**Homework 4: Background Subtraction in Video Streams**

Daniel Nguyen

February 26, 2018

**Abstract**

We use Dynamic Mode Decomposition (DMD) in order to separate a video’s background from its foreground. Given videos filmed from an iPhone, we will apply the SVD to speed up the computation speed of our methods, and we will then use DMD algorithms to extract the video’s foreground and background.

**Contents**

**1 Introduction**

**2 Theory**

**3 MATLAB Procedure**

**4 Results**

**5 Summary and Conclusions**

**Appendix A MATLAB Functions Used and Extra Graphs**

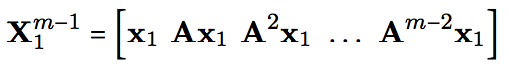
**Appendix B MATLAB Code**

**1 Introduction**

The Dynamic Mode Decomposition (DMD) is a very powerful tool when it comes to the analysis of nonlinear systems and the dynamics that make them. The idea behind DMD is to take a “snapshot” of data in time, , in order to predict the future state of the system . We can express this in the linear form . This tool has originated from the field of fluid dynamics in order to study the behavior of fluids over time, and has been applied to many other areas as a powerful tool in order to analyze the dynamics of a system. In this assignment, we will be using the concepts of the DMD in order to separate the background from the foreground of several different videos.

**2 Theory**

We start off by taking a snapshot of the data that we are given, from , then another snapshot of the data shifted by one step in time, . We can first form a Krylov space, which is given by



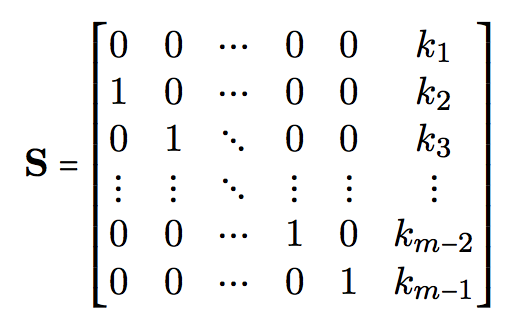
(1)

where is known as a Koopman operator (which estimates the possible nonlinear dynamics of a system), and the relationship between the two snapshots taken is given by the following equation

(2))



where the ’s are the coefficients of the Krylov space basis vectors and is the following matrix



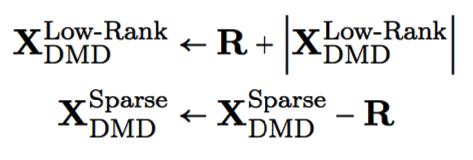
We will use these later in finding the Fourier modes and frequencies necessary to reconstruct our video. With the concept of the DMD in mind, we can use this in order to distinguish between a video’s foreground and background. In the DMD, a video can be represented by two video sequences:

(3)

Where ( is the number of low rank modes), is the eigenvector of the Koopman operator , and represents the Fourier modes. The foreground of the video is represented by the left term, while the background of the video is represented by . If we wish to find DMD, we can use the following equation to do so:

(4)

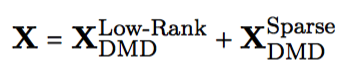
Where represents the modulus of each element within the matrix. The reason why the absolute value of low-rank representation of X is because that yields a matrix with complex values. We only want real-valued outputs for the analysis of the results of our system, and getting rid of complex values can make a big difference in accuracy. But this may result in negative values in some elements for , and negative pixel intensities do not make sense. So these residual negative values can be put back into as follows:



(6)

(5)

Now the magnitudes of the complex values for the DMD reconstruction are accounted for while maintaining the important constraints that



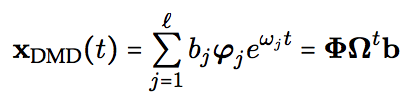
(7)

so that there are no pixel intensities below zero, and gives us the approximate low-rank and sparse DMD construction of the video.

**3 MATLAB Procedure**

We will work with three videos in this assignment: one video of me moving my hand in front of a camera in a study room, one of cars moving down a street near Dempsey Hall at the University of Washington, and one of me eating an egg. For the first video 1920 x 1080 video was recorded using an iPhone 6 of a hand moving in front of a camera. The video was converted to grayscale then reduced to a 480 x 270 video using imresize for each frame in order to reduce the computational speed. The video is then reshaped into a (480\*270) x 80 matrix (where 80 is the number of frames).

Once we have this matrix, we can perform an SVD on it in order to truncate our data in order to get the few dominant modes we need to get a low-rank representation of our video. We will use to represent the number of modes that we want, and so we take the first columns of our U and V matrices and the first singular value of our matrix. Using this, we can compute the a low rank representation of our video

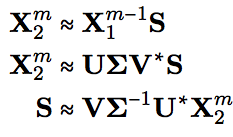


(8)

Where and are found using the eigen-decomposition of a Koopman operator in



The Fourier modes can be calculated after doing a eigen-decomposition on a a matrix similar to , which we will call . This can be found by first taking the SVD of the first snapshot, , and we can take



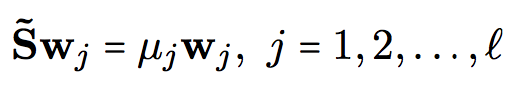
(9)

and using the similarity transform , we get



(10)

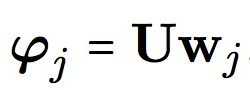
and doing the eigen-decomposition on gives us , solving the eigenvalue problem



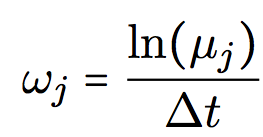
(11)

We can now use this to find the approximate eigenvalue, approximated by

(12)



and the DMD eigenvalues can be converted to Fourier modes by defining



(13)

Once we have calculated our background video, , we can subtract it the modulus of it from the original video X in order get our , which would be our foreground video.

And to avoid negative pixel intensities as mentioned earlier, we calculate a residual matrix which consists of the negative values from , and we subtract that matrix from the sparse video and add it to to get our new DMD representations.

We repeat this process for videos 2 and 3, but these videos have varying dimensions (different quality and length as indicated by the number of frames.

**4 Results**

For the first video, I recorded myself moving my hand across the front of the camera while attempting keeping the camera while holding it in the air.

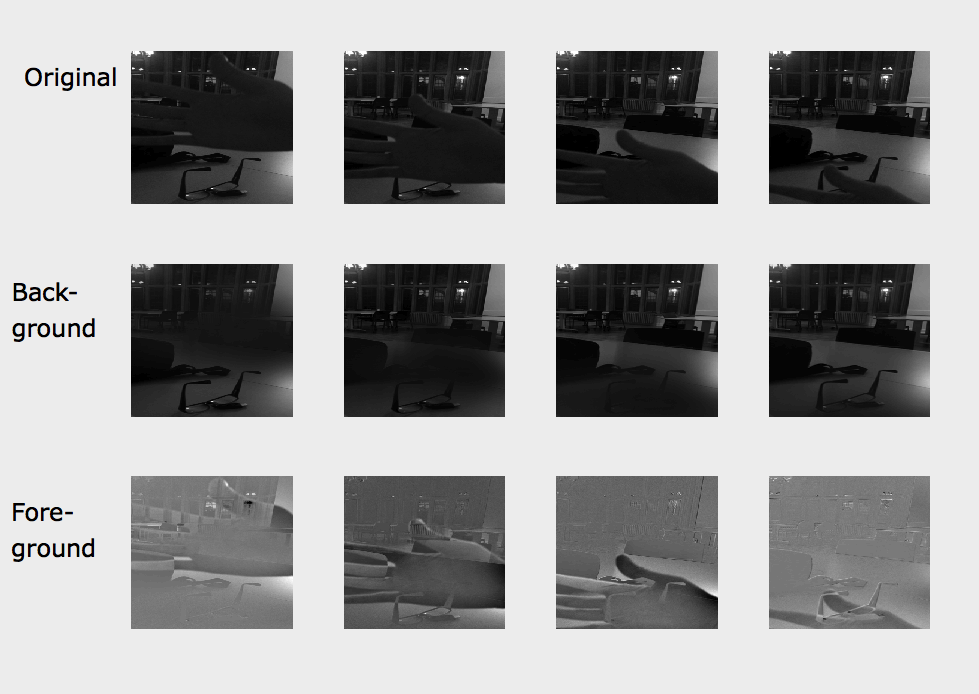


Figure 1: Snapshots of the original video, the background, and the foreground at frames 15, 30, 45 and 60. This case was constructed using the first 5 dominant modes.

For the most part, the separation here works very well. The hand in the foreground is correctly captured moving in the foreground video. Although I have decided to use the figure involving 5 modes, any traces of the hand in not visible in the background of the video. This brings us to another matter; I have also noticed that the more modes used, the more of the foreground is picked up in the sparse video, which we can see as indicated by the moving shadow in the background video.

For my next case, I chose to record a car run down the street on the north campus of the University of Washington while keeping the camera stationary.

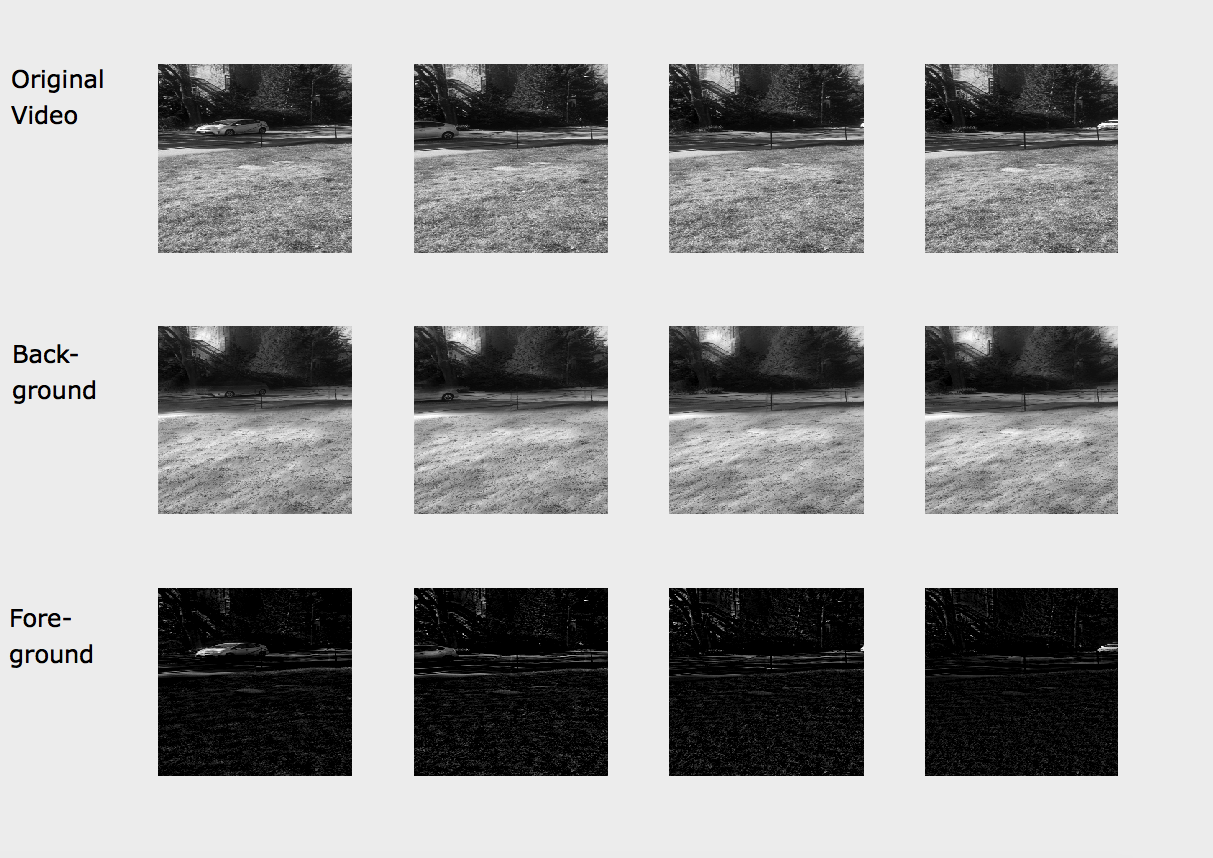


Figure 2: Snapshots of a car driving down a hill at the same frame interval as in Figure 1. However, these videos show frames based on the DMD reconstruction using 1 mode only, rather than 5.

The video separation works mostly well in this case. It doesn’t perfectly black out the grass and the trees in the original video, but this can be explained by the wind and perhaps the ever-slight motion of the camera (I did not use a tripod). We see in the background video, the car is, for the most part, successfully filtered out of the video. The motion of the car was very clear and distinct from the rest of the features in the video.

Lastly, I chose a video where the camera moves during the video as well as the foreground, which shifts the background as the video progresses.



Figure 3: Snapshots of me eating an egg at the same frame interval as in the previous videos.

Here, our DMD separation of the video does not work so well. Even if we use just 1 mode to approximate our video, my great deal of my general figure is taken in to be a part of the background, and this fact only becomes more true as we increase the number of modes. I can attribute this to my body being very stationary in the lower half of the video for the entire duration, so there is no motion of my body for PCA to pick up, and so I am interpreted as part of the background. And for the foreground video, the kitchen cabinets in the back were also picked up. This can be explained by the fact that the camera is moving in the video, and so the motion of the cabinets is taken to be as a part of the foreground.

**5 Summary and Conclusions**

A video can be separated into a background and a foreground, both of which can be extracted using a Dynamic Mode Decomposition, which utilizes Principal Component Analysis in order to have a low-rank representation of the original stream of data from which to distinguish from. But the algorithm that I built did not work 100% perfectly, since it was reliant on movement & stillness of the features in the original video in order to make its distinctions, and there is always some noise when it comes to recording (like wind blowing grass and leaves on trees in the second video, or slight shakiness of the camera from the person recording). But we found that when a camera is kept still and controlled, the DMD algorithm, for the most part, works great, as in cases 1 and 2. But for case 3, where there is limited movement of the foreground object and more movement of the background, the algorithm does not work to its fullest potential. There are perhaps more advanced algorithms involving more nuanced math in order to handle these sensitive dependencies on background movement, which the author should like to investigate.

**Appendix A MATLAB Functions Used**

No special functions were used in this particular assignment besides svd(). Most of the functions were just for image manipulation/resizing and plotting.

rgb2gray();

imresize(); to rescale

reshape();

svd();

**Appendix B MATLAB Code**

clear all;

close all;

clc;

%% Video 1: Study Hall

video = [];

v = VideoReader('IMG\_3132.MOV');

while hasFrame(v)

frame = readFrame(v);

frame = rgb2gray(frame);

frame = reshape (frame, [], 1);

video = [video, frame];

end

video = reshape(video, [1920,1080,80]);

video = imresize(video,.25);

video = double(video);

video = reshape(video, [480\*270,80]);

v1 = video(:,1:end-1);

v2 = video(:,2:end);

[U, Sigma, V] = svd(v1, 'econ');

%%

r=1;

Sr = Sigma(1:r, 1:r);

Ur = U(:, 1:r);

Vr = V(:, 1:r);

Stilde = Ur'\*v2\*Vr\*diag(1./diag(Sr));

[eV, D] = eig(Stilde);

mu = diag(D);

omega = log(mu);

Phi = v2\*Vr/Sr\*eV;

y0 = Phi\video(:,1);

v\_modes = zeros(r,length(v1(1,:)));

for i = 1:length(v1(1,:))

v\_modes(:,i) = (y0.\*exp(omega\*i));

end

v\_dmd = Phi\*v\_modes;

v\_dmd = abs(v\_dmd);

v\_sparse = v1 - v\_dmd;

residual\_matrix = v\_sparse.\*(v\_sparse < 0);

v\_dmd = residual\_matrix + abs(v\_dmd);

v\_sparse = v\_sparse - residual\_matrix;

uvid = reshape(v\_dmd, [480, 270, 79]);

figure(1)

for i = 1:12

subplot(3,4,i)

vidtype = floor((i-1)/4);

timeframe = mod(i,4);

if timeframe == 0

timeframe = 4;

end

if vidtype == 0

temp = video(:,timeframe\*15);

elseif vidtype == 1

temp = v\_dmd(:,timeframe\*15);

elseif vidtype == 2

temp = v\_sparse(:,timeframe\*15);

end

temp = reshape(temp, 480, 270);

imagesc(temp);

colormap(gray);

axis off;

end

%% Video 2: Cars Running Downhill

video = [];

v = VideoReader('IMG\_3154.MOV');

while hasFrame(v)

frame = readFrame(v);

frame = rgb2gray(frame);

frame = reshape (frame, [], 1);

video = [video, frame];

end

video = reshape(video, [1920,1080,147]);

video = imresize(video,.25);

video = double(video);

video = reshape(video, [480\*270,147]);

v1 = video(:,1:end-1);

v2 = video(:,2:end);

[U, Sigma, V] = svd(v1, 'econ');

%%

r=3;

Sr = Sigma(1:r, 1:r);

Ur = U(:, 1:r);

Vr = V(:, 1:r);

Stilde = Ur'\*v2\*Vr\*diag(1./diag(Sr));

[eV, D] = eig(Stilde);

mu = diag(D);

omega = log(mu);

Phi = v2\*Vr/Sr\*eV;

y0 = Phi\video(:,1);

v\_modes = zeros(r,length(v1(1,:)));

for i = 1:length(v1(1,:))

v\_modes(:,i) = (y0.\*exp(omega\*i));

end

v\_dmd = Phi\*v\_modes;

v\_dmd = abs(v\_dmd);

v\_sparse = v1 - v\_dmd;

residual\_matrix = v\_sparse.\*(v\_sparse < 0);

v\_dmd = residual\_matrix + abs(v\_dmd);

v\_sparse = v\_sparse - residual\_matrix;

uvid = reshape(v\_dmd, [480, 270, 146]);

figure(2)

for i = 1:12

subplot(3,4,i)

vidtype = floor((i-1)/4);

timeframe = mod(i,4);

if timeframe == 0

timeframe = 4;

end

if vidtype == 0

temp = video(:,timeframe\*15);

elseif vidtype == 1

temp = v\_dmd(:,timeframe\*15);

elseif vidtype == 2

temp = v\_sparse(:,timeframe\*15);

end

temp = reshape(temp, 480, 270);

imagesc(temp);

colormap(gray);

axis off;

end

%% Video 3: Eating an Egg

video = [];

v = VideoReader('52359343966\_\_6F3FF01D-F7C5-473A-A049-FD070AC65BD6.MOV');

while hasFrame(v)

frame = readFrame(v);

frame = rgb2gray(frame);

frame = reshape (frame, [], 1);

video = [video, frame];

end

video = reshape(video, [1280,720,95]);

video = imresize(video,.25);

video = double(video);

video = reshape(video, [320\*180,95]);

v1 = video(:,1:end-1);

v2 = video(:,2:end);

[U, Sigma, V] = svd(v1, 'econ');

%%

r=1;

Sr = Sigma(1:r, 1:r);

Ur = U(:, 1:r);

Vr = V(:, 1:r);

Stilde = Ur'\*v2\*Vr\*diag(1./diag(Sr));

[eV, D] = eig(Stilde);

mu = diag(D);

omega = log(mu);

Phi = v2\*Vr/Sr\*eV;

y0 = Phi\video(:,1);

v\_modes = zeros(r,length(v1(1,:)));

for i = 1:length(v1(1,:))

v\_modes(:,i) = (y0.\*exp(omega\*i));

end

v\_dmd = Phi\*v\_modes;

v\_dmd = abs(v\_dmd);

v\_sparse = v1 - v\_dmd;

residual\_matrix = v\_sparse.\*(v\_sparse < 0);

v\_dmd = residual\_matrix + abs(v\_dmd);

v\_sparse = v\_sparse - residual\_matrix;

uvid = reshape(v\_dmd, [320, 180, 94]);

figure(3);

for i = 1:12

subplot(3,4,i)

vidtype = floor((i-1)/4);

timeframe = mod(i,4);

if timeframe == 0

timeframe = 4;

end

if vidtype == 0

temp = video(:,timeframe\*15);

elseif vidtype == 1

temp = v\_dmd(:,timeframe\*15);

elseif vidtype == 2

temp = v\_sparse(:,timeframe\*15);

end

temp = reshape(temp, 320, 180);

imagesc(temp);

colormap(gray);

axis off;

end