AMATH 482 Homework 5
Background Subtraction in Video Streams

Weien Guo

March 17, 2021

Abstract

In this assignment, we will focus on Dynamic Mode Decomposition (DMD), which is a very useful tool that helps us to forecast the data, meaning to predict the future movement. It does an amazing performance in finding spatial modes and using exponential functions for calculating the dynamics. Here, we are given two videos, and our task is to apply DMD algorithm to separate the videos into foreground and background parts.

1 Introduction and Overview

After exploring SVT and PCA in the previous assignment, we know that we would always want to reduce a high dimensional data set to have lower dimensionality. However, we need the governing equation, in other words, the model. When the observed object is low dimensional and dynamic, SVD would not work since we do not have a governing equation for each time step. In order to predict what will happen in the data, we would use dynamic Mode Decomposition. With the video files provided, DMD is able to separate the foreground and background in the videos since the data is spatio-temporal, meaning it evolves in both time and space (i.e. object moves, and the frames has constant time gaps).

The first video, `monte_carlo_low.mp4` has 379 frames with 540 pixels in height and 960 pixels in width. The video is shooting from a stable location, with racing cars come and go (Figure 1). Our expected foreground result from DMD would be the racing cars, and the background is with buildings and racing track. The second video `ski_drop_low.mp4` has 453 frames of size 540*960. It films a skier skiing down from a mountain (Figure 2). The expected foreground would include the movement of the skier and the snow splashes from the skis. The background would be the snow track, the mountain, and the trees.

Figure 1: The original frame 30 from video 1

Figure 2: The original frame 50 from video 2

2 Theoretical Background

Before we actually getting to the core of DMD algorithm, we would first need to perform singular value decomposition on a matrix. The details about SVD could be found in previous assignments.

2.1 Dynamic Mode Decomposition

Our data $\mathbf{X}$ is spatio-temporal, and is collected at regularly spaced intervals $\Delta t > 0$. The *Koopman Operator* $\mathbf{A}$ is a linear, time-dependent operator such that

$$x_{j+1} = \mathbf{A}x_j,$$

x_j is the data collected at the j^{th} time step and x_{j+1} is t the $(j+1)^{th}$. This means that by multiplying $\mathbf{A}$ to the data at one time step, it returns the prediction of data at the next time step. With this, we can write

$$\begin{aligned} X_1^{\mu-1} &= \begin{bmatrix} x_1 & x_2 & x_3 & ..... & x_{\mu-1} \end{bmatrix} \\ &= \begin{bmatrix} x_1 & \mathbf{A}x_1 & \mathbf{A}^2x_1 & ..... & \mathbf{A}^{\mu-2}x_1 \end{bmatrix} \end{aligned}$$

Then we have

$$X_2^{\mu} = \mathbf{A}X_1^{\mu-1} + \mathbf{r}e_{\mu-1}^T$$

where $e_{\mu-1}^T$ is a vector of zeros but 1 at the μ^{th} element. $\mathbf{r}$ is the residual. Since SVD gives us that $X_1^{\mu-1} = U\Sigma V^*$, the previous equation becomes

$$X_2^{\mu} = AU\Sigma V^* + \mathbf{r}e_{\mu-1}^T$$

Because $\mathbf{r}$ is orthogonal to U, so $U^*\mathbf{r} = 0$, and the equation becomes

$$U^*AU = U^*X_2^{\mu}V\Sigma^{-1} = \widetilde{S}$$

Now, $\widetilde{S}$ and A are similar, meaning they will have the same eigenvalues, when

$$\widetilde{S}y_k = \mu_k y_k$$

where k is the rank that is ideally very small comparing to N and M, then the eigenvalues of A, called the DMD modes are given as

$$\psi_k = Uy_k.$$

The eigen-basis

$$X_{DMD}(t) = \sum_{k=1}^{k} b_k\psi_k e^{\omega_k t},$$

where b is the initial amplitude of each mode which is $b = \Psi^{\dagger}x_1$ and $\omega_k = log(\mu_k)/\Delta t$. Since we are extracting the foreground, we would define $omega_p$ where $p \in \{1, 2, ...l\}$, and

$$X_{DMD} = b_p\psi_p e^{\omega_p t} + \sum_{j\neq p} b_j\psi_j e^{\omega_j t}.$$

The first term is called DMD's approximate low-rank reconstruction, the second term is DMD's approximate sparse reconstruction. They gives the foreground videos and the background video respectively. Lastly, it is worth noticing that

$$X = X_{DMD}^{Low-Rank} + X_{DMD}^{Sparse}$$

3 Algorithm Implementation and Development

1. Load the video data into MATLAB, record the number of frames, the number of pixels in each frame and the duration of the video.

2. Convert the data into grayscale, and reshape the 4-D data so that the columns in matrix X have information about each frame.

3. Perform the Dynamic Mode Decomposition.

   (a) Define $X_1^{\mu-1}$ and X_2^{μ}, run the `svd()` function on $X_1^{\mu-1}$, and choose the value k for the low-rank approximation.

   (b) Compute $\widetilde{S}$ and get its eigenvalues and eigenvectors. Compute frequency ω and associated Ψ.

4. Calculate the low-rank reconstruction of X, that is the background. By subtracting the low-rank matrix from the original matrix, we get the sparse reconstruction, which is the foreground. We noticed there are negative values in the sparse matrix. In order to have a better image visualization later, we add a value to fix the magnitude.

5. Reshape the low-rank reconstruction and adjusted sparse reconstruction to the original size of the frames.

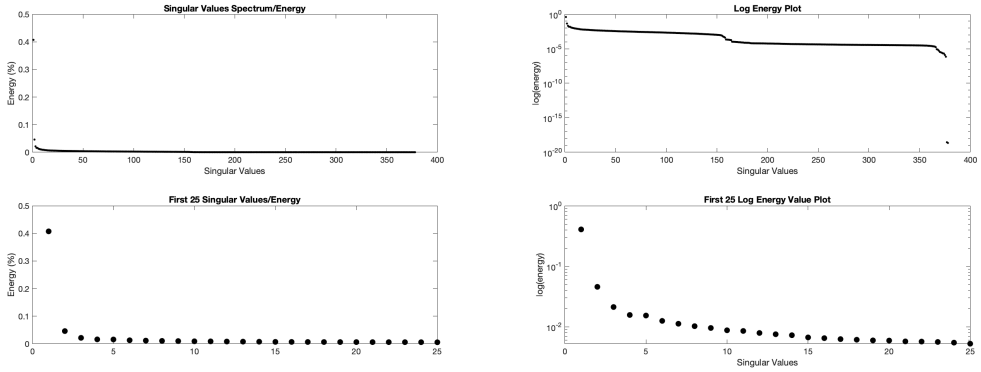

Figure 3: Singular Value Spectrum and Enlarged Spectrum for Video 1

4 Computational Results

4.1 Video 1

After loading the video 1 and setting up for DMD, we plot the singular values as energy percentage as in Figure 3. Top left is the energy for all 378 singular values, we can see there are only the first few contributed, and the rest are very close to zero. So we zoom in on the first 25 points and find that indeed the first singular value contributes significant more than the rest. The second one also captures some energy, but not a lot. For the value of k, we would choose 2 for low-rank approximation and background reconstruction.

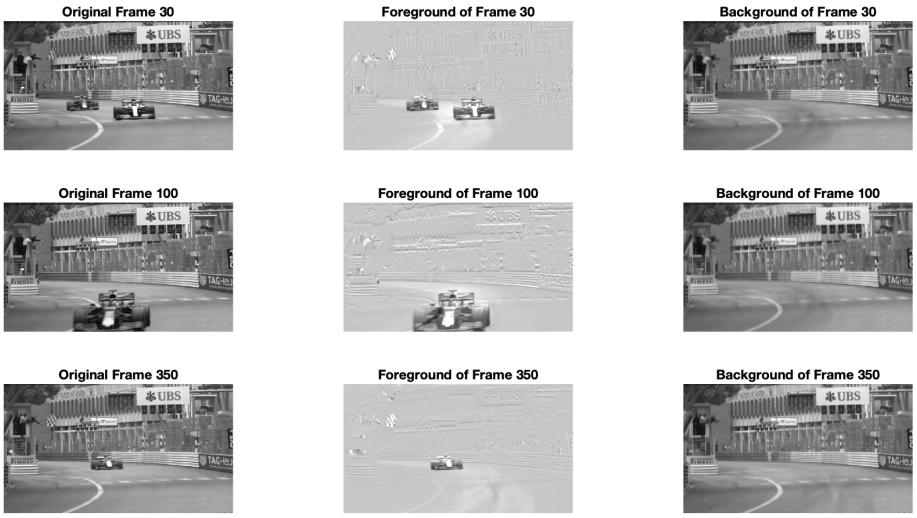

Figure 4: Selected Computational Results for Video 1 with Original, Foreground and Background of Frames 30, 100, 350

Figure 4 shows the DMD results for selected frames and have the original frames on the left for comparison. originally in frame number 30, we can see there are two cars on the track. On its right is the foreground reconstructed by DMD algorithm. It shows that in this frame, the foreground consists of only two racing cars. The rest of the image are trivial. This is confirmed by the background reconstruction image, as there are nothing on the track, but the buildings and the track are clearly reconstructed. The algorithm has successfully separated the foreground and the background. We can imagine if we add the foreground on top of the background, we get the original image, which is what we expected. Middle and bottom row of Figure 4 gives similar conclusion.

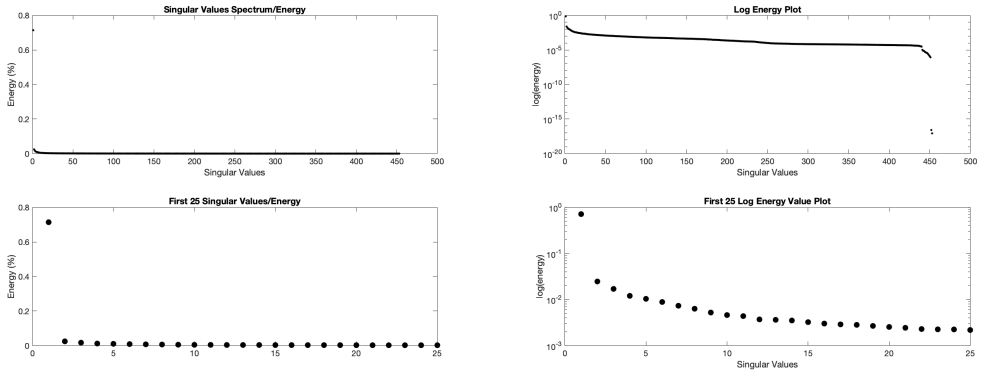

Figure 5: Singular Value Spectrum and Enlarged Spectrum for Video 2

4.2 Video 2

For video 2, we perform the same thing as for video 1. This time, the spectrum (Figure 5) shows a even more dominant feature in the first singular value. The first singular value captures more than 70% of the variance in the data matrix, leaving the rest 451 singular values very close to zero. In the zoomed plots, the dominance is clear and distinct. So we can just use $k = 1$ for our reconstruction.

The results are shown in Figure 6. Because the skier is very small in the original video, he/she is not easy to locate in these plots. The person is most obvious in the foreground for frame 100. The dark dot in the middle of the image is the skier. His movement caused the floating snow around him, which is also captured as foreground because it has certain movement. When comparing with background for frame 100 with the original one, we can see that the skier which is above the right pine tree disappears in the background. For frame 300, the skier is in between the rocks on the bottom left corner, and his body is not in the background image. But in the foreground, we can clearly see a person has bent his body on skis and has poles in his hands. Combining the results from these frames, we are confident about the success in using DMD algorithm for separating fore- and background.

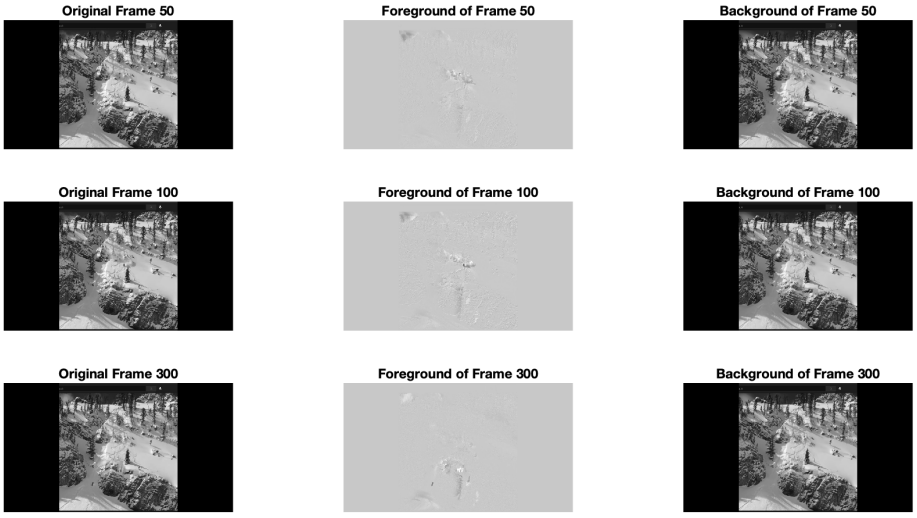

Figure 6: Selected Computational Results for Video 2 with Original, Foreground and Background of Frames 50, 100, 300

5 Summary and Conclusions

From the computational results for video 1 and 2, we have seen the power of Dynamic Mode Decomposition in separating foreground and background. The task is very successful. But this results are from videos with unchanged background. So it is possible for DMD to not have a satisfactory result for more unstable videos, like with lots of noise, or changed angle while filming, etc. We can explore it and experiment on our own later.

Appendix A MATLAB Functions

- `OBJ = VideoReader(FILENAME)` can read in video data and constructs a media reader object, `OBJ`.
- `I = rgb2gray(RGB)` converts RGB image to gray scale.
- `I = uint8(X)` converts `X` to unsigned 8-bit integer. The values of a uint8 range from 0 to 255.
- `[U,S,V] = svd(X)` produces a diagonal matrix S with non-negative diagonal elements in decreasing order, and unitary matrices `U` and `V` so that $X = U * S * V'$.
- `E = eig(A)` produces a column vector `E` containing the eigenvalues of matrix `A`.
- `y = reshape(X, M, N, P)` returns a 3-D array with elements in `X` but has the size M*N*P.
- `diag(X)` returns the diagonal of matrix `X`.

Appendix B MATLAB Code

```
%% first video
clear all; close all; clc

video1 = VideoReader('monte_carlo_low.mp4');
vid1 = read(video1);

Nframes = size(vid1);
Nframes = Nframes(4);
duration = video1.Duration;
height = video1.height;
width = video1.width;

frames = zeros(height*width, Nframes);
for i = 1:Nframes
    f = rgb2gray(vid1(:, :, :, i));
    frames(:, i) = f(:);
end

X = reshape(frames(:, 30), height, width);
imshow(uint8(X))
saveas(gcf, 'orig1.png')
clf

% DMD
X = frames;

X1 = X(:, 1:end-1);
X2 = X(:, 2:end);
dt = duration / Nframes;

[U, S, V] = svd(X1, 'econ');
```

```
%% Singular value plots
sig = diag(S)/sum(diag(S));
subplot(2,2,1), plot(sig,'k.')
title('Singular Values Spectrum/Energy')
xlabel('Singular Values')
ylabel('Energy (%)')
subplot(2,2,2), semilogy(sig,'k.')
ylabel('log(energy)')
xlabel('Singular Values')
title('Log Energy Plot')

subplot(2,2,3), plot(sig(1:25),'k.', 'MarkerSize', 20)
title('First 25 Singular Values/Energy')
ylabel('Energy (%)')
subplot(2,2,4), semilogy(sig(1:25),'k.', 'MarkerSize', 20)
ylabel('log(energy)')
xlabel('Singular Values')
title('First 25 Log Energy Value Plot')

set(gcf,'position',[200, 500, 1500,500])
saveas(gcf, 'SV.png')
clf
%%
k = 2;
U_l = U(:, 1:k);
S_l = S(1:k, 1:k);
V_l = V(:, 1:k);
Stilde = U_l' * X2 * V_l / S_l;
[eV, D] = eig(Stilde);
mu = diag(D);
omega = log(mu)/dt;
Phi = U_l*eV;

t = linspace(0,duration,Nframes+1);
t = t(1:(end-1));
y0 = Phi\X1(:,1);
u_modes = zeros(k,Nframes-1);
for iter = 1:(length(t)-1)
    u_modes(:,iter) = y0.*exp(omega*t(iter));
end
u_dmd = Phi*u_modes;

low_rank = uint8(u_dmd);
sparse = X(:,1:Nframes-1) - u_dmd;

min(min(sparse));
% -201.3870
sparse = uint8(X(:,1:Nframes-1) - u_dmd + 200);

background = low_rank;
foreground = sparse;
```

```
vidbackg = zeros(height, width, Nframes-1);
vidforeg = zeros(height, width, Nframes-1);

for i = 1: (Nframes - 1)
    vidbackg(:, :, i) = reshape(background(:, i), height, width);
    vidforeg(:, :, i) = reshape(foreground(:, i), height, width);
end

%% visualization
i = 30;
subplot(3, 3, 1)
imshow(uint8(reshape(X(:, i), height, width)))
title('Original Frame 30')
subplot(3, 3, 2)
imshow(reshape(foreground(:, i), height, width))
title('Foreground of Frame 30')
subplot(3, 3, 3)
imshow(reshape(background(:, i), height, width))
title('Background of Frame 30')

i = 100;
subplot(3, 3, 4)
imshow(uint8(reshape(X(:, i), height, width)))
title('Original Frame 100')
subplot(3, 3, 5)
imshow(reshape(foreground(:, i), height, width))
title('Foreground of Frame 100')
subplot(3, 3, 6)
imshow(reshape(background(:, i), height, width))
title('Background of Frame 100')

i = 350;
subplot(3, 3, 7)
imshow(uint8(reshape(X(:, i), height, width)))
title('Original Frame 350')
subplot(3, 3, 8)
imshow(reshape(foreground(:, i), height, width))
title('Foreground of Frame 350')
subplot(3, 3, 9)
imshow(reshape(background(:, i), height, width))
title('Background of Frame 350')

set(gcf,'position',[200, 500, 1000,500])
saveas(gcf, 'Monte.png')
clf
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% second video
clear all; close all; clc
```

```
video1 = VideoReader('ski_drop_low.mp4');
vid1 = read(video1);

Nframes = size(vid1);
Nframes = Nframes(4);
duration = video1.Duration;
height = video1.height;
width = video1.width;

frames = zeros(height*width, Nframes);
for i = 1:Nframes
    f = rgb2gray(vid1(:, :, :, i));
    frames(:, i) = f(:);
end

X = reshape(frames(:, 50), height, width);
imshow(uint8(X))
saveas(gcf, 'orig2.png')
clf
%% DMD
X = frames;

X1 = X(:, 1:end-1);
X2 = X(:, 2:end);
dt = duration / Nframes;

[U, S, V] = svd(X1, 'econ');

%% Singular value plots
sig = diag(S)/sum(diag(S));
subplot(2,2,1), plot(sig,'k.')
title('Singular Values Spectrum/Energy')
xlabel('Singular Values')
ylabel('Energy (%)')
subplot(2,2,2), semilogy(sig,'k.')
ylabel('log(energy)')
xlabel('Singular Values')
title('Log Energy Plot')

subplot(2,2,3), plot(sig(1:25),'k.', 'MarkerSize', 20)
title('First 25 Singular Values/Energy')
ylabel('Energy (%)')
subplot(2,2,4), semilogy(sig(1:25),'k.', 'MarkerSize', 20)
ylabel('log(energy)')
xlabel('Singular Values')
title('First 25 Log Energy Value Plot')

set(gcf,'position',[200, 500, 1500,500])
saveas(gcf, 'SV2.png')
clf
%%
```

```
k = 1;
U_l = U(:, 1:k);
S_l = S(1:k, 1:k);
V_l = V(:, 1:k);
Stilde = U_l' * X2 * V_l / S_l;
[eV, D] = eig(Stilde);
mu = diag(D);
omega = log(mu)/dt;
Phi = U_l*eV;

t = linspace(0,duration,Nframes+1);
t = t(1:(end-1));
y0 = Phi\X1(:,1);
u_modes = zeros(k,Nframes-1);
for iter = 1:(length(t)-1)
    u_modes(:,iter) = y0.*exp(omega*t(iter));
end
u_dmd = Phi*u_modes;

low_rank = uint8(u_dmd);
sparse = X(:,1:Nframes-1) - u_dmd;

min(min(sparse));
% -216.5599
sparse = uint8(X(:,1:Nframes-1) - u_dmd + 200);

background = low_rank;
foreground = sparse;

vidbackg = zeros(height, width, Nframes-1);
vidforeg = zeros(height, width, Nframes-1);

for i = 1: (Nframes - 1)
    vidbackg(:, :, i) = reshape(background(:, i), height, width);
    vidforeg(:, :, i) = reshape(foreground(:, i), height, width);
end

%% visualization
i = 50;
subplot(3, 3, 1)
imshow(uint8(reshape(X(:, i), height, width)))
title('Original Frame 50')
subplot(3, 3, 2)
imshow(reshape(foreground(:, i), height, width))
title('Foreground of Frame 50')
subplot(3, 3, 3)
imshow(reshape(background(:, i), height, width))
title('Background of Frame 50')

i = 100;
subplot(3, 3, 4)
```

```
imshow(uint8(reshape(X(:, i), height, width)))
title('Original Frame 100')
subplot(3, 3, 5)
imshow(reshape(foreground(:, i), height, width))
title('Foreground of Frame 100')
subplot(3, 3, 6)
imshow(reshape(background(:, i), height, width))
title('Background of Frame 100')

i = 300;
subplot(3, 3, 7)
imshow(uint8(reshape(X(:, i), height, width)))
title('Original Frame 300')
subplot(3, 3, 8)
imshow(reshape(foreground(:, i), height, width))
title('Foreground of Frame 300')
subplot(3, 3, 9)
imshow(reshape(background(:, i), height, width))
title('Background of Frame 300')

set(gcf,'position',[200, 500, 1000,500])
saveas(gcf, 'ski.png')
clf
```