AMATH 482 Assignment 5

Carter Peyton

March 17, 2021

Abstract

This assignment aims to use the dynamic mode decomposition in order to separate foreground and background videos. It does so through the separation of sparse and low-rank approximation matrices in the DMD solution, where the sparse matrices correspond to the foreground of a video and the low-rank matrices correspond to the background of the video.

1 Introduction and Overview

Given two videos, one of a skier going down a mountain and the second of Formula 1 cars crossing the finish line, separate the foreground and background videos for each clip. Note that the frames (i.e the camera angle/position) are still. This corresponds to the only movement in each clip being the skier and cars respectively. The background of these videos does not change because the camera does not move. Therefore, the DMD will allow us to separate each video into foreground and background i.e we will extract the skier and cars from each video clip.

2 Theoretical Background

2.1 The Dynamic Mode Decomposition

The dynamic mode composition (DMD) is a decomposition technique that finds the basis of spatial modes for which the time dynamics become can be represented by exponential functions. The exponential nature means that only three behaviors can be observed: growth, decay, and oscillation. These behaviors are governed by the eigenvalues where positive values result in growth, negative values result in decay, and imaginary values result in oscillations. Note: in order for DMD to work properly the temporal data needs to be captured in consistent intervals. In order to compute the DMD we consider a data matrix, X, with M time snapshots and alternative matrix, X_1^{M-1}

$$X = |\vec{X}_1, \vec{X}_2, ..., \vec{X}_M| \tag{1}$$

$$X_1^{M-1} = |\vec{X}_1, \vec{X}_2, ..., \vec{X}_{M-1}| \tag{2}$$

We then apply a Koopman matrix A in additional to a vector $e_{M-1} = |0, 0, ..., 1|$. This Koopman matrix, A is what evolves our data in time as it has the property $x_{m+1} = Ax_m$

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

We can take the SVD of X_1^{M-1} in order to solve for A. Note: we can use the first r nonzero singular values and corresponding columns to reduce the dimensions of our data.

$$X_2^M = AU\Sigma V + re_{M-1}^T \tag{4}$$

Moreover, we want to choose A such that the columns of X_2^M are linear combinations of the columns of U. This is equivalent to them being written as linear combinations of the POD modes, therefore r has to be orthogonal to U (the POD basis). This results in

$$U^*AU = U^*X_2^M V\Sigma^{-1} \tag{5}$$

Where we refer to the right hand side as $\tilde{S}$, which has the same eigenvalues as A because they are similar matrices. If we write the standard eigenvalue/vector equation for $\tilde{S}$ as

$$\tilde{S}y_k = \mu_k y_k \tag{6}$$

We know that the corresponding eigenvectors of A (our DMD modes) will be in the form

$$\psi_k = Uy_k \tag{7}$$

Where U is an eigenvector of A. Moreover, we can describe the continual multiplications of A (which iterates us forward in time) by performing an eigenbasis expansion so that

$$x_{DMD}(t) = \sum_{k=1}^{r} b_k \psi_k e^{\omega_k t} = \Phi diag(e^{\omega_k t)\vec{b}} \tag{8}$$

Where r is the number of modes we want to use (we determined this from the SVD) and $\omega_k = ln(\mu_k)/dt$. Lastly we can compute $\vec{b}$ (our initial conditions) by taking the fact that at $t = 0$ we get x_1. This is our first value, so we compute $\vec{b}$ by solving

$$\Phi\vec{b} = x_1 \tag{9}$$

$$\vec{b} = \Phi^\dagger x_1 \tag{10}$$

2.2 Using DMD to Seperate Foreground and Background Videos

The DMD spectrum of frequencies can be used to subtract background modes from videos. This is accomplished by noticing that

$$X_{DMD} = b_p\phi_p e^{\omega_p t} + \sum_{j!=p} b_j\phi_j e^{\omega_j t} \tag{11}$$

where we assume that $\omega_p \in 1, 2, ..., l$ and satisfies $||\omega_p|| \approx 0$. Moreover, $||\omega_j||\forall j \neq p$ meaning that it is bounded away from 0. This means that the first term serves as a low-rank approximation of our DMD solution to X and the second term serves as a sparse reconstruction of the DMD solution. However, we need some method of accounting for the complex values of the DMD solution, so we construct a residual matrix, R, of the negative values in X_{DMD}^{Sparse}. Then our adjusted (acconting for complex values) background and foreground matrices are

$$X_{DMD}^{low} = R + |X_{DMD}^{low}| \tag{12}$$

$$X_{DMD}^{Sparse} = X_{DMD}^{Sparse} - R \tag{13}$$

Together these satisfy

$$X = X_{DMD}^{low} + X_{DMD}^{Sparse} \tag{14}$$

3 Algorithm Implementation and Development

3.1 Initialization and SVD

Each video clip is loaded in to MATLAB and reshaped into a M*N by time matrix. Then the X1 and X2 matrices are constructed via $X1 = X(:, 1 : end - 1)$ and $X2 = X(:, 2 : end)$. The SVD of the X1 matrix is taken, and the resulting spectrum is plotted.

3.2 DMD Initialization

Once the proper number of modes has been determined (based on the SVD spectrum), we reduce the size of U, Σ, and V to these modes (i.e the first r values of Σ and r columns of U and V). Then we construct $\tilde{S}$ and initialize our timestep $dt = \frac{1}{framerate}$. We then take the eigenvalues of $\tilde{S}$ and use them to compute Φ and ω.

3.3 DMD Reconstruction

We use the backslash command (pseudo-inverse) to solve for our initial conditions in $\vec{b}$ using the first column of X1. These initial conditions are used to created a dynamics matrix of exponentials by iterating over our time vector. We obtain our low-rank approximation of the DMD solution by multiplying Φ against this dynamics matrix.

3.4 Separating foreground and background videos

Now that we have our low-rank approximation of the DMD solution, we can subtract the absolute value of it from our original data to obtain the sparse DMD matrix. This new matrix corresponds to the foreground video. However, this sparse matrix may contain negative values, so we create a residual matrix, R, of these negative values. We add this residual matrix with the absolute value of our low-rank approximation to get a new approximation that accounts for the magnitudes of the complex values in the DMD reconstruction.

$$X_{DMD}^{low} = R + |X_{DMD}^{low}| \tag{15}$$

Moreover, we have to subtract the residual matrix from our sparse matrix because negative values are uninterpretable in images, the final sparse matrix contains only positive values.

$$X_{DMD}^{sparse} = X_{DMD}^{sparse} - R \tag{16}$$

3.5 Reshaping and Displaying

We reshape each matrix (foreground/sparse, background/low-rank, and DMD solution) back into 3D matrices of dimensions N by M by time, so that they can be played as videos. Various frames of these videos are displayed in the computational results section.

4 Computational Results

4.1 SVD Spectrum

4.1.1 Monte Carlo

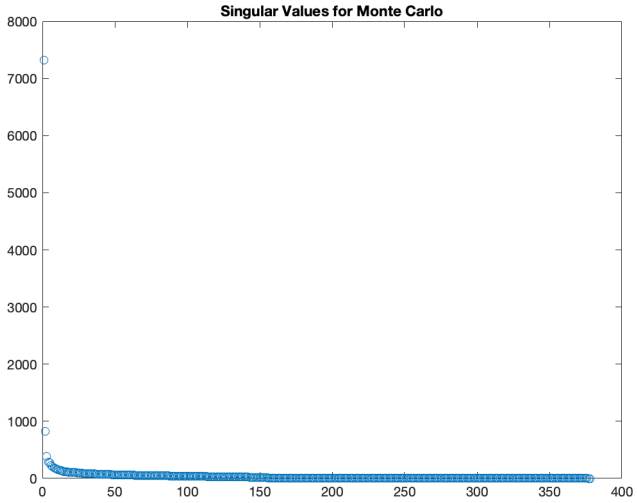

Figure 1: SVD spectrum for X1 in monte carlo.

There seem to be two dominant singular values in monte carlo, both on the order of 10^3 whereas there is a drop off to many values in the range of 10^2. Therefore we will let $r = 2$.

4.1.2 Ski Drop

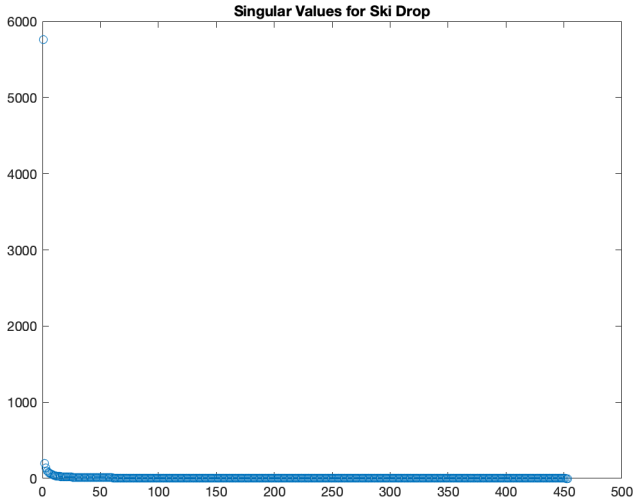

Figure 2: SVD spectrum for X1 in ski drop.

There seems to be one dominant mode in the ski drop video, therefore we will let $r = 1$ for this clip.

4.2 Frame Comparison between Initial, Background, and Foreground Isolations

4.2.1 Monte Carlo Results

Figure 3: 50th frame for the original, background (low rank), and foreground (sparse) videos of the monte carlo clip.

We see that the DMD does a good job of separating out the foreground video as the only things that we can see are the Formula 1 cars (the moving aspect of the video). However, the DMD is less successful in isolating the background, and we see these "ghost cars" that are partially removed from the image, but not entirely.

4.2.2 Ski Drop Results

Figure 4: The original and isolated background videos. The background video is hard to distinguish from the original, however, the skier is less prominent (both are at frame 50 in the clip).

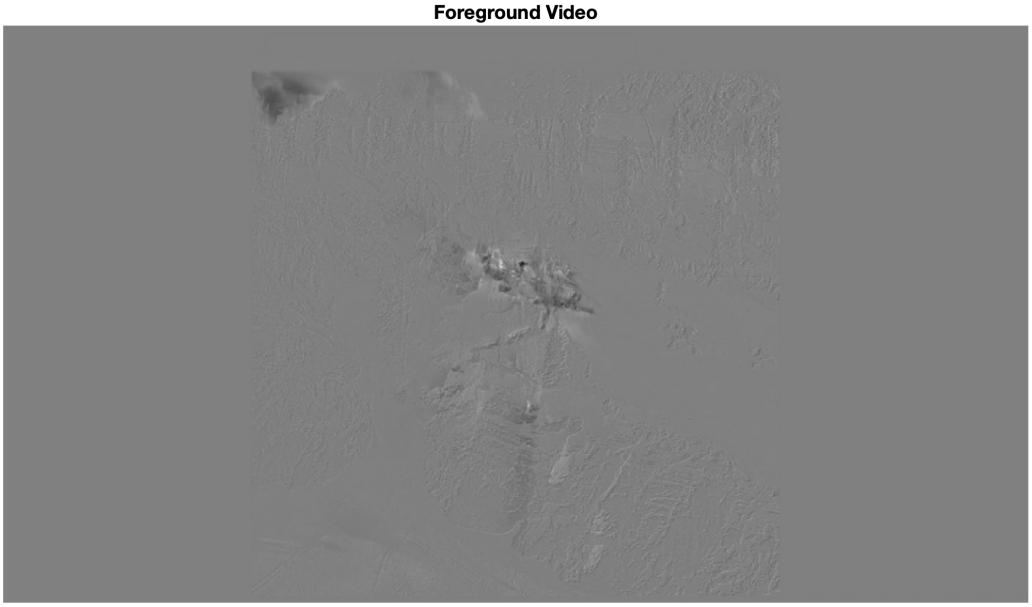

Figure 5: Foreground video of the skier (the black figure in the center) at frame 50.

Due to the dark nature of the skier's clothing in the ski drop video, some uniform grey was added to the sparse matrix in order to provide better contrast. Similar to the background video in the monte carlo clip, there is some "ghost skier" in the frame. However, the foreground video does a good job of isolating just the skier (and snow that he kicks up during his turn).

5 Summary and Conclusion

Through the use of the DMD and SVD on two different, stable video clips we were able to isolate the moving objects into a foreground video and scenery into a static background video. The DMD performed well in separating the foreground image for monte carlo. However, the skier's dark clothing in the ski drop clip required some uniform grey in order to improve contrast in that foreground video. The background videos were less successful than the foreground videos. However, using the DMD to isolate moving objects in videos is an empirically sound method.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `v = VideoReader(filename)` creates object v to read video data from the file named filename.
- `I2 = im2double(I)` converts the image I to double precision. I can be a grayscale intensity image, a truecolor image, or a binary image. im2double rescales the output from integer data types to the range [0, 1].
- `I = rgb2gray(RGB)` converts the truecolor image RGB to the grayscale image I.
- `B = reshape(A,sz)` reshapes A using the size vector, sz, to define size(B). For example, reshape(A,[2,3]) reshapes A into a 2-by-3 matrix. sz must contain at least 2 elements, and prod(sz) must be the same as numel(A).
- `[U,S,V] = svd(A,'econ')` produces an economy-size decomposition of m-by-n matrix A:

- `[V,D] = eig(A)` returns diagonal matrix D of eigenvalues and matrix V whose columns are the corresponding right eigenvectors, so that A*V = V*D.
- `imshow(I)` displays the grayscale image I in a figure. imshow uses the default display range for the image data type and optimizes figure, axes, and image object properties for image display.

Appendix B MATLAB Code

```
%% Init for videos
clear all; clc

monte = VideoReader('monte_carlo_low.mp4')
ski = VideoReader('ski_drop_low.mp4')
N = 540;
M = 960;
count = 1;
while hasFrame(monte)
    monteFrame = im2double(rgb2gray(readFrame(monte)));
    monte_dat(:,count) = reshape(monteFrame,[M*N 1]);
    count = count + 1;
end
count = 1;
while hasFrame(ski)
    skiFrame = im2double(rgb2gray(readFrame(ski)));
    ski_dat(:,count) = reshape(skiFrame,[M*N 1]);
    count = count + 1;
end
monte_length = size(monte_dat,2);
ski_length = size(ski_dat,2);
% each column of monte_dat and ski_dat is a time evolution

%% Set up X matrices and take SVD
% chan change monte_dat for ski_dat to switch between videos
X1 = monte_dat(:,1:end-1);
X2 = monte_dat(:,2:end);
[U, Sigma, V] = svd(X1,'econ');

%% Plot singular values and determine necessary modes
sing_vals = diag(Sigma);
plot(linspace(1,length(sing_vals),length(sing_vals)),sing_vals,'o')
title('Singular Values for Monte Carlo')
print('-dpng','svd.png')

% only use the significant modes
modes = 2;
U_trunc = U(:,1:modes);
Sigma_trunc = Sigma(1:modes,1:modes);
V_trunc = V(:,1:modes);

%% DMD Init
dt = 1/monte.FrameRate;

S_tilde = U_trunc'*X2*V_trunc/Sigma_trunc;
```

```
[eV, D] = eig(S_tilde);
Phi = X2*V_trunc/Sigma_trunc*eV;
mu = diag(D);
omega = log(mu)/dt;

%% DMD Reconstruction
b = Phi\X1(:,1);
iter_length = size(X1,2);
time_vect = 0:dt:iter_length-1;

dynamics = zeros(modes,iter_length);
for iter = 1:iter_length
    dynamics(:,iter) = b.*exp(omega*time_vect(iter));
end
X_dmd = Phi*dynamics;

%% Sparse and nonsparse construction
X_sparse = X1-abs(X_dmd); % computes real-valued sparse matrix (add 0.5 to this for ski drop contrast)
R = X_sparse.*(X_sparse<0); % create residual matrix of negative values

background = R + abs(X_dmd);
foreground = X_sparse - R;

X_approx = background + foreground;

%% Reshape into videos
dur = monte_length - 1; % will need to change to ski_length for other video
bg_vid = reshape(background, [N, M, dur]);
fg_vid = reshape(foreground, [N, M, dur]);
dmd_vid = reshape(X_approx, [N, M, dur]);

%% Play videos
for i = 1:dur
    imshow(fg_vid(:,:,i))
end

%% Capture images
subplot(3,1,1)
imshow(reshape(monte_dat(:,50),[N M]))
title('Original Video','Fontsize',18)
subplot(3,1,2)
imshow(bg_vid(:,:,50))
title('Background Video','Fontsize',18)
subplot(3,1,3)
imshow(fg_vid(:,:,50))
title('Foreground Video','Fontsize',18)
print('-dpng','monte_comparison1.png')
```

Code from homework5.m