Video Foreground-Background Separation Using Dynamic Mode Decomposition

Ben Francis,

March 17, 2021

Abstract

Dynamic mode decomposition is used to separate the foreground and background of videos. This is done with exceptional algorithmic simplicity and with strong grounding in the underlying dynamic mode decomposition theory. This was successfully done for two different videos, one with a small moving target and the other with multiple moving targets.

1 Introduction and Overview

The purpose of this paper is to use dynamic mode decomposition to separate out the foreground and background of a video. This highlights the capability of dynamic mode decomposition to separate out spatial coherent structures based upon their dynamic behavior. Since the background will not change, and since it represents a large proportion of the entire video feed, it is expected to be the spatial mode with the highest percent of captured variance, and to have essentially static dynamics.

This analysis is performed on short videos of a skier skiing down a mountain and a race car scene. These videos provide good test cases, due to the disparity between the small size of the skier and the larger size of the race cars, as well as the presence of multiple race cars.

2 Theoretical Background

2.1 Dynamic Mode Decomposition

Dynamic Mode Decomposition (DMD) is a data-driven method for linearizing the dynamics of non-linear dynamical systems. The intuition behind the method is that the behavior is driven by oscillations of non-linear spatial modes. If these spatial modes could be determined, then the dynamics of the system would simply evolve as the linear combination of these spatial modes oscillating at different frequencies, or exponentially growing or decaying. Determining these spatial modes is reminiscent of principal component analysis (PCA) and defining their oscillatory behavior is similar to the Fourier transform. Critically, this decomposition of the system into coherent spatiotemporal structures is entirely data-driven. Since the natural world is composed mostly of complex, non-linear systems whose dynamics are not yet well modeled by equations, this gives this relatively new method tremendous potential and scope.

Dynamical systems are defined by an equation of the form:

$$\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, t, \mu) \tag{1}$$

Here $\boldsymbol{x}$ is an n-dimensional vector, t represents time, and μ represents a set of parameters of the system. f then describes the dynamics of the system. A linear approximation of these dynamics can be given by:

$$\frac{d\boldsymbol{x}}{dt} = A\boldsymbol{x} \tag{2}$$

This system has the solution:

$$\boldsymbol{x}(t) = \sum_{k=1}^{n} \boldsymbol{\phi} \exp(\omega_k t) b_k = \boldsymbol{\Phi} \exp(\Omega t) \boldsymbol{b} \tag{3}$$

Φ represents the dynamic modes, Ω determines the time dynamics of these modes, and $\boldsymbol{b}$ determines the linear combination of these time-evolving dynamic modes at the initial time point of $\boldsymbol{x}(t)$. Since this is a data-driven method, there will only ever be discrete time samplings, which represent a subset of the continuous measurements, resulting in a difference equation instead of a differential equation. This is given by:

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k \tag{4}$$

The goal of the DMD algorithm is to minimize the error for the first $m-1$ measurements in the data:

$$||\boldsymbol{x}_{k+1} - A\boldsymbol{x}_k||_2 \tag{5}$$

Since this A may be very large, a low-rank approximation of A, given by $\tilde{A}$, will be produced instead. From this, the low-rank eigenvalues and eigenvectors of A can be determined. The regression for this transformation is only performed for the measurements given, so if there is non-stationarity in the system not exhibited in the provided data, this will not be able to extrapolate forward into the future. However, it services well as a diagnostic tool in either case, since the spatial modes and their dynamics-determining eigenvalues can still be studied.

The algorithm for determining the eigenvectors and eigenvalues of A will now be described. The goal of the algorithm is to determine a best fit linear transformation from measurements of a system in one time window to measurements one time step in the future. This is given by:

$$X' = AX \tag{6}$$

where X is the first time window of measurements and X' is the time window of measurements shifted one step in the future:

$$X = \left[\begin{array}{c|c|c|c} x_1 & x_2 & ... & x_{n-1} \end{array} \right], X' = \left[\begin{array}{c|c|c|c} x_2 & x_2 & ... & x_n \end{array} \right] \tag{7}$$

To approximate the leading eigenvalues and eigenvectors of A, the singular value decomposition (SVD) of X is taken, and the above relationship can be rewritten:

$$X = U\Sigma V^*, X' = AU\Sigma V^* \tag{8}$$

$$U^* X' V, X' = U^* A U = A \tag{9}$$

If only the dominant modes of the SVD are retained, then $\tilde{A}$ is constructed, which retains the variance of the data corresponding to how many modes are used. This is done by projecting A onto the dominant singular vectors, $U_r = U(:, 1:r)$, such that:

$$U_r^* A U_r = \tilde{A} \tag{10}$$

Importantly, $\tilde{A}$ shares the same eigenvalues as A, yet $\tilde{A}$ is proportionally smaller with the number of modes chosen in its construction, so it represents a more computationally efficient method for calculating the dominant eigenvalues of A. An eigendecomposition is performed to get these values:

$$\tilde{A}W = W\Lambda \tag{11}$$

W is the eigenvector matrix and Λ is the diagonal eigenvalue matrix. However, it is the eigenvectors of A, not $\tilde{A}$ that provide the dominant spatial structures of interest. The eigenvectors of A are given by the relationship:

$$\Phi = X' V \Sigma^{-1} W \tag{12}$$

The columns of Φ are the eigenvectors of A, which are called the DMD modes, or dynamic modes.

The data can then be reproduced from the initial measurement up to any time t. If not all of the modes were used, then this will represent a version of the data with less dynamic modes. This can also be used for predictive purposes, by selecting t values beyond those measured. This is done using:

$$\hat{X}(k\Delta t) = \Phi\Lambda^t \boldsymbol{b} \tag{13}$$

Here Φ represents the spacial coherent structures and Λ describes the time dynamics, and the vector b_0 is the linear combination of these modes specified by the initial conditions. By considering that Λ is a diagonal matrix, it can be seen that taking it to the tth power represents evolving each eigenvector over time. In order to get it into the form shown in equation 3, the transformation $\Omega = \log(\Lambda)/\Delta t$ is used. Since at time $t = 0$, $\Lambda^t = 1$, $\boldsymbol{b}$ can be found by:

$$\boldsymbol{b} = \Phi^\dagger \boldsymbol{x}_1 \tag{14}$$

$\Phi^\dagger$ is the pseudo-inverse of Φ, necessary since Φ is usually not a square matrix; this finds a least-squares best fit for b given $\boldsymbol{x}_1$. In contrast to other predictive tools, such as recurrent neural networks, DMD is readily interpretable by examining the eigenvalues and eigenvectors of Φ.

In order to remove the background, only the first few dominant dynamic modes will be used in the reconstruction of $\hat{X}$, for all t present in the original measurements. This will likely only contain the background, since it represents a large portion of the video with the same dynamics, since it is essentially constant. As a standalone result, this should be able to reconstruct the background of the video. Subtracting this from the original video should then only leave the foreground. [2][1]

3 Algorithm Implementation and Development

Algorithm 1: Data Pre-Processing

Data: Video V
Result: Video Background `background`, Video Foreground `foreground`
`num_frames, duration` $\leftarrow$ number of frames in, duration of video V;
$dt \leftarrow$ `duration / num_frames`, Time interval between measurements;
$X \leftarrow$ Convert video V to gray scale, double data type, reshape spatial dimensions to vector;
`m` $\leftarrow$ Width of X, number of time measurements;
$X1 \leftarrow$ First $m-1$ measurements of data;
$X2 \leftarrow$ Last $m-1$ measurements data;
$r \leftarrow$ Number of modes;
$X_{DMD} \leftarrow$ `DMD`$(X1, X2, r, dt)$, perform DMD algorithm, see below;
`background` $\leftarrow X_{DMD}$;
`foreground` $\leftarrow X - X_{DMD}$;
Add uniform background to `foreground`

Algorithm 2: DMD

Data: Measurements $X1$, Measurements $X2$, Number of modes r, time interval dt
Result: Data matrix reconstructed by using DMD modes, X_{DMD}
$U, S, V \leftarrow$ `svd`$(X1)$;
$U_r, S_r, V_r \leftarrow$ Truncate U, S, V to only r modes;
$\tilde{A} \leftarrow U_r' * X2 * V_r/S_r$, Approximate low-rank version of Koopman operator;
$[W_r, D] =$ `eig`$(\tilde{A})$;
$\Phi = X2 * V_r/S_r * W_r$, Determine dymanic modes;
$\Omega \leftarrow$ `log`(Eigenvalues of D)$/dt$;
$b \leftarrow$ Psuedo-inverse of Φ times first measurement of X_1 $X_{DMD} \leftarrow \Phi \exp(\Omega t)\boldsymbol{b}$, Reconstruct X based on low-dimensional dynamic modes;

Algorithm Development

Having a consistent difference in time between measurements is important in the basic DMD algorithm. This is given by both video recordings, since the video equipment records in this manner automatically. The video is then converted to grayscale, reshaped such that each measurement is represented by a vector, and converted to double type. The number of measurements is recorded, such that the data matrix can be separated out into the first $m-1$ measurements, in the new matrix $X1$ and the last $m-1$ measurements, in the new matrix $X2$. The number of modes to be used is determined, based on analysis of the singular mode spectrum of $X1$. These are all then input into the DMD algorithm.

The DMD algorithm takes the SVD of the first set of measurements $X1$ and truncates it to the specified number of modes. This is used to approximate the low-rank version of the Koopman operator, $\tilde{A}$. The eigenvalues and eigenvectors of this are determined, and the eigenvectors used to construct the low-rank dynamic modes, which represent the dominant coherent spatial structures. These are given as the columns of Φ. The eigenvalues are converted into exponential form, and the linear combination of dynamic modes $\boldsymbol{b}$ that match the initial conditions of X are found. These are used to reconstruct X only using the dominant dynamic modes, given by X_{DMD}.

X_{DMD} immediately gives the background of the video and the foreground is found by subtracting this from the original data. For the both videos, a uniform value was added to each pixel in the foreground video frames in order to see the results more clearly.

4 Computational Results

After performing the SVD, the squared singular values are examined to determine what percentage of the total variance each dynamic mode captures. These are shown in Figure 1 and Figure 2. Each of these is extremely skewed toward a high energy first mode, so the modes are plotted on a log scale. For both videos, over 99.9% of the variance is captured by the first mode. This should correspond to the background. As such, only $r = 1$ is needed to separate out the background and foreground.

The result of the DMD algorithm is X_{DMD}, which when plotted is seen to be the background. This is subtracted from the original video to yield the foreground. The original video, background, and foreground are shown for both videos, for three different time clips. This is shown for the ski video in Figure 3 and for the car video in Figure 4. By visual inspection, the procedure is clearly successful. However, the method is not quite perfect. Some geography can be seen in the foreground of the ski video, and some features of the racetrack, including the "UBS" sign, can be seen in the racecar track. However, DMD still presents an extremely powerful method for foreground-background separation processes, given the small amount of data needed and its ease of algorithmic implementation. This is expected to extend to other foreground-background separation processes beyond video.

5 Summary and Conclusions

Dynamic mode decomposition is shown to be an excellent tool for background/foreground video separation. Since DMD decomposes the dynamic data into spatial structures with coherent periodic behavior, the background is easily determined by selecting the largest dynamic mode, with no periodic behavior. The background video is then simply the reconstruction of the video data using only the dynamic modes corresponding to the background. The foreground is shown to be the original video minus this newly constructed background.

The power of this method is seen by considering that the exact same procedure was applied to both videos. This shows the separation is independent of the foreground activity, at least insofar as these examples show. Further research might investigate if each car in the race car video could be separated out individually.

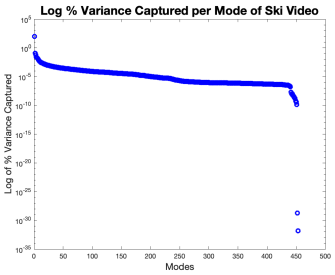

Figure 1: Percent of total variance captured by each mode for the ski video. Shown on logarithmic scale due to exponential difference between modes.

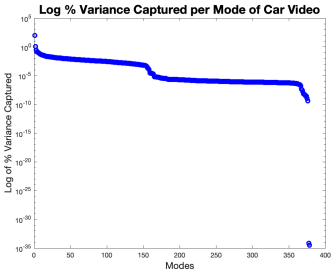

Figure 2: Percent of total variance captured by each mode for the car video.

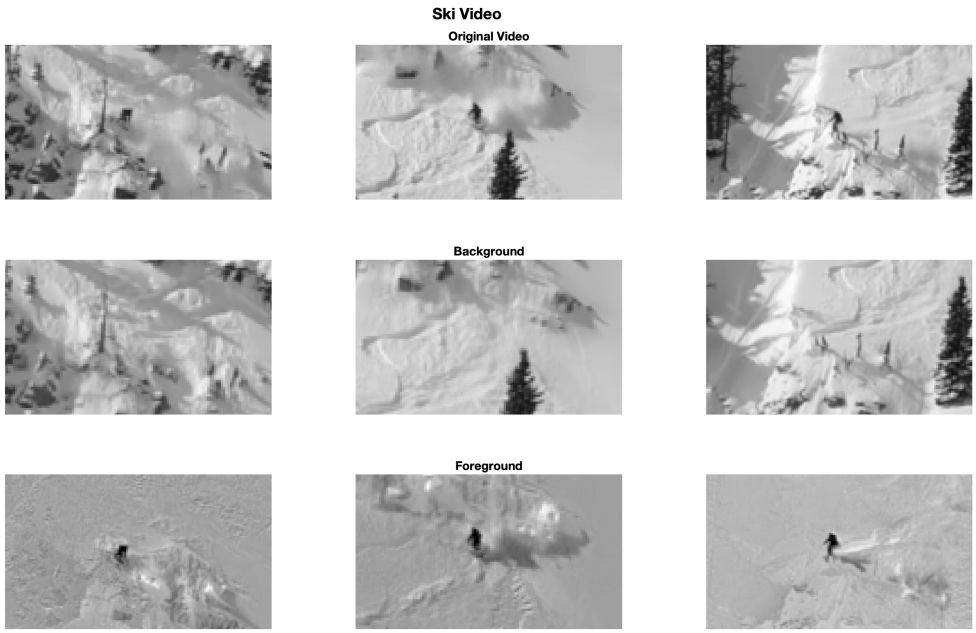

Figure 3: The original video, background, and foreground for three different time frames of the ski video.

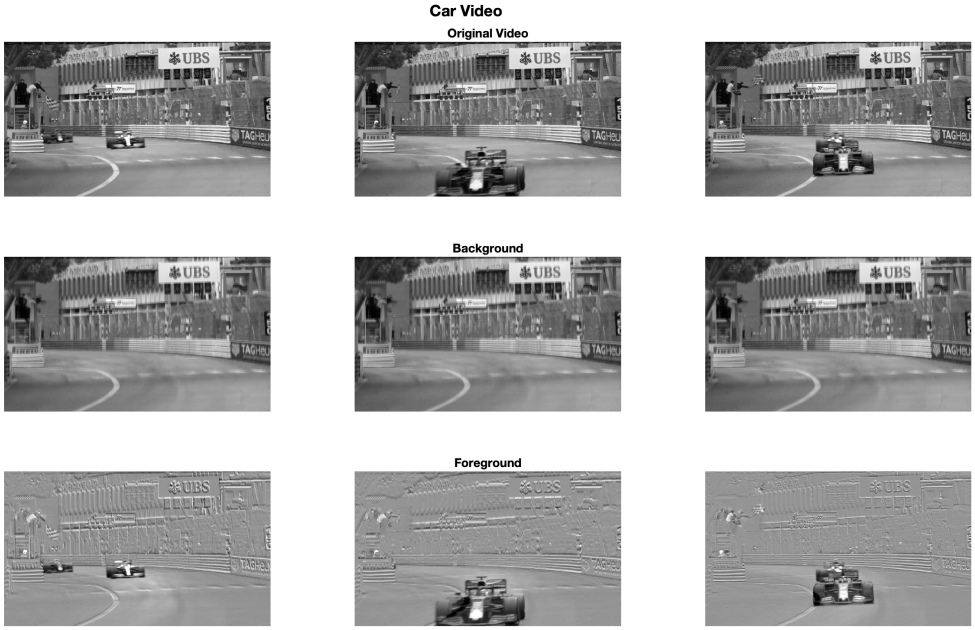

Figure 4: The original video, background, and foreground for three different time frames of the car video.

References

[1] *Dynamic Mode Decomposition (Overview)*. URL: `https://www.youtube.com/watch?v=sQvrK8AGCAo`.

[2] Nathan Kutz et al. *Dynamic Mode Decomposition*. 2016.

Appendix A MATLAB Functions

- `[U, S, V] = svd(X)`: Takes the singular value decomposition of the matrix X.
- `[W,D] = eig(X)`: Takes the eigendecomposition of the matrix X .

Appendix B MATLAB Code

See next page. Github: https://github.com/benfrancis314/AMATH582

```
X = vid_mat;
x1 = X(:,1:end-1);
x2 = X(:,2:end);
r=1;
dt=duration/numFrames;
[Phi,omega,lambda,b,X_dmd,S] = dmd(x1,x2,r,dt);

original = video;
background = uint8(X_dmd);
foreground = uint8(X(:,1:numFrames-1) - X_dmd + 150);

for i=1:numFrames-1
    foreground_reshaped(:,:,:,i) = reshape(foreground(:,i), 540, 960);
    background_reshaped(:,:,:,i) = reshape(background(:,i), 540, 960);
end

sig = diag(S).^2/sum(diag(S).^2)*100;
figure(1);
semilogy(sig,'bo','Linewidth',2)
title('Log % Variance Captured per Mode of Car Video', 'FontSize', 18);
ylabel('Log of % Variance Captured', 'FontSize', 14)
xlabel('Modes', 'FontSize', 14)

figure(2);
subplot(3,3,1), imagesc(rgb2gray(original(:,:,:,1))), colormap(gray), axis off
subplot(3,3,2), imagesc(rgb2gray(original(:,:,:,100))), colormap(gray), axis off, ...
    title('Original Video', 'FontSize', 14);
subplot(3,3,3), imagesc(rgb2gray(original(:,:,:,200))), colormap(gray), axis off
subplot(3,3,4), imagesc(background_reshaped(:,:,:,1)), colormap(gray), axis off
subplot(3,3,5), imagesc(background_reshaped(:,:,:,100)), colormap(gray), axis off, ...
    title('Background', 'FontSize', 14);
subplot(3,3,6), imagesc(background_reshaped(:,:,:,200)), colormap(gray), axis off
subplot(3,3,7), imagesc(foreground_reshaped(:,:,:,1)), colormap(gray),axis off
subplot(3,3,8), imagesc(foreground_reshaped(:,:,:,100)), colormap(gray),axis off, ...
    title('Foreground', 'FontSize', 14);
subplot(3,3,9), imagesc(foreground_reshaped(:,:,:,200)), colormap(gray),axis off
sgtitle('Car Video', 'FontSize', 18, 'FontWeight', 'bold')
```

Listing 1: Main program to run DMD and display background and foreground videos.

```
function [Phi,omega,lambda,b,X_dmd,S] = my_dmd(X1,X2,r,dt)
[U, S, V] = svd(X1, 'econ');
U_r = U(:, 1:r);
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);
A_tilde = U_r'*X2*V_r/S_r;
[W_r, D] = eig(A_tilde);
Phi = X2*V_r/S_r*W_r;
lambda = diag(D);
omega = log(lambda)/dt;
x1 =X1(:,1);
b = Phi\x1;
measurements_1 = size(X1, 2);
time_dynamics = zeros(r, measurements_1);
t = (0:measurements_1 - 1)*dt;
for i = 1:measurements_1
    time_dynamics(:, i) = (b.*exp(omega*t(i)));
end
X_dmd = Phi*time_dynamics;
```

Listing 2: Code for DMD algorithm.

```
% Change from _car to _ski and v.v. if switch vids
% vidObj = VideoReader('ski_drop_low.mp4');
vidObj = VideoReader('monte_carlo_low.mp4');

video = read(vidObj);
numFrames = get(vidObj,'NumFrames');
duration = get(vidObj,'Duration');

vid_mat = [];
for j=1:numFrames
    vid_mat(:,j) = double(reshape(rgb2gray(video(:,:,:,j)), [], 1));
end
```

Listing 3: Code to load in videos.