Dynamic Mode Decomposition of Video Data

Ashley Batchelor

Abstract

I used Dynamic Mode Decomposition (DMD) to analyze three videos, and to separate those videos into foreground and background frame data.

Sec. I. Introduction and Overview

I selected three videos to explore Dynamic Mode Decomposition and to visualize the foreground and background frame data and the combination of the two. The first video was a cat by a gate. The second video was monarch butterflies with a sky background. The third video was monarch butterflies on a forest floor. Each video was 1980x1080 resolution. The first video was 30 fps. The second and third videos were 60 fps.

Sec. II. Theoretical Background

A time varying dynamical system may be modeled as an n x m matrix $\boldsymbol{X}$ which contains columns constructed from $\boldsymbol{n}$ dimensional vectors $\boldsymbol{x_m}$ which are states of the system separated by a time increment $\Delta\boldsymbol{t}$.[1]

$$\boldsymbol{X} = [\boldsymbol{x_1}\ \boldsymbol{x_2} \dots \boldsymbol{x_m}] \tag{1}$$

Through Dynamic Mode Decomposition, this matrix $\boldsymbol{X}$ may be separated into two parts:

$$\boldsymbol{X}_{DMD} = b_p \phi_p e^{\omega_p \boldsymbol{t}} + \sum_{j \neq p} \phi_j e^{\omega_j \boldsymbol{t}} \tag{2}$$

where the first term forms a low rank matrix $\boldsymbol{X}_{DMD}^{Low-Rank}$ and the second term forms a sparse matrix $\boldsymbol{X}_{DMD}^{Sparse}$, ϕ_p and ϕ_j are the projected dynamic modes, b_p and b_j are the initial amplitudes of each dynamic mode, and ω_p and ω_j are continuous time eigenvalues determined from the discrete time eigenvalues based on the time increment $\Delta\boldsymbol{t}$. For a video, each frame may be reshaped as vectors $\boldsymbol{x_m}$ and the Dynamic Mode Decomposition may be understood as separating the frames into the low rank matrix which contains background frame data, and the sparse matrix which contains foreground data. Since the original matrix $\boldsymbol{X}$ is equal to the sum of these:

$$\boldsymbol{X} = \boldsymbol{X}_{DMD}^{Low-Rank} + \boldsymbol{X}_{DMD}^{Sparse} \tag{3}$$

we may empirically determine the low rank matrix $\boldsymbol{X}_{DMD}^{Low-Rank}$ using a DMD algorithm from the matrix $\boldsymbol{X}$.

$$\boldsymbol{X}_{DMD}^{Low-Rank} = b_p \phi_p e^{\omega_p \boldsymbol{t}} \tag{4}$$

We may then use the assumption of equation 3 to determine the sparse matrix $\boldsymbol{X}_{DMD}^{Sparse}$.

[1] Kutz, J. Nathan. Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press, 2018

$$\boldsymbol{X}_{DMD}^{Sparse} = \boldsymbol{X} - \left|\boldsymbol{X}_{DMD}^{Low-Rank}\right| \quad (5)$$

This difference may result in some negative values in the sparse matrix $\boldsymbol{X}_{DMD}^{Sparse}$. This may be mitigated with a residual n x m matrix $\boldsymbol{R}$ which contains all of the negative values of $\boldsymbol{X}_{DMD}^{Sparse}$ and zeros in the place of the non-negative values. The residual matrix $\boldsymbol{R}$ may be added to the modulus of the low rank matrix to yield a new low rank matrix:

$$\boldsymbol{X}_{DMD}^{Low-Rank} \rightarrow \left|\boldsymbol{X}_{DMD}^{Low-Rank}\right| + \boldsymbol{R} \quad (6)$$

The residual matrix $\boldsymbol{R}$ may be subtracted from the sparse matrix to yield a new sparse matrix.

$$\boldsymbol{X}_{DMD}^{Sparse} \rightarrow \boldsymbol{X}_{DMD}^{Sparse} - \boldsymbol{R} \quad (7)$$

A total DMD reconstructed matrix $\boldsymbol{X}_{DMD}$ may be computed as the sum of these new low rank and sparse matrices.

$$\boldsymbol{X}_{DMD} = \boldsymbol{X}_{DMD}^{Low-Rank} + \boldsymbol{X}_{DMD}^{Sparse} \quad (8)$$

Sec. III. Algorithm Implementation and Development

For each video, I used the MATLAB VideoReader to create a video object that I then read to an array. I resized each frame to 480x270 pixels and converted the color to grayscale to provide the matrix $\boldsymbol{X}$.

For the DMD portion, I first defined matrices $\boldsymbol{X_1}$ and $\boldsymbol{X_2}$, where $\boldsymbol{X_1}$ includes the columns 1 through m-1 of the matrix $\boldsymbol{X}$, and $\boldsymbol{X_2}$ includes the columns 2 through m. I then did SVD on the matrix $\boldsymbol{X_1}$ to determine matrices $\boldsymbol{U}$, $\boldsymbol{S}$, and $\boldsymbol{V}$. I then defined rank 2 truncated versions of these matrices $\boldsymbol{U_r}$, $\boldsymbol{S_r}$, and $\boldsymbol{V_r}$. I then determined a matrix $\widetilde{\boldsymbol{A}}$, where:

$$\widetilde{\boldsymbol{A}} = \boldsymbol{U_r^* X_2 V_r S_r^{-1}} \quad (9)$$

From the matrix $\widetilde{\boldsymbol{A}}$, I determined a set of eigenvalues λ_p and eigenvectors $\boldsymbol{\Omega_r}$. I determined a matrix $\boldsymbol{\Phi}$ of the DMD modes ϕ_p, where:

$$\boldsymbol{\Phi} = \boldsymbol{X_2 V_r S_r^{-1} \Omega_r^{-1}} \quad (10)$$

I calculated a set of values ω_p, where:

$$\omega_p = ln\lambda_p/\Delta t \quad (11)$$

I calculated the DMD mode amplitudes b_p where:

$$b_p = \boldsymbol{\Phi}^{\dagger}\mathbf{x_1} \quad (12)$$

and $\boldsymbol{x_1}$ is the first column of $\boldsymbol{X_1}$, i.e. the initial state of the system $\boldsymbol{X}$, using the pseudoinverse $\boldsymbol{\Phi}^{\dagger}$. I then used equations 4 and 5 and the adjustments with the residual matrix R described in the theoretical background, and took the real values of each to calculate f the low rank matrix $\boldsymbol{X}_{DMD}^{Low-Rank}$ and the sparse matrix $\boldsymbol{X}_{DMD}^{Sparse}$.

I then used the 50th frame of each as a sample to show the foreground, background, and the combined DMD and compared it with the original 50th frame.

Sec IV. Computational Results

Figures 2 a) and 2 b) show the foreground and background of the cat video. The background shows a significant amount of ghosting of the cat, but a recognizable image of the yard and walkway. The combined image shown in Figure 2 c) shows a very clear scene which is visually indistinguishable from the original image show in Figure 2 d). The first two modes of the principal components used to model the low rank dynamics represented 37.9 % of the total energy.

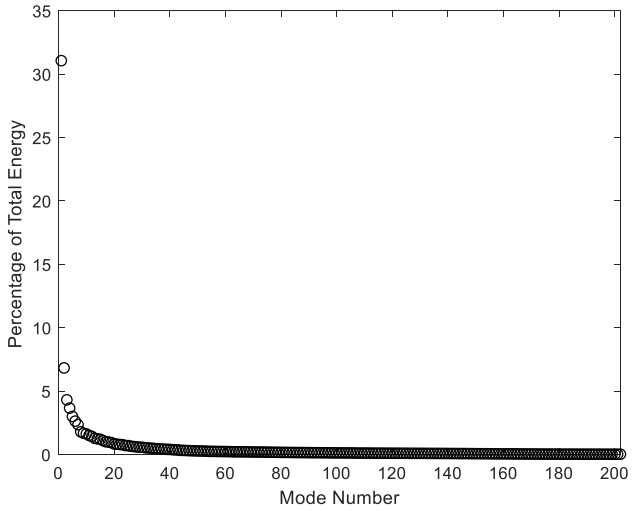
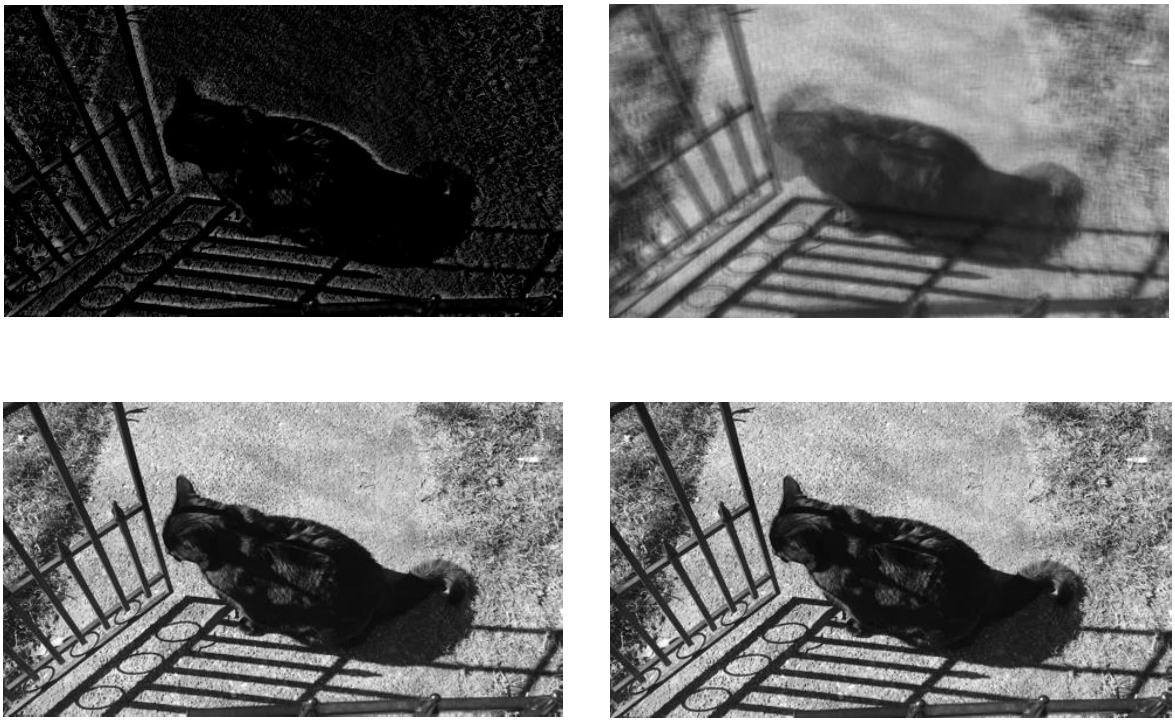


Figure 1 – Energy of the principal modes.



Figures 2 a), b), c), d) left to right, to bottom – Sparse foreground frame data, low rank foreground data, combined foreground and background, original frame.

Figure 4 a) and 4 b) show the foreground and background of the butterflies on the forest floor video. The foreground shows very distinct butterflies. The background does not show a significant amount of ghosting of the butterflies. The combined image shown in Figure 4 c) shows a very clear scene which is visually indistinguishable from the original image show in Figure 4 d). The first two modes of the principal components used to model the low rank dynamics represented 40.4% of the total energy.

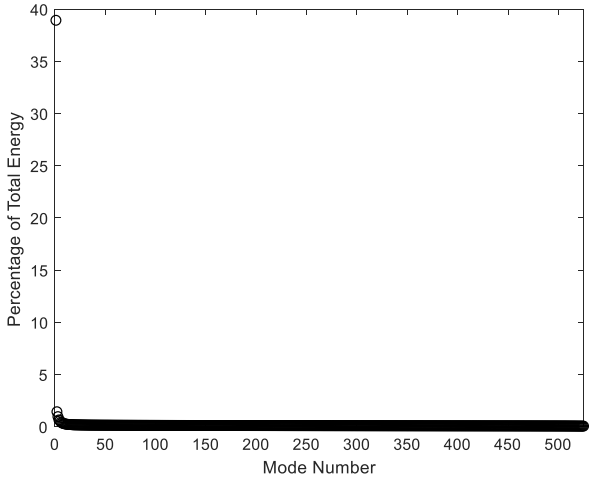
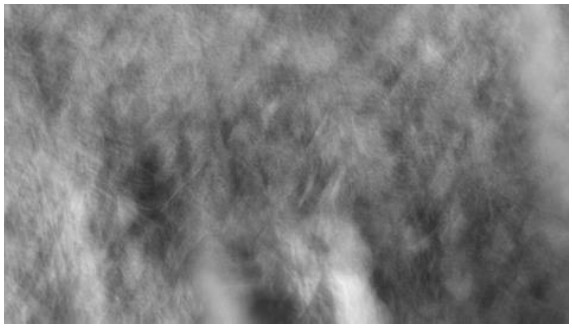


Figure 3 - Energy of the principal modes.



Figures 4 a), b), c), d) left to right, to bottom – Sparse foreground frame data, low rank foreground data, combined foreground and background, original frame.

Figure 5 a) and 5 b) show the foreground and background of the butterflies on the forest floor video. The foreground is very dark, although some faint aspects of clouds are visible. The butterflies are not distinguishable in the foreground, most likely because of their dark color against a bright sky. The background does not show any discernible ghosting of the butterflies. The combined image shown in Figure 5 c) shows a very clear scene which is visually indistinguishable from the original image show in Figure 5 d). The first two modes of the principal components used to model the low rank dynamics represented 17.4% of the total energy.

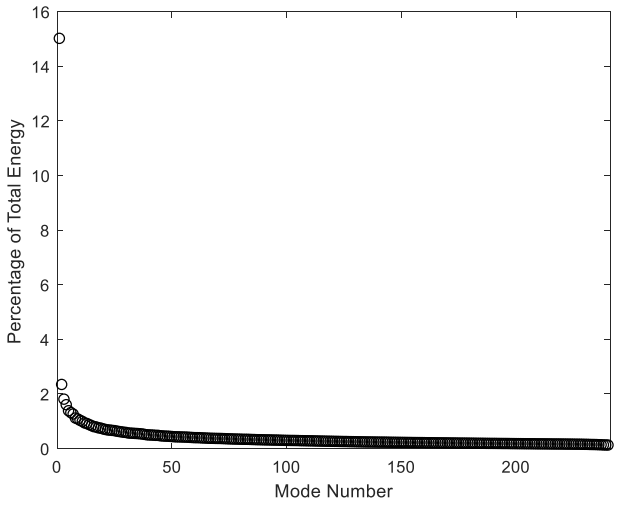


Figure 4 - Energy of the principal modes.

Figures 5 a), b), c), d) left to right, to bottom – Sparse foreground frame data, low rank foreground data, combined foreground and background, original frame.

Sec. V. Summary and Conclusions

I successfully separated the videos into foreground and background data. The cat video showed some ghosting of the cat in the background. The butterflies on the forest floor video clearly showed the butterflies in the foreground data. The background data looked quite blurry compared to the backgrounds of the other videos. The butterflies in the sky foreground data appeared all black, but did in fact contain the information for the butterflies which were easily visible after combining the foreground and background. For each video, the sum of the foreground and background looked just like the original frame. The algorithm for handling the residuals and converting to real numbers allowed me to successfully reconstruct the frames by summing the foreground and background.

Appendix A MATLAB functions used and brief implementation explanations

imresize() – This resizes an image to a specified size.

read() – This reads a video object into an array.

reshape() – This reshapes an array, e.g. a vector to a matrix or vice versa.

svd() – This calculates the three matrices of SVD based on the input matrix.

VideoReader() – This reads a video file into a MATLAB object.

Appendix B MATLAB codes

```
clear all; close all; clc;

%Read in the video data
obj=VideoReader('Butterflies2.mp4');
vidFrames = read(obj);
numFrames = get(obj,'numberOfFrames');
bwVidFrames=[];
%downsize resolution and turn to grayscale
for k = 1 : numFrames
currentFrame=uint8(rgb2gray(vidFrames(:,:,:,k)));
bwVidFrames(:,:,k)=uint8(currentFrame);

end


X=[];
for j=1:numFrames
    %convert current frame to a vector and store it as the data X
    xCurrent=reshape(bwVidFrames(:,:,j),[1,numel(bwVidFrames(:,:,j))]);
    X=[X;xCurrent];
end
%Transpose X
X=X';


numFrames=203;
frameRate = 29.977;
t=linspace(0,(numFrames/frameRate),numFrames);
dt=t(2)-t(1);
X1 = X(:,1:end-1); X2 = X(:,2:end);
r=2; %Select rank 2

[U, S, V] = svd(X1, 'econ');
r = min(r, size(U,2));
U_r = U(:, 1:r); % truncate to rank-r
S_r = S(1:r, 1:r);
V_r = V(:, 1:r);
Atilde = U_r' * X2 * V_r / S_r; % low-rank dynamics
[W_r, D] = eig(Atilde);
Phi = X2 * V_r / S_r * W_r; % DMD modes
lambda = diag(D); % discrete-time eigenvalues
omega = log(lambda)/dt; % continuous-time eigenvalues
```

```
%% Compute DMD mode amplitudes b
x1 = X1(:, 1);
b = Phi\x1;

% DMD reconstruction
time_dynamics = zeros(r, length(t));
for iter = 1:length(t),
time_dynamics(:,iter)=(b.*exp(omega*t(iter)));
end;

X_dmd_background = Phi*time_dynamics;
X_dmd_foreground = X - abs(X_dmd_background);
%Determine matrix R of residual negative values
[m,n]=size(X_dmd_background);
R=zeros(m,n);
for i=1:m
    for j=1:n
        if X_dmd_background(m,n) > 0
            R(m,n)= X_dmd_background(m,n);
        end
    end
end

%Adjust the foreground and background with the residuals.
X_dmd_background = R + abs(X_dmd_background);
X_dmd_foreground = X_dmd_foreground - R;

X_dmd_background_real = real(X_dmd_background);
X_dmd_foreground_real = real(X_dmd_foreground);

%Reconstruct sample foreground image
testimage = X_dmd_foreground_real(:,50);
testimage1 = reshape(testimage,[270,480]);
testimage1 = uint8(testimage1);

%Reconstruct sample background image
testimage = X_dmd_background_real (:,50);
testimage2 = reshape(testimage,[270,480]);
testimage2 = uint8(testimage2);

%Reconstruct sample combined image
X_dmd = X_dmd_background_real + X_dmd_foreground_real;
testimage = X_dmd(:,50);
testimage3 = reshape(testimage,[270,480]);
testimage3 = uint8(testimage3);

%Reconstruct sample original image
testimage = X(:,50);
testimage4 = reshape(testimage,[270,480]);
testimage4 = uint8(testimage1);

%Show each image
imshow(testimage1);
imshow(testimage2);
imshow(testimage3);
imshow(testimage4);
```