# Background Subtraction in Video Streams

Jiasui Qin

March 15, 2020

**Abstract**

This paper introduces the method of dynamic mode decomposition (DMD) for separating video frames into background (low-rank) and foreground (sparse) components in real-time. DMD is a dimension reduction algorithm that deals with sequential time series for which the measurement dimension is much larger than the number of measurements taken. I will explore the use of DMD and its performance on the problem of separating the foreground and the background of video clips.

## 1 Introduction and Overview

I have two video clips, each has a moving object in the foreground and non-moving objects in the background. My job is to separate the video stream to both the foreground video and the background video. To achieve this, I will apply the dynamic mode decomposition (DMD) technique. Different from the proper orthogonal decomposition (POD), DMD is a data-based algorithm insteasd of a model-based algorithm. It takes advantage of low-dimensionality in experimental data without having to rely on a given set of governing equations. The main strategy of DMD is to find the oscillations, decay, or growth given a time series of data. Since video frames are equally spaced in time, it would be applicable to use the DMD algorithm if we represent the data as a matrix with each column representing each frame. After this, we can use the DMD algorithm to find a basis of spatial modes that show us the oscillation frequency and decay/growth rate. Getting information about the Fourier frequencies allow us to distinguish the low-rank (background) and the sparse (foreground) components.

## 2 Theoretical Background

### 2.1 Dynamic Mode Decomposition (DMD)

To understand DMD, we first need to know what the Koopman operator is. Suppose we have a linear, time-independent Koopman operator $A$ such that $X_{j+1} = AX_j$, then $A$ maps the data from time $t_j$ to $t_{j+1}$. Applying $A$ helps us advance the data forward from one timestep to the next. Therefore, if columns 1 through $M-1$ of the full snapshot matrix $X$ is collected as:

$$X_1^{M-1} = [x_1 \quad x_2 \quad x_3 \dots x_{M1}] \tag{1}$$

We can rewrite it using the Koopman operator as below:

$$X_1^{M-1} = [x_1 \quad Ax_1 \quad A^2x_1 \dots A^{M-2}x_1] \tag{2}$$

The columns of $X_1^{M-1}$ form a Krylov space. Thus, we can get

$$X_2^M = AX_1^{M-1} + re_{M-1}^T \tag{3}$$

$T$ is the conjugate transpose operator, $e_{m1}$ is the $(m1)$th unit vector, and $r$ is the residual(or error) vector to account for the final point $x_M$ that is not included in the Krylov basis. We can then use the SVD to

represent $X_1^{M-1} = USV^T$, so $X_2^M = AUSV^T + re_{M-1}^T$. The residual vector $r$ is orthogonal to the POD basis, which gives us $U^T r = 0$. Therefore, if we multiply both sides of the equation by $U^T$, we would have

$$U^T X_2^M = U^T AUSV^T \tag{4}$$

$$U^T AU = U^T X_2^M VS^{-1} \tag{5}$$

If we denote the right side as $S$, we notice that $AU = US$. Thus, $A$ and $S$ are similar matrices, which means that eigenvalues of the matrix $S$ approximate the eigenvalues of the Koopman operator $A$. Since everything on the right side is known, we can compute the eigenvectors and eigenvalues of $S$, $k$ is the rand of $X_1^{M-1}$:

$$SW_j = u_j W_j, j = 1, 2, \dots k \tag{6}$$

Therefore, the approximate $j$th eigenvector of A, which is the $j$th DMD mode, can be represented as:

$$\varphi_j = UW_j \tag{7}$$

Then, the DMD reconstruction of $X_{DMD}$ at time t for any time after the initial vector $x1$ would be:

$$X_{DMD}(t) = \sum_{k=1}^{k} b_k \varphi_k e^{w_k t} = \varphi diag(e^{w_k t}) b \tag{8}$$

$b_k$ is the initial amplitude of the $k$th mode, and $\varphi_k$ is the $k$th eigenvector of A. In addition, we can convert DMD eigenvalues to Fourier modes by defining $w_k = ln(u_k)/\Delta t$. The real part of $w_k$ and the imaginary part of $w_k$ regulates the growth/decay of the DMD modes and the oscillations of the DMD modes separately. To compute $b$, we know that $x_1 = \varphi b$ at $t = 0$, so $b$ equals the pseudoinverse of $\varphi$ times $x_1$. [2]

# 3 Algorithm Implementation and Development

First of all, we need to represent our video as a matrix with each column representing each frame. Suppose there are $m$ frames in total, each frame has $n$ pixel values, then we can form a matrix with $x_1, x_2, ..., x_m$, which are $n \times 1$ vectors. Then, we can use formula (8) to reconstruct the video. As illustrated in the last section, by calculating $X_{DMD}(t)$ at the corresponding time $t$, we're able to reconstruct any given frames in the future. Therefore, we can first use the theory of DMD to construct the DMD solution that best approximates the nonlinear dynamics using the following steps:

1. Construct the two submatrices $X_1^{M-1}$ and $X_2^M$ from the matrix $X$ we formulate.
2. Compute the SVD of $X_1^{M-1}$ using the Matlab function svd()[3].
3. Compute the matrix $S$ using formula(5) and find its eigenvalues $u$ and eigenvectors $W$ using Matlab function eig()[1].
4. Compute the Fourier modes of eigenvalues using $w = ln(u)/\Delta t$.
5. Compute $\varphi = UW$ using eigenvectors of $S$.
6. Compute $b$ using the initial frame $x1$ and the pseudoinverse of $\varphi$.
7. Using the initial condition and the time dynamics computed to compute the solution at future time.

**Algorithm 1:** DMD

```
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);
[U, Sigma, V] = svd(X1, 'econ');
S = U' × X2 × V × diag(1/diag(Sigma));
[W, u] = eig(S);
w = log(diag(u))/dt;
φ = U × W;
b = φ \X1(:,1);
for iter = 1: total number of frames do
   modes(:,iter) = b × exp(w × t(iter));
end for
DMD = φ× modes;
```

After getting the DMD solution, we can then separate it into the low-rank (background) and the sparse (foreground) components. For the low-rank (background) information, we need to find the Fourier mode of eigenvalues $w$ that is located near the origin in complex space, which represents items that stay stable or change very slowly in time. Therefore, we can divide the DMD solution into:

$$\begin{aligned} X_{DMD} &= X_{DMD}^{\text{Low Rank}} + X_{DMD}^{\text{Sparse}} \\ &= b_p \varphi_p e^{w_p t} + \sum_{j \neq p} b_j \varphi_j e^{w_j t} \end{aligned} \tag{9}$$

We'll find the low-rank reconstruction and the sparse reconstruction using the following steps:

1. Compute the low-rank reconstruction $X_{DMD}^{\text{Low Rank}} = b_p \varphi_p e^{w_p t}$
2. Compute the sparse reconstruction using $X_{DMD}^{\text{Sparse}} = X - |X_{DMD}^{\text{Low Rank}}|$
3. To avoid negative pixel values in $X_{DMD}^{\text{Sparse}}$, put residual negative values into a matrix $R$ and add it back into $X_{DMD}^{\text{Low Rank}}$:

$$\begin{aligned} R + |X_{DMD}^{\text{Low Rank}}| &\to X_{DMD}^{\text{Low Rank}} \\ X_{DMD}^{\text{Sparse}} - R &\to X_{DMD}^{\text{Sparse}} \end{aligned} \tag{10}$$

# References


[1] *eig()*. URL: `https://au.mathworks.com/help/matlab/ref/eig.html`.

[2] J. Grosek and J. Nathan Kutz. *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video*. 2014. URL: `https://arxiv.org/pdf/1404.7592.pdf`.

[3] *svd()*. URL: `https://www.mathworks.com/help/matlab/ref/double.svd.html`.