially equivalent to that of Chen et al. [8].

If we set $b_i = \|\hat{\mathbf{B}}^\intercal(:, i)\|_2$, then

$$\tilde{\mathbf{z}_j} = \sum_{i=1}^{r} b_i e^{\lambda_i t_j} \boldsymbol{\varphi}_i \tag{15}$$

is an approximation to $\mathbf{z}_j$ for each $j = 0, \dots, m$. Therefore, if $\hat{\mathbf{B}}$ and $\hat{\boldsymbol{\alpha}}$ are computed, the reconstruction in the optimized DMD basis is trivial.

For a single initial guess for $\boldsymbol{\alpha}$, the Levenberg-Marquardt algorithm will not necessarily converge to the global minimizer of (13). It is therefore technically incorrect to claim that Alg. 2 will always compute the optimized DMD of a given set of snapshots. Indeed, it may be that the proper way to view the optimized DMD algorithm is as post-processors for the initial guess for $\boldsymbol{\alpha}$, which improve on $\boldsymbol{\alpha}$ by computing a nearby local minimizer. In the examples, we see that this post-processing — even when it is unclear whether we've computed the global minimizer — provides significant improvement over other DMD methods.

Algorithm 2 Optimized DMD

1. Let the snapshot matrix $\mathbf{X}$ and an initial guess for $\boldsymbol{\alpha}$ be given.
2. Solve the problem

   $$\text{minimize} \|\mathbf{X}^\intercal - \mathbf{\Phi}(\boldsymbol{\alpha}) \mathbf{B}\|_F \qquad \text{over } \boldsymbol{\alpha} \in \mathbb{C}^k, \mathbf{B} \in \mathbb{C}^{l \times n} \ , \tag{16}$$

   using a variable projection algorithm.
3. Set $\lambda_i = \hat{\alpha}_i$ and

   $$\boldsymbol{\varphi}_i = \frac{1}{\|\hat{\mathbf{B}}^\intercal(:, i)\|_2} \hat{\mathbf{B}}^\intercal(:, i) \ , \tag{17}$$

   saving the values $b_i = \|\hat{\mathbf{B}}^\intercal(:, i)\|_2$.

The asymptotic cost of Alg. 2 can be estimated. For each iteration of the variable projection algorithm, the cost is $\mathcal{O}(r^2 mn)$. For large m and n, it is possible to compute the optimized DMD (or an approximation of the optimized DMD) more efficiently. Suppose that instead of computing $\hat{\boldsymbol{\alpha}}$ and $\hat{\mathbf{B}}$ which solve 13, $\breve{\boldsymbol{\alpha}}$ and $\breve{\mathbf{B}}$ are computed which solve

$$\text{minimize} \|\mathbf{X}_r^\intercal - \mathbf{\Phi}(\boldsymbol{\alpha}) \mathbf{B}\|_F \qquad \text{over } \boldsymbol{\alpha} \in \mathbb{C}^k, \mathbf{B} \in \mathbb{C}^{l \times n} \ , \tag{18}$$

where $\mathbf{X}_r$ is the optimal rank r approximation of $\mathbf{X}$ (in the Frobenius norm). Algorithm 3 computes the solution to this problem.

The cost of computing the rank r SVD in step 2 of Alg. 3 is $\mathcal{O}(mn \min(m, n))$ using a standard algorithm or $\mathcal{O}(r^2(m + n) + rmn)$ using a randomized algorithm [14] (the constant is larger for the randomized algorithm, so determining the faster algorithm can be subtle). After this is computed once, the cost for each step of the variable projection algorithm is improved to $\mathcal{O}(r^3 m)$. This can lead to significant speed ups over the original.

3. DMD innovations

Two additional innovations are critical for making the DMD algorithm viable for HD video processing: (i) the ability to significantly subsample pixel space in order to go to scale to high-quality video streams, and (ii) the ability to update the background model in real-time using streaming data. The basic ideas and their performance are given in the following subsections.

3.1. Sparse sampling

Compressed DMD provides a computationally efficient framework to compute the dynamic mode decomposition on massively under-sampled or compressed data [6].

Algorithm 3 Approximate optimized DMD

1. Let the snapshot matrix $\mathbf{X}$ and an initial guess for $\boldsymbol{\alpha}$ be given.

2. Compute the truncated SVD of $\mathbf{X}$ of rank r, i.e. compute $\mathbf{U}_r \in \mathbb{C}^{n\times r}$ $\mathbf{\Sigma}_r \in \mathbb{C}^{r\times r}$, and $\mathbf{V}_r \in \mathbb{C}^{(m+1)\times r}$ such that
$$\mathbf{X}_r = \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^* \ . \tag{19}$$

3. Compute $\grave{\boldsymbol{\alpha}}$ and $\grave{\mathbf{B}}$ which solve
$$\text{minimize}\|\bar{\mathbf{V}}_r\mathbf{\Sigma}-\mathbf{\Phi}(\boldsymbol{\alpha})\mathbf{B}\|_F \qquad \text{over } \boldsymbol{\alpha}\in\mathbb{C}^r, \mathbf{B}\in\mathbb{C}^{r\times r} \tag{20}$$
using a variable projection algorithm.

4. Set $\lambda_i = \grave{\alpha}_i$ and
$$\boldsymbol{\varphi}_i = \frac{1}{\|\mathbf{U}_r\grave{\mathbf{B}}^\intercal(:,i)\|_2}\mathbf{U}_r\grave{\mathbf{B}}^\intercal(:,i)\ , \tag{21}$$
saving the values $b_i = \|\mathbf{U}_r\grave{\mathbf{B}}^\intercal(:,i)\|_2$.

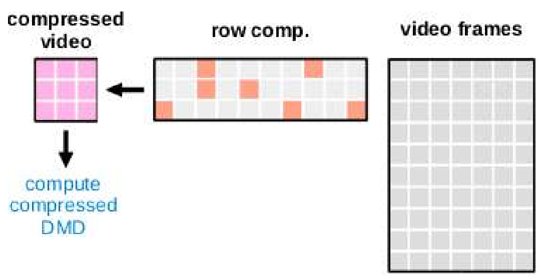

Figure 1: Video compression using a sparse measurement matrix. The compressed matrix faithfully captures the essential spectral information of the video.

The method was originally devised to reconstruct high-dimensional, full-resolution DMD modes from sparse, spatially under-resolved measurements by leveraging compressed sensing. However, it was quickly realized that if full-state measurements are available, many of the computationally expensive steps in DMD may be computed on a compressed representation of the data, providing dramatic computational savings.

The compressed DMD algorithm proceeds similarly to the standard DMD algorithm, either exact or optimized, at nearly every step until the computation of the DMD modes. The key difference is that we first compute a compressed representation of the video sequence, as illustrated in Figure 1. Hence the algorithm starts by generating the measurement matrix $\mathbf{C}\in\mathbb{R}^{p\times n}$ in order to compresses the data matrices:
$$\mathbf{Y} = \mathbf{CX}, \quad \mathbf{Y}' = \mathbf{CX}'. \tag{22}$$
where p is denoting the number of *samples* or *measurements*. There is a fundamental assumption that the input data are low-rank. This is satisfied for video data, because each of the columns of $\mathbf{X}$ and $\mathbf{X}'\in\mathbb{R}^{n\times m-1}$ are sparse in some transform basis $\boldsymbol{\Psi}$. Thus, for sufficiently many incoherent measurements, the compressed matrices $\mathbf{Y}$ and $\mathbf{Y}'\in\mathbb{R}^{p\times m-1}$ have similar correlation structures to their high-dimensional counterparts. Then, compressed DMD approximates the eigenvalues and eigenvectors of the linear map $\mathbf{A_Y}$, where the estimator is defined as:
$$\hat{\mathbf{A}}_\mathbf{Y} = \mathbf{Y}'\mathbf{Y}^\dagger \tag{23a}$$
$$= \mathbf{Y}'\mathbf{V_Y}\mathbf{S}_\mathbf{Y}^{-1}\mathbf{U_Y}^*, \tag{23b}$$
where $*$ denotes the conjugate transpose. The pseudo-inverse $\mathbf{Y}^\dagger$ is computed using the SVD:
$$\mathbf{Y} = \mathbf{U_Y}\mathbf{S_Y}\mathbf{V_Y}^*, \tag{24}$$
where the matrices $\mathbf{U}\in\mathbb{R}^{p\times k}$, and $\mathbf{V}\in\mathbb{R}^{m-1\times k}$ are the truncated left and right singular vectors. The diagonal matrix $\mathbf{S}\in\mathbb{R}^{k\times k}$ has the corresponding singular values as entries. Here k is the target-rank of the truncated SVD approximation to $\mathbf{Y}$. Note that the subscript $\mathbf{Y}$ is included to explicitly denote computations involving the compressed data $\mathbf{Y}$.

As in the standard DMD algorithm, we typically do not compute the large matrix $\hat{\mathbf{A}}_\mathbf{Y}$, but instead compute the low-dimensional model projected onto the left singular vectors:
$$\tilde{\mathbf{A}}_\mathbf{Y} = \mathbf{U_Y}^*\hat{\mathbf{A}}_\mathbf{Y}\mathbf{U_Y} \tag{25a}$$
$$= \mathbf{U_Y}^*\mathbf{Y}'\mathbf{V_Y}\mathbf{S}_\mathbf{Y}^{-1}. \tag{25b}$$
Since this is a similarity transform, the eigenvectors and eigenvalues can be obtained from the eigendecomposition of $\tilde{\mathbf{A}}_\mathbf{Y}$
$$\tilde{\mathbf{A}}_\mathbf{Y}\mathbf{W_Y} = \mathbf{W_Y}\mathbf{\Lambda_Y}, \tag{26}$$
where columns of $\mathbf{W_Y}$ are eigenvectors ϕ_j and $\mathbf{\Lambda_Y}$ is a diagonal matrix containing the corresponding eigenvalues λ_j. The similarity transform implies that $\mathbf{\Lambda}\approx\mathbf{\Lambda_Y}$. The compressed DMD modes are consequently given by
$$\mathbf{\Phi_Y} = \mathbf{Y}'\mathbf{V_Y}\mathbf{S}_\mathbf{Y}^{-1}\mathbf{W_Y}. \tag{27}$$
Finally, the full DMD modes are recovered using
$$\mathbf{\Phi} = \mathbf{X}'\mathbf{V_Y}\mathbf{S}_\mathbf{Y}^{-1}\mathbf{W_Y}. \tag{28}$$
Note that the compressed DMD modes in Equation (28) make use of the full data $\mathbf{X}'$ as well as the linear transformations obtained using the compressed data $\mathbf{Y}$ and $\mathbf{Y}'$.

Algorithm 4 Compressed Dynamic Mode Decomposition. Given a matrix $\mathbf{D} \in \mathbb{R}^{n\times m}$ containing the flattened video frames, this procedure computes the approximate dynamic mode decomposition, where $\mathbf{\Phi} \in \mathbb{C}^{n\times k}$ are the DMD modes, $\mathbf{b} \in \mathbb{C}^{k}$ are the amplitudes, and $\mathcal{V} \in \mathbb{C}^{k\times m}$ is the Vandermonde matrix describing the temporal evolution. The procedure can be controlled by the two parameters k and p, the target rank and the number of samples respectively. It is required that $n \geq m$, integer $k, p \geq 1$ and $k \ll m$ and $p \geq k$.

```
        function $[\mathbf{\Phi}, \mathbf{b}, \mathcal{V}] = \texttt{cdmd}(\mathbf{D}, k, p)$
(1)         $\mathbf{X}, \mathbf{X}' = \mathbf{D}$                                  Left/right snapshot sequence.
(2)         $\mathbf{C} = \texttt{rand}(p, m)$                                  Draw $p \times m$ sensing matrix.
(3)         $\mathbf{Y}, \mathbf{Y}' = \mathbf{C} * \mathbf{D}$                          Compress input matrix.
(4)         $\mathbf{U}, \mathbf{S}, \mathbf{V} = \texttt{svd}(\mathbf{Y}, k)$                     Truncated SVD.
(6)         $\tilde{\mathbf{A}} = \mathbf{U}^* * \mathbf{Y}' * \mathbf{V} * \mathbf{S}^{-1}$                Least squares fit.
(7)         $\mathbf{W}, \mathbf{\Lambda} = \texttt{eig}(\tilde{\mathbf{A}})$                       Eigenvalue decomposition.
(8)         $\mathbf{\Phi} \leftarrow \mathbf{X}' * \mathbf{V} * \mathbf{S}^{-1} * \mathbf{W}$             Compute full-state modes $\mathbf{\Phi}$.
(9)         $\mathbf{b} = \texttt{lstsq}(\mathbf{\Phi}, \mathbf{x}_1)$                       Compute amplitudes using $\mathbf{x}_1$ as intial condition.
(10)        $\mathcal{V} = \texttt{vander}(\texttt{diag}(\mathbf{\Lambda}))$                Vandermonde matrix (optional).
```

The expensive SVD on $\mathbf{X}$ is bypassed, and it is instead performed on $\mathbf{Y}$. Depending on the compression ratio, this may provide significant computational savings. The computational steps are summarized in Algorithm 4 using the exact DMD architecture for illustrative purposes. Further numerical details are presented in [6].

In order to compare the cDMD algorithm with other background modeling algorithms the BMC dataset has been used with the exact DMD algorithm [10]. Table 1 shows the evaluation results computed with the BMC wizard for all 9 videos. An individual threshold value has been selected for each video to compute the foreground mask. For comparison the evaluation results of 3 other RPCA methods are shown [4]. Overall cDMD achieves an average F-value of about 0.648. This is slightly better than the performance of GoDec [35] and nearly as good as LSADM [12]. However, it is lower than the F-measure achieved with the RSL method [17]. Figure 2 presents visual results for example frames across 5 videos. The last row shows the smoothed (median filtered) foreground mask.

Figure 3 shows the average frames per seconds (fps) rate required to obtain the foreground mask for varying video resolutions. The results illustrate the substantial computational advantage of the cDMD algorithm over the standard DMD. The computational savings are mainly achieved by avoiding the expensive computation of the singular value decomposition. Specifically, the compression step reduces the time complexity from $O(knm)$ to $O(kpm)$. The computation of the full modes $\mathbf{\Phi}$ in Equation 28 remain the only computational expensive step of the algorithm. However, this step is embarrassingly parallel and the computational time can be further reduced using a GPU accelerated implementation. The decomposition of a HD 1280×720 videos feed using the GPU accelerated implementation achieves a speedup of about 4 and 21 compared to the corresponding CPU cDMD and (exact) DMD implementations. The speedup of the GPU implementation can even further be increased using sparse or single pixel (sPixel) measurement matrices.

3.2. Streaming data

In many applications, data is continually acquired from sensors in a streaming fashion; new data is appended as columns to the right of the matrix $\mathbf{X}$, while old columns may be removed from the left of $\mathbf{X}$ if necessary. In streaming applications, such as online video processing or windowed DMD on transient simulations, the cost of repeated DMD and SVD calculations may be prohibitively expensive. This motivates a suite of complementary techniques to accelerate repeated SVD and DMD computations for streaming data. The core of the streaming DMD algorithm is the streaming method of snapshots SVD, whereby redundant inner product computations in $\mathbf{X}^*\mathbf{X}$ are reused from one timestep to the next, reducing the SVD computational complexity from $\mathcal{O}(mn^2)$ to $\mathcal{O}(mn)$. The streaming SVD can work both for long sequences of stored data, or for live inputs such as a video sequence. The streaming singular value decomposition works by reusing the majority of the calculations in $\mathbf{X}^*\mathbf{X}$, shifting the sub-matrix $(\mathbf{X}_2^n)^*(\mathbf{X}_2^n)$ up and left by one row and column.

As all but the last column of $\mathbf{X}$ will already have been run through the streaming algorithm, and as $\mathbf{X}^*\mathbf{X}$ is sym-

	Measure	BMC real videos									Average
		001	002	003	004	005	006	007	008	009	
RSL De La Torre et al. [17]	Recall	0.800	0.689	0.840	0.872	0.861	0.823	0.658	0.589	0.690	-
	Precision	0.732	0.808	0.804	0.585	0.598	0.713	0.636	0.526	0.625	-
	F-Measure	**0.765**	**0.744**	**0.821**	0.700	**0.706**	**0.764**	0.647	0.556	0.656	0.707
LSADM Goldfarb et al. [12]	Recall	0.693	0.535	0.784	0.721	0.643	0.656	0.449	0.621	0.701	-
	Precision	0.511	0.724	0.802	0.729	0.475	0.655	0.693	0.633	0.809	-
	F-Measure	0.591	0.618	0.793	0.725	0.549	0.656	0.551	**0.627**	**0.752**	0.650
GoDec Zhou and Tao [35]	Recall	0.684	0.552	0.761	0.709	0.621	0.670	0.465	0.598	0.700	-
	Precision	0.444	0.682	0.808	0.728	0.462	0.636	0.626	0.601	0.747	-
	F-Measure	0.544	0.611	0.784	0.718	0.533	0.653	0.536	0.600	0.723	0.632
cDMD	Recall	0.552	0.697	0.778	0.693	0.611	0.700	0.720	0.515	0.566	-
	Precision	0.581	0.675	0.773	0.770	0.541	0.602	0.823	0.510	0.574	-
	F-Measure	0.566	0.686	0.776	**0.730**	0.574	0.647	**0.768**	0.512	0.570	0.648

Table 1: Evaluation results of nine real videos from the BMC dataset. For comparison, the results of three other leading robust PCA algorithms are presented, adapted from [4].

metric, only the last row or column need be recalculated. For example, consider the SVD of $\mathbf{X}_1^{n-1}$:

$$(\mathbf{X}_1^{n-1})^*\mathbf{X}_1^{n-1} = \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x}_1\rangle & \langle \mathbf{x}_1, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_{n-1}\rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1\rangle & \langle \mathbf{x}_2, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_{n-1}\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle \mathbf{x}_{n-1}, \mathbf{x}_1\rangle & \langle \mathbf{x}_{n-1}, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_{n-1}, \mathbf{x}_{n-1}\rangle \end{bmatrix} \tag{29}$$

The entries indicated in blue will be redundant inner-products in the next SVD of the matrix $\mathbf{X}_2^n$:

$$(\mathbf{X}_2^{n})^*\mathbf{X}_2^{n} = \begin{bmatrix} \langle \mathbf{x}_2, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_{n-1}\rangle & \langle \mathbf{x}_2, \mathbf{x}_{n+1}\rangle \\ \vdots & \ddots & \vdots & \vdots \\ \langle \mathbf{x}_{n-1}, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_{n-1}, \mathbf{x}_{n-1}\rangle & \langle \mathbf{x}_{n-1}, \mathbf{x}_{n}\rangle \\ \langle \mathbf{x}_{n}, \mathbf{x}_2\rangle & \cdots & \langle \mathbf{x}_{n}, \mathbf{x}_{n-1}\rangle & \langle \mathbf{x}_{n}, \mathbf{x}_{n}\rangle \end{bmatrix} \tag{30}$$

This cuts the computational complexity down from $O(n^2m)$ to $O(nm)$. As this is the slowest part of the method of snapshots [33], a large performance gain is realized here.

The streaming dynamic mode decomposition relies on the streaming SVD in order to process data in sequence [25], but is also able to realize speed-ups from reusing intermediate steps from the SVD, and by only returning the last column of the matrix $\mathbf{S}$ in the case of background subtraction. Background subtraction with the streaming dynamic mode decomposition is performed by sliding the DMD forward by as many frames as the user wants to keep each iteration. We tested the algorithm with a step width of 1 frame against the non-streaming DMD slid forward by the width of $\mathbf{X}$ frames. When the step size is equal to the width of $\mathbf{X}$, no computations can be reused; thus the streaming and non-streaming DMD become equivalent in execution time. The smaller the number of frames incremented each iteration, the higher the quality of background subtraction and the longer the execution time. This means that the streaming DMD realizes the largest performance improvement for higher-quality small steps. Frames early in the window often have ”ghosting” issues not present when they are the last frame in the input sequence.

Figure 4 shows comparisons of the CPU, GPU, streaming CPU (SCPU) and streaming GPU (SGPU) implementations for the SVD, DMD and DMD background subtraction. This test shows streaming to significantly benefit the CPU implementation, putting it on par with that of the GPU. In real world applications, this is promising as it could reduce the need for a dedicated GPU, while still netting a large performance improvement and eliminating memory transfers. Further, the streaming GPU is significantly faster than the other three versions, and scales more favorably for large input dimensions (i.e., resolution m and number of frames n).

When the resolution is kept constant, we see that the scaling of both streaming algorithms is more favorable than that of the non-streaming algorithms. This is as expected, since that the cost to update $\mathbf{X}^*\mathbf{X}$ is on the order of $O(mn)$ when streaming, rather than $O(mn^2)$. Of particular note is the large jump in time from resolution 1440 to 2160 on all three varying height graphs (left column): this jump shows when the memory available to the GPU becomes an issue. Due to its reduced memory footprint, the streaming versions of each algorithm do not suffer from this memory bottleneck and associated reduction in performance. With regards to the graphs of varying width, we see that both streaming algorithms grow far more slowly than their non-streaming counterparts. This shows that in all cases the streaming CPU and GPU implementations of all 3 algorithms are faster than their traditional counterparts, at both high resolution and frame counts. Additionally, the streaming GPU versions of the SVD, DMD and DMD background subtraction algorithms all show the best potential for scaling

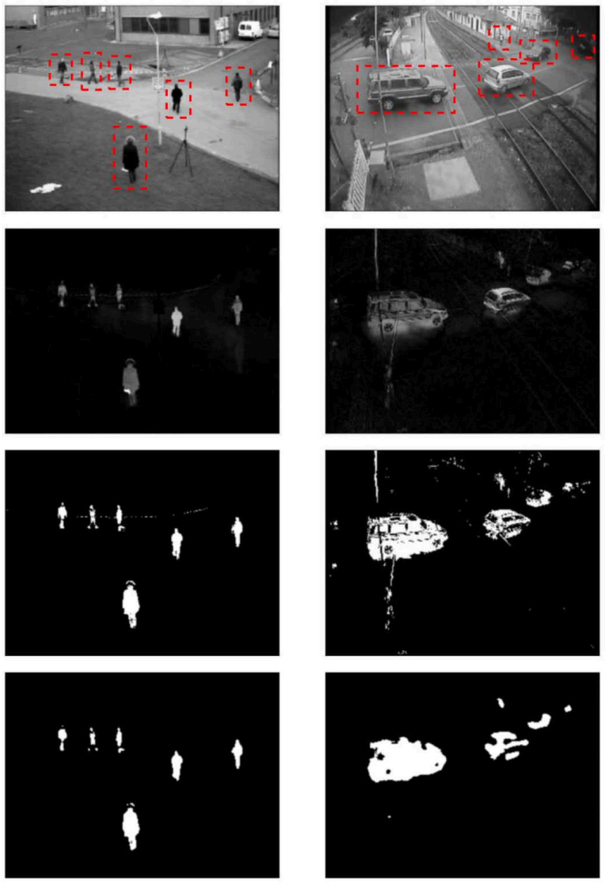

Figure 2: Visual evaluation results for 2 example frames corresponding to the BMC Videos: 003 and 006. The top row shows the original grayscale images (moving objects are highlighted). The second row shows the differencing between the reconstructed cDMD background and the original frame. Row three shows the thresholded and row four the in addition median filtered foreground mask.

to even higher-dimensional inputs.

4. Conclusions and Outlook

Matrix decompositions, such as the singular value decomposition (SVD) and dynamic mode decomposition (DMD), are the cornerstones of numerical linear algebra and data analysis methods. However, these methods typically become computationally intractable for high-dimensional data, and this cost is compounded in streaming applications, where a new matrix decomposition is required for each new measurement in time. These computational issues hinder efforts for real-time processing of high-dimensional data, such as HD and 4K videos, which will only get worse with growing big data volumes. More broadly, the emergence of the big data era across the physical, biological, social and engineering sciences has severely challenged our ability to extract meaningful features from data in a real-time manner. Critical technologies such as LIDAR, 4K video streams, computer vision, high-fidelity numerical simulations, sensor networks, brain-machine interfaces, internet of things, and augmented reality will all depend on scalable algorithms that can produce meaningful decompositions of data in real time.

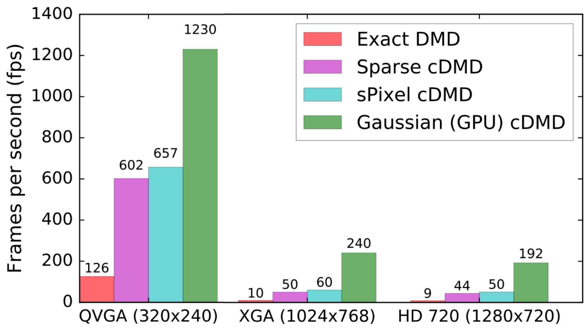

Figure 3: CPU and GPU algorithms runtime (including the computation of the foreground mask) for varying video resolutions (200 frames). The optimal target rank is automatically determined and $p = 1000$ samples are used.

By interpreting video streams as a dynamical system, the DMD algorithm can be leveraged to decompose video data into a set of dynamic modes that are derived from individual snap shots. We introduce a new optimized version of the DMD which removes many of the known problems associated with bias from the standard exact DMD algorithm. Moreover, we show that by using innovations in randomized linear algebra and streaming SVD architectures, significant enhancements in performance can be gained. We document these performance gains on challenge data sets in computer vision and demonstrate that the DMD algorithm is a viable technology for real-time video processing. Importantly, the innovations presented are modular and can be jointly integrated in order to boost performance gains.

Specific algorithmic improvements made to the DMD in-

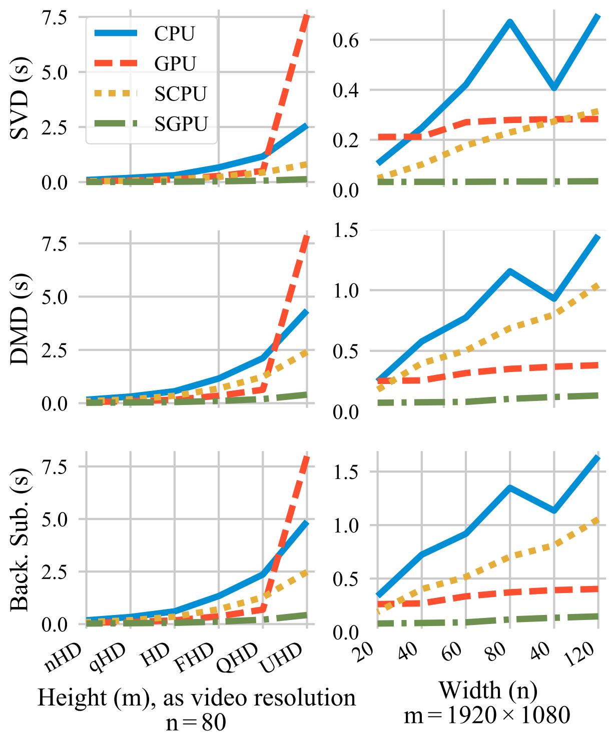

Figure 4: Comparison of CPU, streaming CPU (SCPU), GPU and streaming GPU (SGPU) versions of the SVD, DMD and DMD background subtraction. Times represent a one-frame update from steady state with all singular values in use. In the case of background subtraction, only the closest single or complex conjugate pair of λ to 1 is used. Timings are a best of 5 mean with memory transfers to and from the GPU excluded. Tests were run on PEViD "Walking Day Indoor 4" [18].

clude a streaming architecture that exploits redundant computations and that may be readily parallelized on a graphics processing unit (GPU), providing significant acceleration of the algorithm. We have developed and analyzed streaming singular value and dynamic mode decomposition algorithms and their GPU implementations. In addition, we show performance benefits for streaming video background subtraction. In all cases, a large number of calculations are able to be carried forward from frame to frame by exploiting the structure of the method of snapshots SVD. This allows both the SVD and DMD to process large data streams in real-time, whether for video or otherwise. We have evaluated the proposed algorithms on multiple datasets, demonstrating the significantly improved computational performance for stream processing with negligible loss in accuracy.

We have also demonstrated that randomized sampling via the cDMD algorithm remains competitive with other leading algorithms in the quality of the decomposition itself. Our results show, that for both standard and challenging environments, the cDMD's background subtraction accuracy in terms of the F-measure is competitive to leading RPCA based algorithms [4]. Though, the algorithm cannot compete, in terms of the F-measure, with highly specialized algorithms, e.g. optimized Gaussian mixture-based algorithms for background modeling [30]. The main difficulties arise when video feeds are heavily crowded or dominated by non-periodic dynamic background objects. Overall, the trade-off between speed and accuracy of compressed DMD is compelling.

Future work will aim to improve the background subtraction quality as well as to integrate a number of innovative techniques. One technique that is particularly useful for object tracking is the multi-resolution DMD [20]. This algorithm has been shown to be a potential method for target tracking applications. Thus one can envision the integration of multi-resolution ideas with cDMD, i.e. a multi-resolution compressed DMD method, in order to separate the foreground video into different dynamic targets when necessary.

References

[1] T. Askham and J. N. Kutz. Variable projection methods for an optimized dynamic mode decomposition. *arXiv preprint arXiv:1704.02343*, 2017.

[2] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. *Comparative Study of Background Subtraction Algorithms. Journal of Electronic Imaging 19 (2010)*, 19(3):033003, 2010.

[3] T. Bouwmans. *Recent advanced statistical background modeling for foreground detection: a systematic survey. RPCS*, 4(3):147–176, 2011.

[4] T. Bouwmans, A. Sobral, S. Javed, S. K. Jung, and E.-H. Zahzah. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset, 2015.

[5] T. Bouwmans and E. H. Zahzah. *Robust PCA via Principal Component Pursuit: A review for a comparative evaluation in video surveillance. Comp. Vis. Imag. Under.*, 122:22–34, 2014.

[6] S. Brunton, J. Proctor, and J. N. Kutz. *Compressive sampling and dynamic mode decomposition. Journal of Computational Dynamics*, to appear (2015).

[7] E. Candès, X. Li, Y. Ma, and J. Wright. *Robust Principal Component Analysis? Computing Research Repository*, abs/0912.3599, 2009.

[8] K. K. Chen, J. H. Tu, and C. W. Rowley. Variants of dynamic mode decomposition: boundary condition, koopman, and fourier analyses. *Journal of nonlinear science*, 22(6):887–915, 2012.

[9] S. T. Dawson, M. S. Hemati, M. O. Williams, and C. W. Rowley. Characterizing and correcting for the effect of sen-

sor noise in the dynamic mode decomposition. *Experiments in Fluids*, 57(3):1–19, 2016.

[10] N. B. Erichson, S. L. Brunton, and J. N. Kutz. Compressed dynamic mode decomposition for background modeling. *Journal of Real-Time Image Processing*, pages 1–14, 2016.

[11] N. B. Erichson and C. Donovan. Randomized low-rank dynamic mode decomposition for motion detection. *Computer Vision and Image Understanding*, 146:40–50, 2016.

[12] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming*, 141(1-2):349–382, 2013.

[13] J. Grosek and J. N. Kutz. *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video. arXiv*, page arXiv:1404.7592, 2014.

[14] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[15] J. He, L. Balzano, and A. Szlam. *Incremental Gradient on the Grassmannian for Online Foreground and Background Separation in Subsampled Video.* In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1568–1575, 2012.

[16] M. S. Hemati, C. W. Rowley, E. A. Deem, and L. N. Cattafesta. De-biasing the dynamic mode decomposition for applied koopman spectral analysis. *arXiv preprint arXiv:1502.03854*, 2015.

[17] M. Kleinsteuber, F. Seidel, and C. Hage. *pROST: A Smoothed ℓ_p-Norm Robust Online Subspace Tracking Method for Realtime Background Subtraction in Video. Machine Vision and Applications, Special Issue on Background Modeling for Foreground Detection in Real-World Dynamic Scenes*, 2013.

[18] P. Korshunov and T. Ebrahimi. UHD video dataset for evaluation of privacy. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*. IEEE, 09 2014.

[19] J. N. Kutz, S. L. Brunton, B. W. Brunton, and J. L. Proctor. Dynamic mode decomposition: Data-driven modeling of complex systems, 2016.

[20] J. N. Kutz, X. Fu, and S. L. Brunton. Multiresolution dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 15(2):713–735, 2016.

[21] J. N. Kutz, X. Fu, S. L. Brunton, and J. Grosek. Dynamic mode decomposition for robust pca with applications to foreground/background subtraction in video streams and multi-resolution analysis. *CRC Handbook on Robust Low-Rank and Sparse Matrix Decomposition: Applications in Image and Video Processing, T. Bouwmans Ed*, 2016.

[22] L. Li, W. Huang, I. Gu, and Q. Tian. *Statistical Modeling of Complex Backgrounds for Foreground Object Detection. IEEE Transactions on Image Processing*, 13(11):1459–1472, 2004.

[23] L. Maddalena and A. Petrosino. *A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications. IEEE Transactions on Image Processing*, 17(7):1168–1177, 2008.

[24] I. Mezić. *Analysis of Fluid Flows via Spectral Properties of the Koopman Operator. Annual Review of Fluid Mechanics*, 45:357–378, 2013.

[25] S. D. Pendergrass, J. N. Kutz, and S. L. Brunton. Streaming gpu singular value and dynamic mode decompositions. *arXiv preprint arXiv:1612.07875*, 2016.

[26] C. Rowley, I. Mezić, S. Bagheri, P. Schlatter, and D. Henningson. *Spectral analysis of nonlinear flows. Journal of Fluid Mechanics*, 641:115–127, 2009.

[27] P. Schmid. *Dynamic mode decomposition of numerical and experimental data. Journal of Fluid Mechanics*, 656:5–28, 2010.

[28] M. Shah, J. Deng, and B. Woodford. *Video Background Modeling: Recent Approaches, Issues and Our Solutions. Machine Vision and Applications, Special Issue on Background Modeling fro Foreground Detection in Real-World Dynamics*, 25:1105–1119, 2014.

[29] A. Shimada, Y. Nonaka, H. Nagahara, and R. Taniguchi. *Case Based Background Modeling-Towards Low-Cost and High-Performance Background Model. Machine Vision and Applications, Special Issue on Background Modeling fro Foreground Detection in Real-World Dynamics*, 25:1121–1131, 2015.

[30] A. Sobral and A. Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, 2014.

[31] E. T. Bouwmans. *Handbook on Robust Decomposition in Low Rank and Sparse Matrices and its Applications in Image and Video Processing.* CRC Press, 2015.

[32] Y. Tian, M. Lu, and A. Hampapur. *Robust and Efficient Foreground Analysis for Real-Time Video Surveillance.* In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.*, volume 1, pages 1182–1187, 2005.

[33] J. Tu, C. Rowley, D. Luchtenberg, S. Brunton, and J. N. Kutz. *On Dynamic Mode Decomposition: Theory and Applications. Journal of Computational Dynamics*, 1:391–421, 2014.

[34] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz. On dynamic mode decomposition: theory and applications. *arXiv preprint arXiv:1312.0041*, 2013.

[35] T. Zhou and D. Tao. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *International Conference on Machine Learning*, pages 1–8. ICML, 2011.