AMATH 482 Homework 5

Sophia Jannetty

March 17, 2021

Abstract

Here I use Dynamic Mode Decomposition (DMD) to separate the foreground and the background of a video of the Monte Carlo car race and a video of a Skier. I use two methods. For one I performed DMD on all Proper Orthogonal Decomposition (POD) modes of my dataset and then selected the DMD modes associated with the lowest magnitude continuous-time DMD eigenvalues to be the background of the video. For the second I performed DMD on a low number of SVD modes of my dataset and selected all resulting DMD modes to be the background of the video. Background frames were reconstructed from selected DMD modes and subtracted from the input data matrix yielding the foreground video (which was then brightened such that there were no negative values). Upon visual inspection, the second method seemed to yield better results.

1 Introduction and Overview

Dynamic Mode Decomposition is a modal decomposition of a dataset in which each mode consists of spatially correlated structures that have the same linear behavior in time [1]. Unlike POD modes, DMD modes do not have to be orthogonal, giving them more flexibility to describe datasets. Additionally the amount of change seen in a mode over time (and the type of change, be it growth, decay, or oscillation) can be inferred from each mode's continuous time eigenvalue. These attributes make DMD modes a potentially good tool for separating the unchanging background video in our video samples from the changing foreground.

The DMD relies on first determining the POD modes of the dataset using the SVD. Once these modes are determined, the rest of the DMD can be performed using all POD modes or a subset of the first few. In this project I tried separating the foregrounds and backgrounds two ways. One, termed the Full Rank Data DMD Mode Selection method, used all POD modes to conduct the DMD and then selected a small number of DMD modes with low-magnitude continuous-time eigenvalues to be the background. The other, termed the Low Rank Data Approximation Subtraction method only used a few POD modes for the DMD, and then selected all resulting DMD modes to be the background.

2 Theoretical Background

2.1 Singular Value Decomposition (SVD)

Assume we have a large data set $X \in \mathbb{C}^{nxm}$ in which each column x_k contains pixel data from a distinct frame from a video. The SVD expresses this matrix as the product of three constituent matrices as shown in equation 1.

$$X = U\Sigma V^* \tag{1}$$

The economy SVD is a version of the SVD that minimizes the dimensions of these matrices to eliminate the zero entries in Σ (this reduced Σ is now called $\hat{\Sigma}$) and to eliminate the corresponding columns of U (this reduced U is now called $\hat{U}$. Equation 2 shows to reduced SVD [1].

$$X = \hat{U}\hat{\Sigma}V^* \tag{2}$$

The values σ_n along the diagonal of $\hat{\Sigma}$ are called the singular values of matrix A and are ordered from smallest to largest. The vectors u_n that make up the columns of $\hat{U}$ are called the left singular values of X and the vectors v_n that make up the columns of V are called the right singular values of X (note that the MATLAB implementation of the econ SVD returns matrices $\hat{U}$, $\hat{\Sigma}$, and V, NOT the complex conjugate transpose V^* in Equation 2. In this document I will describe V, not V^*).

2.1.1 Use in Proper Orthogonal Decomposition (POD)

POD finds a sequence of orthonormal basis functions for a discretized system.These basis functions are computed using the SVD. Upon running the SVD the basis functions are stored hierarchically in the columns of $\hat{U}$.

2.1.2 Use for Low-Rank Matrix Approximation

The SVD provides a hierarchy of optimal low-rank approximation of matrix X. Matrix X is the sum of r rank-one matrices calculated in Equation 3 [2].

$$X = \sum_{j=1}^{r} \sigma_j u_j v_j^* \tag{3}$$

The SVD can be used to see how many independent basis vectors are present in the dataset and will order them from most to least important. By setting r in Equation 3 to a value smaller than the number of columns of $\hat{U}$, you can create a low-rank approximation of your input Matrix. This allows for accurate and efficient matrix reduction. Performing the DMD on the a subset of the leading POD modes essentially performs the DMD on such a low-rank approximation of the data.

2.2 Dynamic Mode Decomposition (DMD)

2.2.1 Overview

The DMD provides a modal decomposition in which each mode consists of spatially correlated structures that have the same linear behavior in time. It allows for dimensionality reduction of data (by reconstructing the dataset using a select few set of modes) and provides a model for how each mode changes over time.

The DMD algorithm requires a number of pairs of snapshots of the state of a system as it evolves over time, taken at a constant sampling rate that is fast enough to capture the highest frequencies present in the system dynamics. The DMD seeks to find the eigenvalues and eigenvectors of Matrix A that relates two sequential snapshot matrices in time as shown in Equation 4 (assuming uniform sampling rate). As a preprocessing step, data must be organized such that each column of data matrix X contains data pertaining to one snapshot. Then the first n-1 columns are put in data matrix X_1 and columns 2 through n are put in data matrix X_2. The operator A is defined as shown in equation 5 where $||.||_F$ is the Frobenius norm and † means the pseudo-inverse.

$$x_{k+1} = Ax_k \tag{4}$$

$$A = \underset{A}{argmin}||X_2 - AX_1||_F = X_2{X_1}^\dagger \tag{5}$$

A can be computed directly but all we need in order to apply this matrix to our data are the matrix's eigenvalues and eigenvectors. Instead of computing A, we compute the projection of A onto the r singular vectors (or principal components or POD modes) found in the singular value decomposition of X, resulting in matrix $\tilde{A}$ of size rxr. The high-dimensional DMD modes can be calculated from $\tilde{A}$ and X.

2.2.2 Steps Required for DMD

As a preprocessing step, matrices X_1 and X_2 are constructed as described above.

Step one: Compute the singular value decomposition of data matrix X_1 (as described in section 2.1) to get the POD modes of the system. If desired, a low-rank approximation of X_1 can be used for all subsequent data matrices by determining the r number of ranks you would like to use and setting $\tilde{U} = \hat{U}[:, 1:r]$, setting $\tilde{\Sigma} = \hat{\Sigma}[1:r, 1:r]$, and setting $\tilde{V} = \mathrm{V}[:, 1:r]$. Otherwise r can be set such that all SVD ranks are included in all matrices.

Step two: Project A onto the POD modes in $\tilde{U}$ to find $\tilde{A}$ as seen in Equation 6.

$$\tilde{A} = \tilde{U}^* A \tilde{U} = \tilde{U}^* X_2 \tilde{V} \tilde{\Sigma}^{-1} \tag{6}$$

This equation is derived from equation 5. $\tilde{A}$ will have the same nonzero eigenvaues as A, even though $\tilde{A}$ is potentially a reduced-order matrix that defines a linear model for the dynamics of an r-rank approximation of the dataset X_1.

Step three: Compute the spectral decomposition of $\tilde{A}$ as shown in equation 7 such that you have the

DMD eigenvalues (along the diagonal of diagonal matrix Λ) and the eigenvectors of $\tilde{A}$ (along the columns of W). These eigenvectors can be thought of as linear combinations of POD mode amplitudes that behave linearly with a temporal pattern given by λ [1].

$$\tilde{A}W = W\Lambda \tag{7}$$

Step four: The high-dimensional DMD modes Φ are reconstructed using W and X_2 using equation 8.

$$\Phi = X_2\tilde{V}\tilde{\Sigma}^{-1}W \tag{8}$$

These DMD modes are eigenvectors of A corresponding to the eigenvalues in Λ. The eigenvalues give you information about how the DMD modes change over time. They are complex. A positive real part implies growth over time, a negative real part implies decay over time, and a large magnitude complex part implies oscillation over time. Low absolute value eigenvalues imply little change of a mode over time.

2.2.3 Use for Video Background/Foreground Separation

The assumption is made that the background of these videos changes very little over time [3]. Thus DMD modes with eigenvalues close to 0 are assumed to describe the background and all other modes are assumed to describe the foreground. Once the DMD modes are calculated, continuous time eigenvalues ω are calculated and omegas very close to 0 are considered to be associated with DMD modes that describe the background of the video. Those background modes are consolidated into one matrix and the background is reconstructed according to Equation 9.

$$X_{DMD}(t) = \sum_{k=1}^{k=num\_frames} b_k(0)\psi_k(x)e^{\omega t} \tag{9}$$

Where $b_k(0)$ is the initial condition of the background, and the ψs and ωs considered are only those associated with background DMD modes. Once the background has been reconstructed, the background matrix can be subtracted from the input data matrix X, leaving a foreground matrix. This foreground matrix will have many zeros and may have negative values. The assignment describes a means of identifying these negative values, setting them to zero in the foreground image, and subtracting them from the background image (thus adding the absolute values of the negative pixel values to the background matrix). However I did not have very much success with this method and opted instead to artificially brighten the foreground matrix by adding a constant pixel value to each pixel.

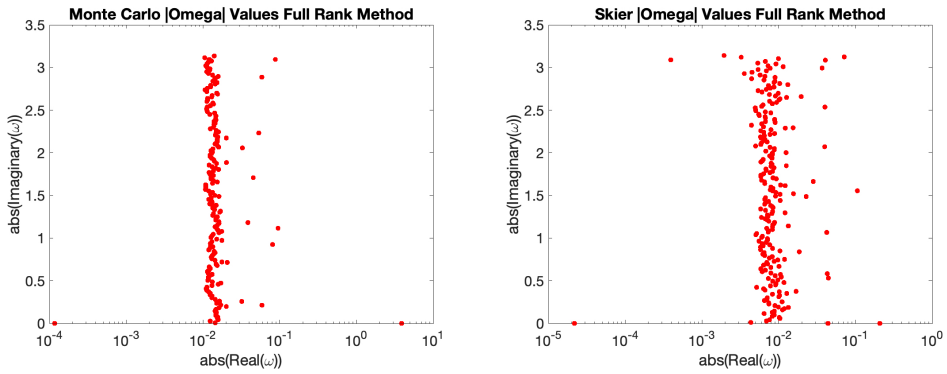
3 Algorithm Implementation and Development

Separation was performed using two techniques; Full Rank Data DMD mode selection and Low-Rank Data Approximation subtraction. The initial steps for each method were the same.

Each video was imported and converted to grayscale. A time vector was constructed counting from 1 to the number of frames in the video by time steps of 1 (so time was conflated with frames). Each frame of the video was converted to a column vector and stored in matrix X. Matrix X1 consisting of all columns of X except the last column was constructed along with matrix X2 which consisted of all columns of X except the first column. The SVD of X1 was found using MATLAB's svd function. From here the two methods diverged.

3.1 Full Rank Data DMD Mode Selection Method

In this method A was projected on all POD modes found in the svd above (as seen in Equation 6 when setting r = the number of frames in the video). Next the spectral decomposition of $\tilde{A}$ was found using MATLAB's eig function, resulting in the matrix of eigenvectors W and diagonal matrix Λ of eigenvalues. Then high-dimensional DMD modes Φ were found using Equation 8. Next the continuous time eigenvalues ω were found by dividing $\log(diag(\Lambda))$ by the change in time between snapshots (which here was set to 1 by considering time in terms of frame). The absolute values of omegas were plotted (see Figure 1, note X axis is on a log scale) and those that were considered to be appropriately close to 0 were considered to be eigenvalues of background modes. For the Monte Carlo video this was one mode and for the Skier video this



(a) Full Rank Monte Carlo Omega Magnitudes (b) Full Rank Skier Omega Magnitudes

Figure 1: Absolute values of real and imaginary parts of continuous time eigenvalues ω for the Monte Carlo (a) and Skier (b) full-rank data DMD mode selection analyses. One mode was selected to be the background for the Monte Carlo video and three modes were selected to be the background for the Skier video.

was 3 modes. These numbers were determined through experimentation and visual inspection. The ω values and DMD modes stored in the columns of Φ associated with these background modes were consolidated into their own omega_bg and Phi_bg matrices. The background video was then reconstructed using equation 9. The background initial condition was calculated by multiplying inv(Phi_bg) by the first frame in X (see code in Appendix B.1 lines 170-177). Each column of this background reconstructed matrix was reshaped to be the dimensions of a frame and viewed. The absolute value of this reconstructed background matrix was then subtracted from the input data matrix X to construct the sparse foreground matrix (the absolute value was taken to account for the fact that each term in the DMD reconstruction is complex). The minimal pixel value in the foreground matrix was identified and the absolute value of that pixel was added to all pixels in the matrix. Each column was reshaped back into frame dimensions and viewed.

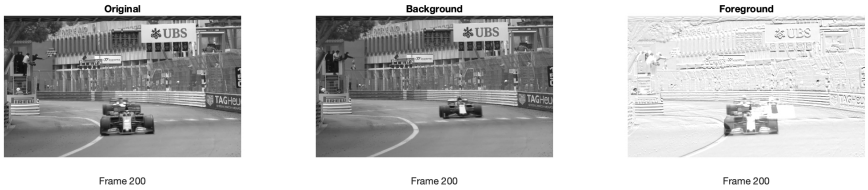
3.2 Low-Rank Data Approximation Subtraction Method

Following the SVD, A was projected onto a very small number of POD modes; two for the Monte Carlo video and 8 for the Skier. These numbers of modes were determined through experimentation and visual inspection. The spectral decomposition of this $\tilde{A}$ was computed and the high-dimensional DMD modes Φ were constructed as described above. All DMD modes in this approach were considered to be background modes, with the idea that the first few SVD modes contain information that is most common throughout all video frames, which would be the background. Plots of the ω values can be seen in Figure 4 in Appendix C. All DMD modes were used to reconstruct the background using equation 9 as described above and columns of the resulting background matrix were reshaped into frames and viewed. The absolute value of this background matrix was subtracted from the input data X matrix to create the foreground matrix. The minimal pixel value of the foreground matrix was identified and its absolute value was added to each pixel in the matrix. Columns of this matrix were then reshaped into frames and then viewed.

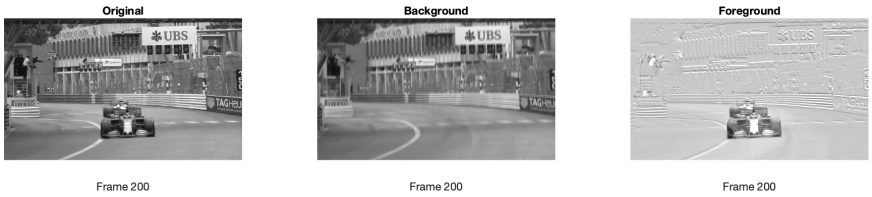
4 Computational Results

Figures 2 and 3 show results for each analysis method on one frame of the Monty Carlo and Skier video respectively. Three additional Monte Carlo frames can be seen in Figure 5 in Appendix C.

The foreground videos are very bright and contain many values higher than the maximum 255. Other methods of adjusting the foreground video were explored (including rescaling the pixel values so they all fell between 0 and 255, making all negative values 0 and leaving the rest, and rescaling pixel values to be between 0 and 255 and then thresholding out most of the grey values), but ultimately the method shown here yielded the most clear foreground video.

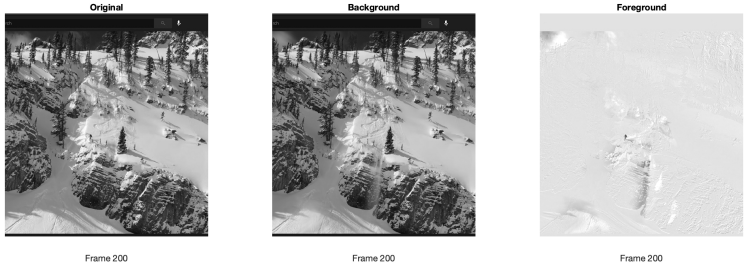


(a) Frame Constructed using Full-Rank DMD mode selection method



(b) Frame Constructed using Low-Rank Data Approximation Subtraction method

Figure 2: One frame from the foreground (right) and background (middle) of the Monte Carlo video constructed using the full-rank DMD mode selection method (a) and the low-rank data approximation subtraction method (b) compared against the original frames (left).



(a) Frame Constructed using Full-Rank DMD mode selection method

(b) Frame Constructed using Low-Rank Data Approximation Subtraction method

Figure 3: One frame from the foreground (right) and background (middle) of the Skier video constructed using the full-rank DMD mode selection method (a) and the low-rank data approximation subtraction method (b) compared against the original frames (left).

5 Summary and Conclusions

All of my conclusions for this project are qualitative. For the Monte Carlo video, I feel the low-rank data method did a better job of separating the background and the foreground when compared to the full-rank DMD mode selection method. In the center panel of Figure 4, you can see that the background found by the full-rank method contains a car that isn't present in this frame in the original video. This frame is absent from the low-rank background. Similarly, the car that is present in the original video looks clearer in the low-rank image than in the high-rank image.

I could not see much of a difference between the methods for the Skier video. Neither method did a terrific job of excluding the outlines of trees from the the foreground video, but the skier was clearly absent from the backgrounds and present in the foregrounds of both.

Because the POD modes must be orthogonal, it is not necessarily safe to assume that the first few POD modes accurately describe the background of a video. However in these videos this seems to have been the case. Future exploration could include merging these ideas and selecting a few DMD modes from the DMD of a relatively low-rank approximation of a dataset. Additionally, more efforts would need to be spent optimizing the foreground video. The foreground could be constructed using the DMD modes not used in the background, but this would not solve the issue that sparse matrices are very dark. I found that adding universal brightness yielded enough contrast to make out what was happening in the videos, but ideally there would be no pixel values outside the range of 0-255. My efforts to re scale pixels or set thresholds all yielded worse contrast, but there is surely a better method than the one I implemented. The method of adding residuals back into the background described in the spec added some movement to my background, but it is possible more tinkering could have yielded a better result.

This method of separating background from foreground relies entirely on the camera being fixed such that the background of the video does not change very much between frames. Any change in the background showed up in my foreground videos (this was most noticeable with the outlines of the trees in the skier video that sometimes shifted a bit between frames).

References

[1] Steven L. Brunton and Jose Nathan Kutz. *Data-driven science and Engineering: machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[2] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

[3] Jose Nathan Kutz et al. *Dynamic mode decomposition: data-driven modeling of complex systems*. Society for Industrial and Applied Mathematics, 2017.

Appendix A MATLAB Functions

Table 1: Notable Matlab Functions

`mcv = VideoReader('filename')`	creates object mcv to read video data from 'filename'
`B = reshape(A, sz)`	reshapes a given vector into a vector of specified size sz
`[U,S,V] = svd(AllNums, 'econ')`	performs single value decomposition on data in matrix AllNums. 'econ' flag eliminates zero rows/cols from U and S.
`k = find(X == 1)`	returns a vector of the same length of X containing 1's in indices where the value equals 1 and 0's otherwise. Used to parse individual digits out of data.
`[V2, D] = eig(Sb, Sw)`	returns diagonal matrix D of generalized eigenvalues and full matrix V2 whose columns are the corresponding left eigenvectors, so that W'*Sb = D*W'*Sw
`[omega_bg, bg] = mink(abs(omega), k)`	returns vector omega_bg of the minimum k values of abs(omega) and vector bg of the index locations of each of those values in omega
`fg = setdiff(1:r, bg)`	returns vector fg of the data in 1:r that isn't in bg, in sorted order.

Appendix B MATLAB Code

B.1 Full Rank Data DMD Mode Selection Method

```
%% Monte Carlo
clear all; close all;
%% Read Video
mcv = VideoReader('monte_carlo.mov');
mcduration = mcv.Duration;
mcframerate = mcv.FrameRate;
mcnumframes = mcv.NumFrames;
mcframeheight = mcv.Height;
mcframewidth = mcv.Width;

%% Make Matrix to hold all frames, each col = frame, each row = timepoint
mcmat = zeros(mcframeheight*mcframewidth, mcnumframes);

%% Read All Frames
for i = 1:mcnumframes
    frame = readFrame(mcv);
    frame = rgb2gray(frame);
    %whos frame
    %imshow(frame); drawnow
    mcmat(:,i) = reshape(frame, mcframeheight*mcframewidth, 1);
end
%% Create Matricies X_1^{M-1} amd X_2^M
X = mcmat;
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);

%% Step 1, compute SVD of X1
[U, Sigma, V] = svd(X1, 'econ');
r = mcnumframes-1; %start setting rank to number of frames in X1, full svd
%r = 2;
Ur = U(:, 1:r);
Sigmar = Sigma(1:r, 1:r);
Vr = V(:, 1:r);

%% Step 2, Project A onto the POD modes of U (leading r POD modes of A)
Atilde = Ur'*X2*Vr/Sigmar;

%% Step 3, Compute spectral decomposition of Atilde
[W, Lambda] = eig(Atilde);

%% Step 4, High-dimensional DMD modes Phi are reconstructed using eigenvectors
% W of reduced system and time-shifted snapshot matrix X2
Phi = X2*(Vr/Sigmar)*W;
alpha1 = Sigmar*Vr(1,:)';
b = (W*Lambda)\alpha1; %initial condition

%% Define t vector and omega (continuous time eigenvalues)
dt = 1;
mu = diag(Lambda);
omega = log(mu)/(dt);
%Omega is continuous time eigenvalues
```

```
t = 1:dt:mcnumframes;

%% plot absolute values of omegas
figure()
semilogx(abs(real(omega))*dt,abs(imag(omega))*dt,'r.','Markersize',15)
xlabel('abs(Real(\omega))')
ylabel('abs(Imaginary(\omega))')
set(gca,'FontSize',16)
xline(0);
yline(0);
title('Monte Carlo |Omega| Values Full Rank Method');

%% Separate foreground and background based on omega values
[omega_bg, bg] = mink(abs(omega), 1);
fg = setdiff(1:r, bg);

omega_fg = omega(fg); % foreground
Phi_fg = Phi(:,fg); % DMD foreground modes
omega_bg = omega(bg); % background
Phi_bg = Phi(:,bg); % DMD background mode

%% find background umodes
bg_b = Phi_bg \ X(:, 1); %initial background condition
bgu_modes = zeros(numel(omega_bg), mcnumframes);
for iter = 1:mcnumframes
    bgu_modes(:,iter) =  bg_b.*exp(omega_bg.*t(iter));
end
%% Find DMD Solutions for background
bg_dmd = Phi_bg*bgu_modes;

%% Watch Background Video

figure()
for i = 1:mcnumframes
    frame = reshape(bg_dmd(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end


%% Watch Foreground Video
foreground = X - abs(bg_dmd);
brightness_to_add = abs(min(foreground(:)));
foreground = foreground + brightness_to_add;
figure()
for i = 1:mcnumframes
    frame = reshape(foreground(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end
%% Make Figures
figure()
subplot(3,3,1)
frame = reshape(X(:,1), mcframeheight,mcframewidth);
```

```
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Original")
xlabel("Frame 1")

subplot(3,3,2)
frame = reshape(X(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
xlabel("Frame 200")

subplot(3,3,3)
frame = reshape(X(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
xlabel("Frame 250")

subplot(3,3,4)
frame = reshape(bg_dmd(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Background")

subplot(3,3,5)
frame = reshape(bg_dmd(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow


subplot(3,3,6)
frame = reshape(bg_dmd(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow

subplot(3,3,7)
frame = reshape(foreground(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Foreground")

subplot(3,3,8)
frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow


subplot(3,3,9)
frame = reshape(foreground(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow


sgtitle('Monte Carlo Full-Rank Data DMD Mode Selection Method')

%% Make big figure for Monte Carlo
figure()
subplot(1,3,1)
frame = reshape(X(:,200), mcframeheight,mcframewidth);
```

```
imshow(uint8(frame)); drawnow
title("Original")
xlabel("Frame 200")

subplot(1,3,2)
frame = reshape(bg_dmd(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
title("Background")
xlabel("Frame 200")

subplot(1,3,3)
frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
title("Foreground")
xlabel("Frame 200")

sgtitle('Monte Carlo Full-Rank Data DMD Mode Selection Method')
%% Skier
clear all; close all;
%% Read Video
mcv = VideoReader('ski_drop.mov');
mcduration = mcv.Duration;
mcframerate = mcv.FrameRate;
mcnumframes = mcv.NumFrames;
mcframeheight = mcv.Height;
mcframewidth = mcv.Width;

%% Make Matrix to hold all frames, each col = frame, each row = timepoint
mcmat = zeros(mcframeheight*mcframewidth, mcnumframes);

%% Read All Frames
for i = 1:mcnumframes
    frame = readFrame(mcv);
    frame = rgb2gray(frame);
    %whos frame
    %imshow(frame); drawnow
    mcmat(:,i) = reshape(frame, mcframeheight*mcframewidth, 1);
end
%% Create Matricies X1 amd X2
X = mcmat;
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);

%% Step 1, compute SVD of X1
[U, Sigma, V] = svd(X1, 'econ');
r = mcnumframes-1; %start setting rank to number of frames in X1, full svd
%r = 2;
Ur = U(:, 1:r);
Sigmar = Sigma(1:r, 1:r);
Vr = V(:, 1:r);

%% Step 2, Project A onto the POD modes of U (leading r eigenvalues and
    eigenvectors of A)
Atilde = Ur'*X2*Vr/Sigmar;
```

```
213
214 %% Step 3, Compute spectral decomposition of Atilde
215 [W, Lambda] = eig(Atilde);
216
217 %% Step 4, High-dimensional DMD modes Phi are reconstructed using eigenvectors
218 % W of reduced system and time-shifted snapshot matrix X2
219 Phi = X2*(Vr/Sigmar)*W;
220 alpha1 = Sigmar*Vr(1,:)';
221 b = (W*Lambda)\alpha1; %initial condition
222
223 %% Define t vector and omega
224 dt = 1;
225 mu = diag(Lambda);
226 omega = log(mu)/(dt);
227 %Omega is continuous time eigenvalues
228 t = 1:dt:mcnumframes;
229
230 %% plot absolute values of omegas
231 figure()
232 semilogx(abs(real(omega))*dt,abs(imag(omega))*dt,'r.','Markersize',15)
233 xlabel('abs(Real(\omega))')
234 ylabel('abs(Imaginary(\omega))')
235 set(gca,'FontSize',16)
236 title('Skier |Omega| Values Full Rank Method');
237
238 %% Separate foreground and background based on omega values
239 [omega_bg, bg] = mink(abs(omega), 3);
240 fg = setdiff(1:r, bg);
241
242 omega_fg = omega(fg); % foreground
243 Phi_fg = Phi(:,fg); % DMD foreground modes
244 omega_bg = omega(bg); % background
245 Phi_bg = Phi(:,bg); % DMD background modes
246
247 %% find background umodes
248 bg_b = Phi_bg \ X(:, 1); %initial background condition
249 bgu_modes = zeros(numel(omega_bg), mcnumframes);
250 for iter = 1:mcnumframes
251     bgu_modes(:,iter) =  bg_b.*exp(omega_bg.*t(iter));
252 end
253 %% Find DMD Solutions for background
254 bg_dmd = Phi_bg*bgu_modes;
255
256 %% Watch Background Video
257
258 figure()
259 for i = 1:mcnumframes
260     frame = reshape(bg_dmd(:,i), mcframeheight, mcframewidth);
261     imshow(uint8(frame)); drawnow
262     disp(i);
263 end
264
265
266 %% Make Foreground Video
```

```
%Subtract background
foreground = X - abs(bg_dmd);
%Brighten so no negative values
brightness_to_add = abs(min(foreground(:)));
foreground = foreground + brightness_to_add;

%% optional rescaling of foreground (decided against this)
% %Rescale pixel values to be between 0 and 255
% foreground = foreground/max(foreground(:)); %as percents
% %threshould out everything grey
% foreground(foreground > .5) = 1;
%
% foreground = foreground .* 255; %rescale to 255

% %% Alternate Make Foreground Video
% %Subtract background
% foreground = X - abs(bg_dmd);
% %Make negative values 0
% foreground(foreground<0) = 0;

%% Watch Foreground Video
figure()
for i = 1:mcnumframes
    frame = reshape(foreground(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end

%% Make Figures
figure()
subplot(3,3,1)
frame = reshape(X(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Original")
xlabel("Frame 1")

subplot(3,3,2)
frame = reshape(X(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
xlabel("Frame 200")

subplot(3,3,3)
frame = reshape(X(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
xlabel("Frame 250")

subplot(3,3,4)
frame = reshape(bg_dmd(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Background")
```

```
321
322 subplot(3,3,5)
323 frame = reshape(bg_dmd(:,200), mcframeheight,mcframewidth);
324 imshow(uint8(frame)); drawnow
325
326
327 subplot(3,3,6)
328 frame = reshape(bg_dmd(:,250), mcframeheight,mcframewidth);
329 imshow(uint8(frame)); drawnow
330
331 subplot(3,3,7)
332 frame = reshape(foreground(:,1), mcframeheight,mcframewidth);
333 imshow(uint8(frame)); drawnow
334 set(gca,'xaxisLocation','top')
335 ylabel("Foreground")
336
337 subplot(3,3,8)
338 frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
339 imshow(uint8(frame)); drawnow
340
341
342 subplot(3,3,9)
343 frame = reshape(foreground(:,250), mcframeheight,mcframewidth);
344 imshow(uint8(frame)); drawnow
345
346
347 sgtitle('Skier Full-Rank Data DMD Mode Selection Method')
348
349 %% Make big figure for Skier
350 figure()
351 subplot(1,3,1)
352 frame = reshape(X(:,200), mcframeheight,mcframewidth);
353 imshow(uint8(frame)); drawnow
354 title("Original")
355 xlabel("Frame 200")
356
357 subplot(1,3,2)
358 frame = reshape(bg_dmd(:,200), mcframeheight,mcframewidth);
359 imshow(uint8(frame)); drawnow
360 title("Background")
361 xlabel("Frame 200")
362
363 subplot(1,3,3)
364 frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
365 imshow(uint8(frame)); drawnow
366 title("Foreground")
367 xlabel("Frame 200")
368
369 sgtitle('Skier Full-Rank Data DMD Mode Selection Method')
```

B.2 Low Rank Data Approximation Subtraction Method

```
%% Monte Carlo
clear all; close all;
%% Read Video
mcv = VideoReader('monte_carlo.mov');
mcduration = mcv.Duration;
mcframerate = mcv.FrameRate;
mcnumframes = mcv.NumFrames;
mcframeheight = mcv.Height;
mcframewidth = mcv.Width;

%% Make Matrix to hold all frames, each col = frame, each row = timepoint
mcmat = zeros(mcframeheight*mcframewidth, mcnumframes);

%% Read All Frames
for i = 1:mcnumframes
    frame = readFrame(mcv);
    frame = rgb2gray(frame);
    %whos frame
    %imshow(frame); drawnow
    mcmat(:,i) = reshape(frame, mcframeheight*mcframewidth, 1);
end
%% Create Matricies X1 amd X2
X = mcmat;
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);

%% Step 1, compute SVD of X1
[U, Sigma, V] = svd(X1, 'econ');
%r = mcnumframes-1; %start setting rank to number of frames in X1, full svd
r = 2;
Ur = U(:, 1:r);
Sigmar = Sigma(1:r, 1:r);
Vr = V(:, 1:r);

%% Step 2, Project A onto the POD modes of U
%(get leading r eigenvalues and eigenvectors of A)
Atilde = Ur'*X2*Vr/Sigmar;

%% Step 3, Compute spectral decomposition of Atilde
[W, Lambda] = eig(Atilde);

%% Step 4, High-dimensional DMD modes Phi are reconstructed using eigenvectors
% W of reduced system and time-shifted snapshot matrix X2
Phi = X2*(Vr/Sigmar)*W;
alpha1 = Sigmar*Vr(1,:)';
b = (W*Lambda)\alpha1; %initial condition?

%% Define t vector and omega
dt = 1;
mu = diag(Lambda);
omega = log(mu)/dt;
t = 0:dt:mcnumframes;

```

```
%% plot absolute values of omegas
figure()
semilogx(abs(real(omega))*dt,abs(imag(omega))*dt,'r.','Markersize',15)
xlabel('abs(Real(\omega))')
ylabel('abs(Imaginary(\omega))')
set(gca,'FontSize',16)
xline(0);
yline(0);
title('Monte Carlo |Omega| Values Low Rank Method');
%% find umodes
u_modes = zeros(length(b), mcnumframes);
for iter = 1:mcnumframes
    u_modes(:,iter) = b.*exp(omega*t(iter));
end
%% Find DMD Solution
u_dmd = Phi*u_modes;

%% Watch Background Video
figure()
for i = 1:mcnumframes
    frame = reshape(u_dmd(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end

%% Watch Foreground Video
foreground = X - abs(u_dmd);
brightness_to_add = abs(min(foreground(:)));
foreground = foreground + brightness_to_add;
figure()
for i = 1:mcnumframes
    frame = reshape(foreground(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end

%% Make Figures
figure()
subplot(3,3,1)
frame = reshape(X(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Original")
xlabel("Frame 1")

subplot(3,3,2)
frame = reshape(X(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
xlabel("Frame 200")

subplot(3,3,3)
frame = reshape(X(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
```

```
set(gca,'xaxisLocation','top')
xlabel("Frame 250")

subplot(3,3,4)
frame = reshape(u_dmd(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Background")

subplot(3,3,5)
frame = reshape(u_dmd(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow


subplot(3,3,6)
frame = reshape(u_dmd(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow

subplot(3,3,7)
frame = reshape(foreground(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Foreground")

subplot(3,3,8)
frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow


subplot(3,3,9)
frame = reshape(foreground(:,250), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow


sgtitle('Monte Carlo Low-Rank Subtraction Results')

%% Make big figure for Monte Carlo
figure()
subplot(1,3,1)
frame = reshape(X(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
title("Original")
xlabel("Frame 200")

subplot(1,3,2)
frame = reshape(u_dmd(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
title("Background")
xlabel("Frame 200")

subplot(1,3,3)
frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
title("Foreground")
```

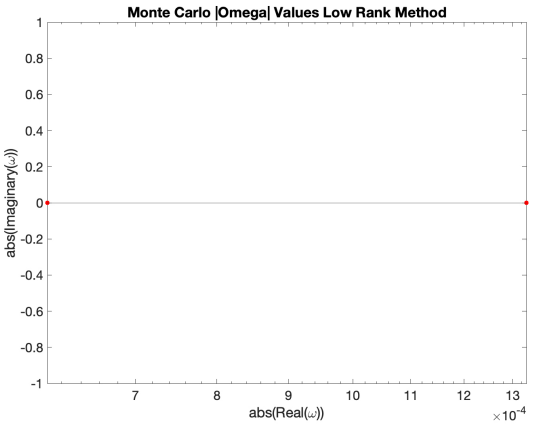
```
xlabel("Frame 200")

sgtitle('Monte Carlo Low-Rank Subtraction Method')

%% Skier
clear all; close all;
%% Read Video
mcv = VideoReader('ski_drop.mov');
mcduration = mcv.Duration;
mcframerate = mcv.FrameRate;
mcnumframes = mcv.NumFrames;
mcframeheight = mcv.Height;
mcframewidth = mcv.Width;

%% Make Matrix to hold all frames, each col = frame, each row = timepoint
mcmat = zeros(mcframeheight*mcframewidth, mcnumframes);

%% Read All Frames
for i = 1:mcnumframes
    frame = readFrame(mcv);
    frame = rgb2gray(frame);
    %whos frame
    %imshow(frame); drawnow
    mcmat(:,i) = reshape(frame, mcframeheight*mcframewidth, 1);
end
%% Create Matricies X_1^{M-1} amd X_2^M
X = mcmat;
X1 = X(:, 1:end-1);
X2 = X(:, 2:end);

%% Step 1, compute SVD of X1
[U, Sigma, V] = svd(X1, 'econ');
%r = mcnumframes-1; %start setting rank to number of frames in X1, full svd
r = 8;
Ur = U(:, 1:r);
Sigmar = Sigma(1:r, 1:r);
Vr = V(:, 1:r);

%% Step 2, Project A onto the POD modes of U (leading r eigenvalues and
   eigenvectors of A)
Atilde = Ur'*X2*Vr/Sigmar;

%% Step 3, Compute spectral decomposition of Atilde
[W, Lambda] = eig(Atilde);

%% Step 4, High-dimensional DMD modes Phi are reconstructed using eigenvectors
% W of reduced system and time-shifted snapshot matrix X2
Phi = X2*(Vr/Sigmar)*W;
alpha1 = Sigmar*Vr(1,:)';
b = (W*Lambda)\alpha1; %initial condition?

%% Define t vector and omega
dt = 1;
mu = diag(Lambda);
```

```
omega = log(mu)/(dt);
t = 1:dt:mcnumframes;

%% plot absolute values of omegas
figure()
semilogx(abs(real(omega))*dt,abs(imag(omega))*dt,'r.','Markersize',15)
xlabel('abs(Real(\omega))')
ylabel('abs(Imaginary(\omega))')
set(gca,'FontSize',16)
xline(0);
yline(0);
title('Skier |Omega| Values Low Rank Method');
%% find umodes
u_modes = zeros(length(b), mcnumframes);
for iter = 1:mcnumframes
    u_modes(:,iter) = b.*exp(omega*t(iter));
end
%% Find DMD Solution
u_dmd = Phi*u_modes;

%% Watch Background Video
figure()
for i = 1:mcnumframes
    frame = reshape(u_dmd(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end

%% Watch Foreground Video
foreground = X - abs(u_dmd);
brightness_to_add = abs(min(foreground(:)));
foreground = foreground + brightness_to_add;

figure()
for i = 1:mcnumframes
    frame = reshape(foreground(:,i), mcframeheight, mcframewidth);
    imshow(uint8(frame)); drawnow
    disp(i);
end

%% Make Figures
figure()
subplot(3,3,1)
frame = reshape(X(:,1), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
ylabel("Original")
xlabel("Frame 1")

subplot(3,3,2)
frame = reshape(X(:,200), mcframeheight,mcframewidth);
imshow(uint8(frame)); drawnow
set(gca,'xaxisLocation','top')
xlabel("Frame 200")
```

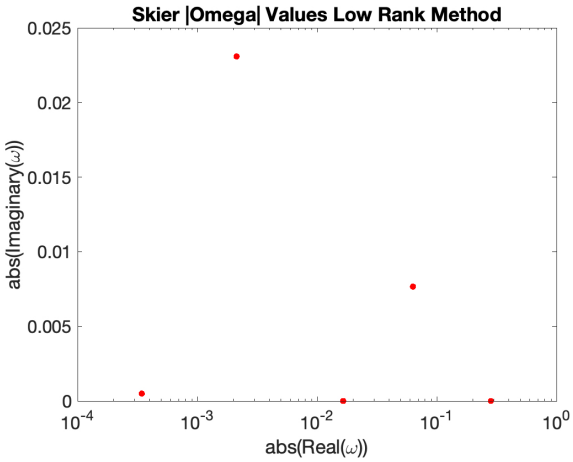
```
269
270 subplot(3,3,3)
271 frame = reshape(X(:,250), mcframeheight,mcframewidth);
272 imshow(uint8(frame)); drawnow
273 set(gca,'xaxisLocation','top')
274 xlabel("Frame 250")
275
276 subplot(3,3,4)
277 frame = reshape(u_dmd(:,1), mcframeheight,mcframewidth);
278 imshow(uint8(frame)); drawnow
279 set(gca,'xaxisLocation','top')
280 ylabel("Background")
281
282 subplot(3,3,5)
283 frame = reshape(u_dmd(:,200), mcframeheight,mcframewidth);
284 imshow(uint8(frame)); drawnow
285
286
287 subplot(3,3,6)
288 frame = reshape(u_dmd(:,250), mcframeheight,mcframewidth);
289 imshow(uint8(frame)); drawnow
290
291 subplot(3,3,7)
292 frame = reshape(foreground(:,1), mcframeheight,mcframewidth);
293 imshow(uint8(frame)); drawnow
294 set(gca,'xaxisLocation','top')
295 ylabel("Foreground")
296
297 subplot(3,3,8)
298 frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
299 imshow(uint8(frame)); drawnow
300
301
302 subplot(3,3,9)
303 frame = reshape(foreground(:,250), mcframeheight,mcframewidth);
304 imshow(uint8(frame)); drawnow
305
306
307 sgtitle('Skier Low-Rank Subtraction Results')
308
309 %% Make big figure for Skier
310 figure()
311 subplot(1,3,1)
312 frame = reshape(X(:,200), mcframeheight,mcframewidth);
313 imshow(uint8(frame)); drawnow
314 title("Original")
315 xlabel("Frame 200")
316
317 subplot(1,3,2)
318 frame = reshape(u_dmd(:,200), mcframeheight,mcframewidth);
319 imshow(uint8(frame)); drawnow
320 title("Background")
321 xlabel("Frame 200")
322
```

```
323 subplot(1,3,3)
324 frame = reshape(foreground(:,200), mcframeheight,mcframewidth);
325 imshow(uint8(frame)); drawnow
326 title("Foreground")
327 xlabel("Frame 200")
328
329 sgtitle('Skier Low-Rank Subtraction Method')
```

Appendix C Additional Figures

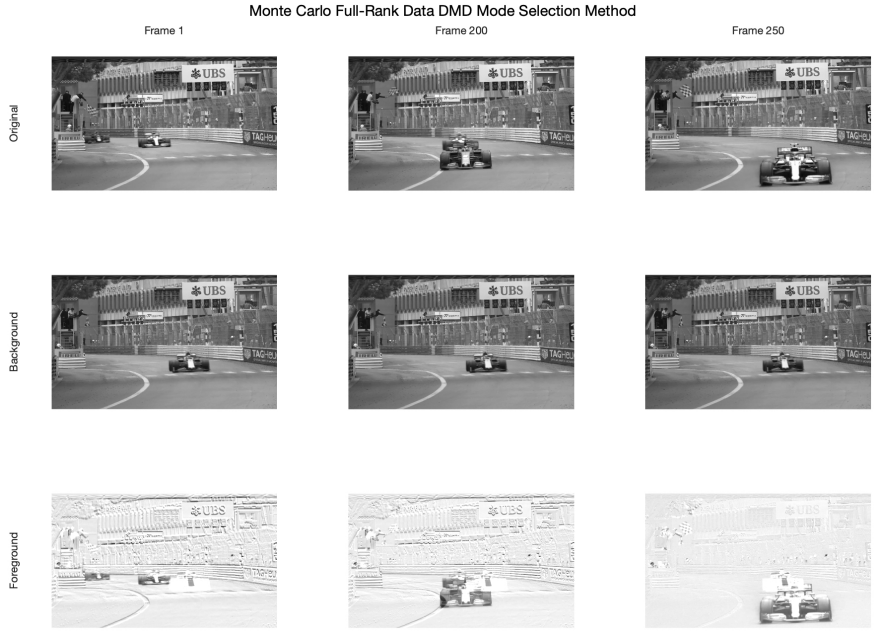


(a) Low Rank Monte Carlo Omega Magnitudes

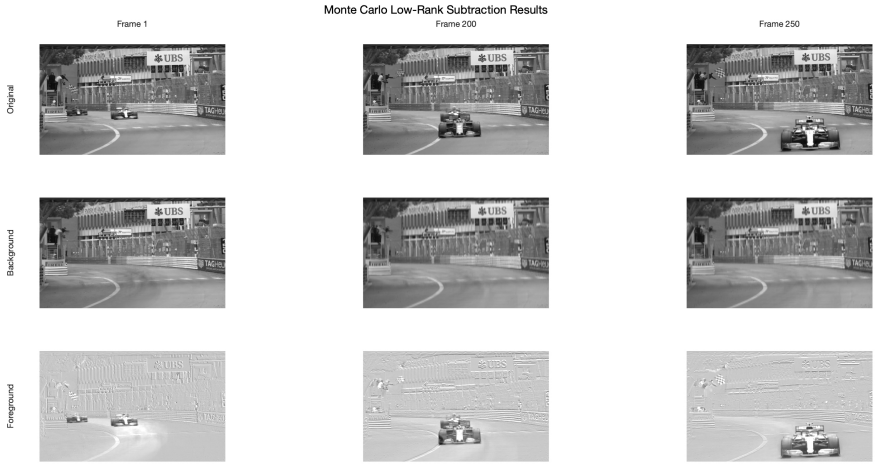


(b) Low Rank Skier Omega Magnitudes

Figure 4: Absolute values of real and imaginary parts of continuous time eigenvalues ω for the Monte Carlo (a) and Skier (b) low-rank data DMD mode selection analyses. All modes (2 for Monte Carlo, 8 for the skier) were selected in both cases to be background modes. Six of the Skier omega values are in pairs of complex conjugates, so when plotting the absolute value as done here there are three instances of two values being plotted in the exact same location (hence the apparent presence of five points when there were eight omega values).



(a) Frames Constructed using Full-Rank DMD mode selection method



(b) Frames Constructed using Low-Rank Data Approximation Subtraction method

Figure 5: Three frames from the foreground (bottom) and background (middle) of the Monte Carlo video constructed using the full-rank DMD mode selection method (a) and the low-rank data approximation subtraction method (b) compared against the original frames (top).