AMATH 482 Assignment 5:

Background Subtraction in Video Streams

Yuqing Cui

March 17, 2021

Abstract

In this report, we are given two videos. One of them is cars racing in the Monte Carlo, and the other is a person skiing down a mountain. We are going to apply the dynamic mode decomposition (DMD) on those videos to separate the foreground and the background by analyzing the difference in frequency.

I. Introduction and Overview

Generally, a video is composed of two parts——the foreground and the background. Usually, when analyzing a dynamic system, we would like to focus on the moving object. Therefore, it would be helpful if we could separate the foreground and the background. The dynamic mode decomposition (DMD) is a useful method in capturing the low dimensionality in a given system. It projects the system into several fundamental components representing spatial information. Without assumptions to the system, it could potentially provide more information. In what follows, we are going to perform DMD on the two videos to see how powerful it is on separating the foreground and the background.

II. Theoretical Background

(a) Singular Value Decomposition (SVD)

The singular value decomposition (SVD) reforms an $m \times n$ matrix $\mathbf{A}$ into the following format:

$$\mathbf{A} = \mathbf{U\Sigma V}^*, \tag{1}$$

in which $\mathbf{U}$ is an $m \times m$ unitary matrix that contains principle components. $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix that contains singular values. $\mathbf{V}$ is an $n \times n$ unitary matrix with time information. $\mathbf{U}$ and $\mathbf{V}$ are orthonormal to each other. All diagonal elements of $\mathbf{\Sigma}$ are non-negative and are sorted from the largest (upper-left) to the smallest (lower-right). The number of nonzero singular values is the rank of matrix $\mathbf{A}$. The matrix multiplication will perform rotations and stretches. Besides, SVD enables every matrix to be diagonal with proper bases for the domain and the range.

(b) Dynamic Mode Decomposition (DMD)

The dynamic mode decomposition (DMD) takes data in the following form:

$$\mathbf{X}_1^{M-1} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{M-1}], \tag{1}$$

$$\mathbf{X}_2^{M} = [\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{M}], \tag{2}$$

where m is the number of snapshots of data over evenly spaced time interval. In our case, it's the number of frames of each video. Each frame $\mathbf{x}_j$ contains n spatial data points. Then the DMD approximates the linear, time-independent operator, called the Koopman operator $\mathbf{A}$ mapping the data from time t_j to t_{j+1} such that

$$\mathbf{x}_{j+1} = \mathbf{A}\mathbf{x}_j, \tag{3}$$

Applying it to Equation (1), we have

$$\mathbf{X}_1^{M-1} = [\mathbf{x}_1, \mathbf{A}\mathbf{x}_1, \ldots, \mathbf{A}^{M-2}\mathbf{x}_1], \tag{4}$$

The columns of $\mathbf{X}_1^{M-1}$ form the basis for the Krylov space. Therefore, we can rewrite $\mathbf{X}_2^{M}$ as

$$\mathbf{X}_2^{M} = \mathbf{A}\mathbf{X}_1^{M-1} + \boldsymbol{r}e_{M-1}^{T}, \tag{5}$$

where $\boldsymbol{r}$ is the residual vector and e_{M-1}^{T} is a vector with one at the $(M-1)th$ component and all zeroes everywhere else. Our goal is to figure out the matrix $\mathbf{A}$ using the approach of finding eigenvalues and eigenvectors. Using SVD, as described above, we have

$$\mathbf{X}_1^{M-1} = \mathbf{U\Sigma V}^*, \tag{6}$$

$$\mathbf{X}_2^{M} = \mathbf{AU\Sigma V}^* + \boldsymbol{r}e_{M-1}^{T}, \tag{7}$$

We are going to choose $\mathbf{A}$ in such a way that the columns in $\mathbf{X}_2^{M}$ can be written as linear combinations of the columns of $\mathbf{U}$. It's the same as requiring that they can be written as linear combinations of the POD modes. Since $\boldsymbol{r}$ must be orthogonal to the POD basis, $\mathbf{U}^*\boldsymbol{r} = 0$. Multiplying Equation (7) by $\mathbf{U}^*$, we have

$$\mathbf{U}^*\mathbf{X}_2^{M} = \mathbf{U}^*\mathbf{AU\Sigma V}^*, \tag{8}$$

Then we isolate $\mathbf{U}^*\mathbf{AU}$ by

$$\mathbf{U}^*\mathbf{AU} = \mathbf{U}^*\mathbf{X}_2^{M}\mathbf{V\Sigma}^{-1} = \widetilde{\mathbf{A}}, \tag{9}$$

$\widetilde{\mathbf{A}}$ and $\mathbf{A}$ have the same eigenvalues because they are related by applying a matrix on one side and the inverse on the other. Therefore, we can write $\widetilde{\mathbf{A}}$ in the following form:

$$\widetilde{\mathbf{A}}\mathbf{y}_k = \mu_k \mathbf{y}_k, \tag{10}$$

where $\mathbf{y}_k$ is the eigenvector of $\widetilde{\mathbf{A}}$. Then we have our DMD modes φ_k, which are the eigenvectors of $\mathbf{A}$, such that

$$\varphi_k = \mathbf{U}\mathbf{y}_k, \tag{11}$$

After expanding our eigenbasis, we get the DMD solution

$$\mathbf{x}_{DMD}(t) = \sum_{k=1}^{K} b_k \varphi_k e^{\omega_k t} = \boldsymbol{\Phi} diag(e^{\omega_k t})\mathbf{b}, \tag{12}$$

where K is the rank of $\mathbf{X}_1^{M-1}$, b_k are the initial amplitude of each mode, and $\boldsymbol{\Phi}$ is the matrix having ψ_k as its columns.

Suppose ω_p, where $p \in \{1,2,\dots,l\}$ satisfies $||\omega_p|| = 0$, and that $||\omega_j|| \; \forall \, j \neq p$ is bounded away from zero, we have

$$\mathbf{X}_{DMD} = b_p \varphi_p e^{\omega_p t} + \sum_{j \neq p} b_j \varphi_j e^{\omega_j t}, \tag{13}$$

where the part before the plus sign represents the background video and the other part represents the foreground video. However, each term of the DMD reconstruction is complex and would cause a problem when separating the DMD terms unto approximate low-rank and sparse reconstructions. We have the following steps to handle those complex elements. Suppose we have the low-rank matrix as

$$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} = b_p \varphi_p e^{\omega_p t}, \tag{14}$$

Since it should be true that

$$\mathbf{X} = \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} + \mathbf{X}_{\text{DMD}}^{\text{Sparse}}, \tag{15}$$

we must have our sparse matrix as

$$\mathbf{X}_{\text{DMD}}^{\text{Sparse}} = \sum_{j \neq p} b_j \varphi_j e^{\omega_j t} = \mathbf{X} - |\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}}|, \tag{16}$$

We generate a new $n \times m$ matrix $\mathbf{R}$ containing residual negative values in $\mathbf{X}_{\text{DMD}}^{\text{Sparse}}$, and subtract if from the original sparse matrix to get the final foreground while maintaining the property in Equation (15).

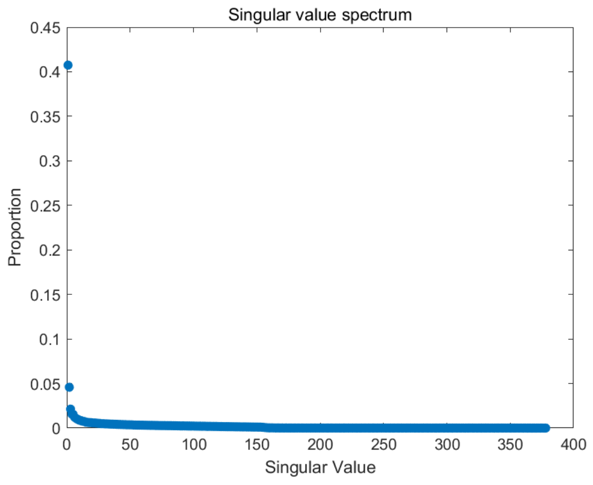
III. Algorithm Implementation and Development

The process is very similar for the two video. We first loaded the information from the video and decided the total time and the time interval. We then reshape each frame into grayscale for DMD. After separating the data into two matrix $\mathbf{X_1}$, $\mathbf{X_2}$, where $\mathbf{X_1}$ contains the first frame to the second to the last frame and $\mathbf{X_2}$ contains the second frame to the last frame, we first perform SVD on $\mathbf{X_1}$ and plotted the singular value spectrum. We chose the rank that takes most part of the energy of the system for DMD based on the spectrum.

We then modified our $\mathbf{U}$, $\boldsymbol{\Sigma}$, and $\mathbf{V}$ matrices to satisfy the size requirement that $\mathbf{U} \in \mathbb{C}^{N*K}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{K*K}$, and $\mathbf{V} \in \mathbb{C}^{M-1*K}$. We obtained the eigenvalues and eigenvectors from the matrix calculated in Equation (9). We then calculated our low-rank and sparse DMD matrix following Equation (11)-(16). However, our sparse matrix may contain negative complex elements. To solve the problem, we identified those elements and put them into matrix **R**. Lastly, we subtracted those values from the sparse matrix to get our final foreground matrix. We then plotted the original video, the background, and the foreground respectively the 200th frame. For clear representation, we also did another brighter picture by adding 200 to the value in grayscale.

IV. Computational Results

Figure 1. shows the singular value spectrum of the *monte carlo* video. We can see that the first two components are essentially important and the percentage seems zero after the 25th singular value. Therefore, we chose 25 to be our rank for this case. On the other hand, only the first component is dominant for the *ski drop* video. The percentage seems zero after the 10th singular value, so we chose 10 to be our rank.



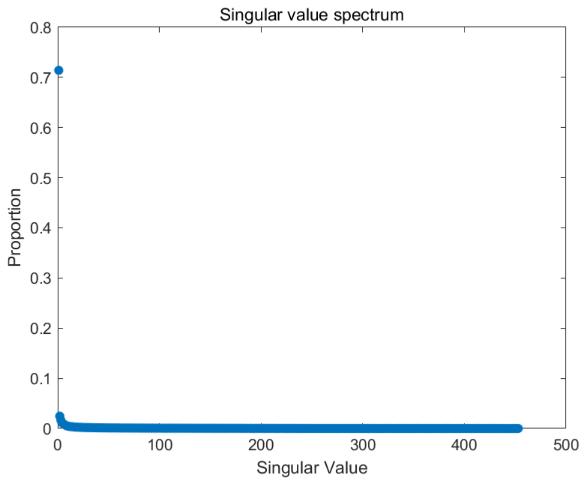


Figure 1. The singular value spectrum of *monte_carlo_low.mp4* (left)

Figure 2. The singular value spectrum of *ski_drop_low.mp4* (right)

(a) Monte Carlo

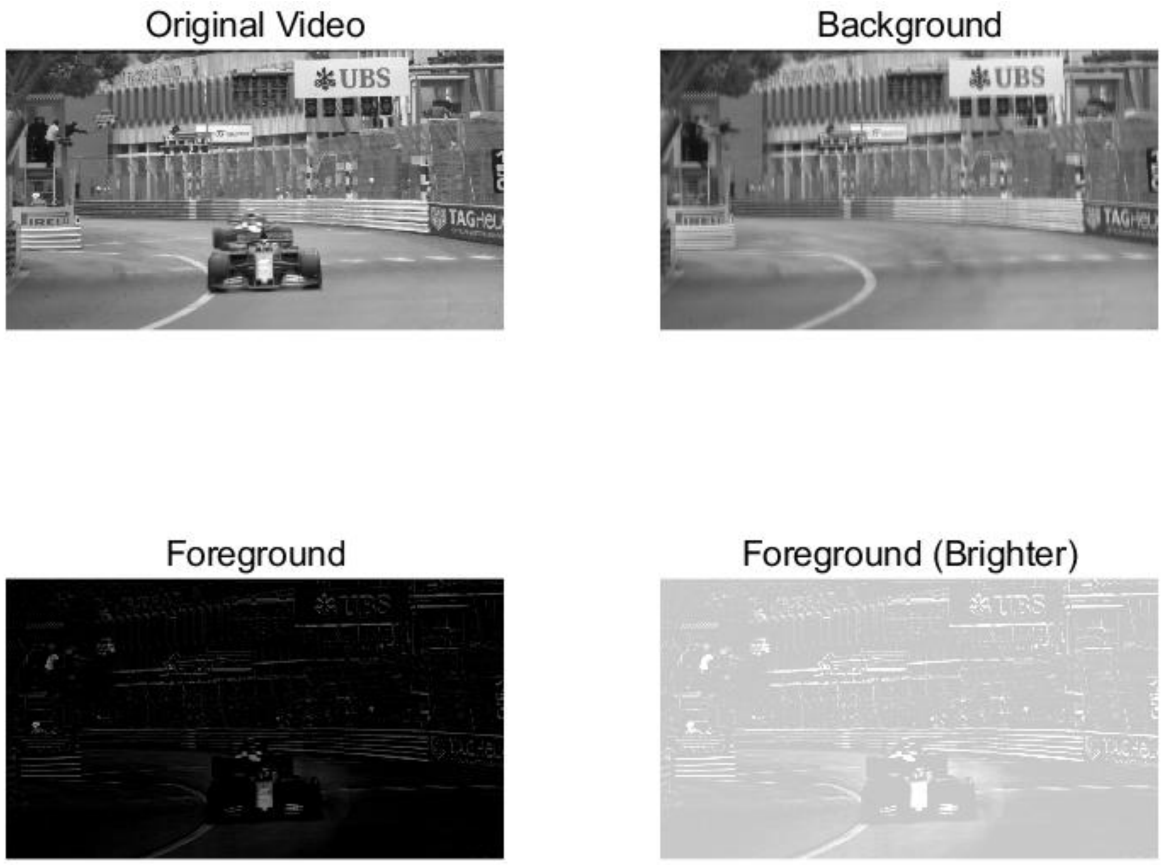


Figure 3. The background and the foreground at the 200th Frame

In the video, cars are racing and there are also many people captured. As shown in **Figure 3.**, the DMD is overall satisfying because we can see that it successfully separate the cars and the backgrounds. It also counts some parts of backgrounds into the foreground due to slight shaking of the camera.

(b) Ski Drop

In this video, a skier is skiing down a mountain. As circled in red in **Figure 4**., we can see that DMD successfully separate the person from the background. However, it’s really hard to see in the foreground even after processing the picture. Also, there are some white dots in the foreground which might capture the drifting snow due to wind or the skier. The performance of DMD is not as good as that in the *monte carlo* video. The small size of the skier in the video might be a reason for the problem.

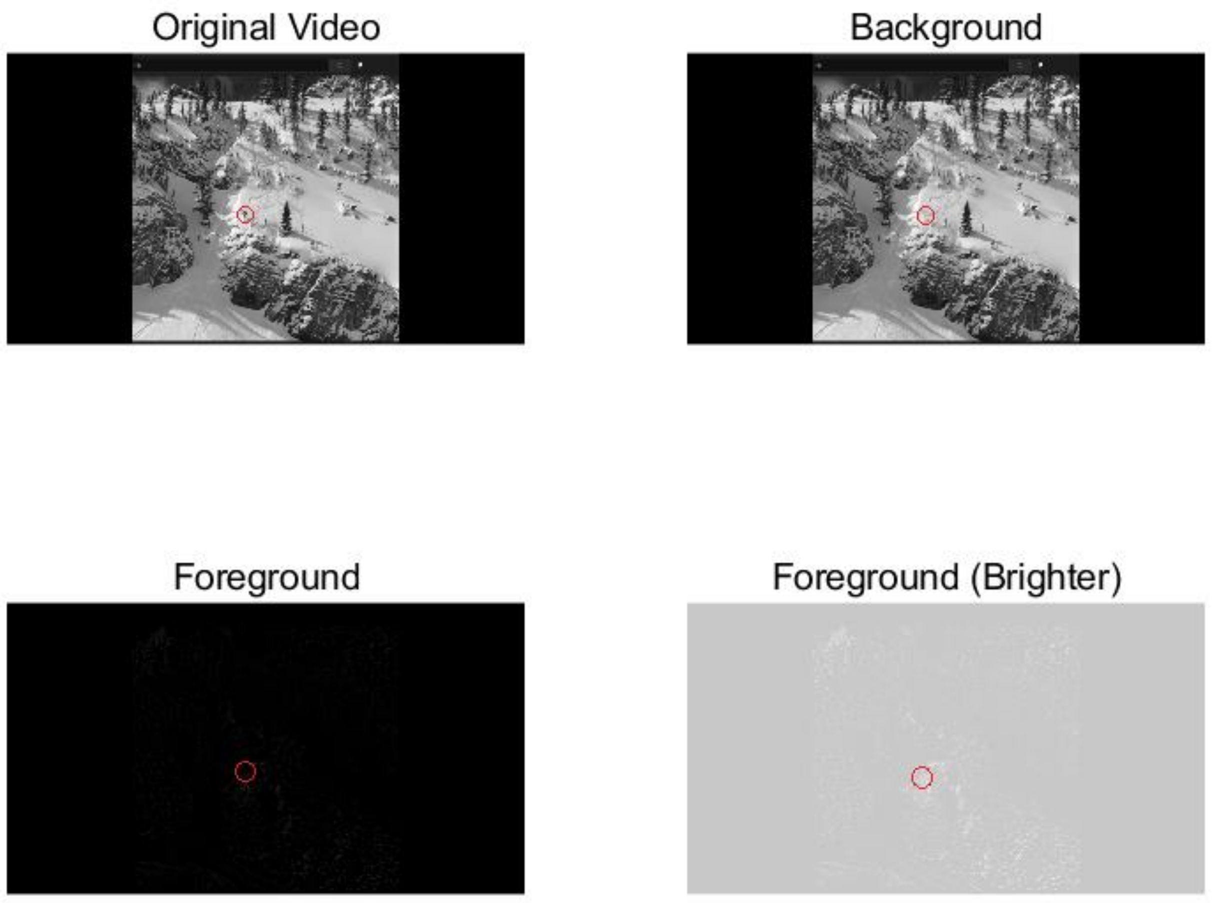


Figure 4. The background and the foreground at the 200th Frame

V. Summary and Conclusions

We performed DMD on the two videos and successfully separate the foreground and the background by calculating the sparse matrix and the low-rank matrix respectively. The DMD did extremely well on the *monte carlo* video, while less ideally on the *ski drop* video due to the small size of the moving object. Overall, it's still a useful approach in processing images and analyzing dynamics.

Appendix A. MATLAB functions[1]

`v = VideoReader(filename)` : creates object `v` to read video data from the file named `filename`. We used it to load our data.

`video = read(v,index)` : reads only the frames specified by `index`. We used it to extract the frame for further processing.

`I = rgb2gray(RGB)` : converts the truecolor image `RGB` to the grayscale image `I`. We used it to turn our video into grayscale.

`B = reshape(A,sz)` : reshapes `A` using the size vector, `sz`, to define `size(B)`. We used it to reshape each frame of our videos with a specific size.

`[u,s,v] = svd(A,'econ')`: Produces an economy-size decomposition of matrix `A`, such that `A = U*S*V'`. We used it to perform SVD.

`D = diag(v)` : returns a square diagonal matrix with the elements of vector `v` on the main diagonal. We used it to obtain the diagonal matrix to plot the singular value spectrum.

`[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`. We used it to find the eigenvalue in order to create modes for DMD.

`Y = abs(X)`: returns the absolute value of each element in array `X`. If `X` is complex, `abs(X)` returns the complex magnitude. We used it in creating the sparse matrix.

`Y = uint8(X)`: converts the values in `X` to type `uint8`. We used it to convert our data in order to use `imshow`.

`imshow(I)`: displays the grayscale image `I` in a figure. We used it to show our frames.

[1] Basic function descriptions from MathWorks Help Center.

Appendix B. MATLAB codes

```
%% Monte carlo
clear all; close all; clc;
monte = VideoReader('monte_carlo_low.mp4');

dt = 1/monte.Framerate;
t = 0:dt:monte.Duration;
video = zeros(monte.Width*monte.Height, monte.NumberOfFrames);
for i=1:monte.NumberOfFrames
    frame = rgb2gray(read(monte,i));
    video(:,i)=double(reshape(frame,monte.Width*monte.Height,1));
end

% DMD
X1 = video(:,1:end-1);
X2 = video(:,2:end);
[u,s,v] = svd(X1,'econ');
lamda = diag(s);

plot(lamda/sum(lamda),'.','Markersize',20);
xlabel('Singular Value'),ylabel('Proportion');
title('Singular value spectrum');
saveas(gcf,'monte spec.png');

r = 25;
u_r = u(:,1:r);
s_r = s(1:r,1:r);
v_r = v(:,1:r);
Atilde = u_r'*X2*v_r/s_r;
[W,D] = eig(Atilde);
Phi = X2*v_r/s_r*W;
mu = diag(D);
omega = log(mu)/dt;

% low-rank and sparse
y0 = Phi\X1(:,1);
X_modes = zeros(length(y0),length(t)-1);
for iter = 1:(length(t)-1)
    X_modes(:,iter) = y0.*exp(omega*t(iter));
end
```

```
X_dmd = Phi*X_modes;

X_s = X1-abs(X_dmd);
R = X_s.*(X_s<0);
X_sparse = X_s-R;
X_sparse_2 = X_sparse+200;

% picture
original = reshape(uint8(video(:,200)), monte.Height,
monte.Width);
background = reshape(uint8(X_dmd(:,200)), monte.Height,
monte.Width);
foreground = reshape(uint8(X_sparse(:,200)), monte.Height,
monte.Width);
foreground2 = reshape(uint8(X_sparse_2(:,200)), monte.Height,
monte.Width);

figure()
subplot(2,2,1);
imshow(original);
title('Original Video');

subplot(2,2,2);
imshow(background);
title('Background');

subplot(2,2,3);
imshow(foreground);
title('Foreground');

subplot(2,2,4);
imshow(foreground2);
title('Foreground (Brighter)');
saveas(gcf,'monte.png');

%% Ski drop
clear all; close all; clc;
ski = VideoReader('ski_drop_low.mp4');

dt = 1/ski.Framerate;
t = 0:dt:ski.Duration;
```

```
video = zeros(ski.Width*ski.Height, ski.NumberOfFrames);
for i=1:ski.NumberOfFrames
    frame = rgb2gray(read(ski,i));
    video(:,i)=double(reshape(frame,ski.Width*ski.Height,1));
end

% DMD
X1 = video(:,1:end-1);
X2 = video(:,2:end);
[u,s,v] = svd(X1,'econ');
lamda = diag(s);

plot(lamda/sum(lamda),'.','Markersize',20);
xlabel('Singular Value'),ylabel('Proportion');
title('Singular value spectrum');
saveas(gcf,'ski spec.png');

r = 10;
u_r = u(:,1:r);
s_r = s(1:r,1:r);
v_r = v(:,1:r);
Atilde = u_r'*X2*v_r/s_r;
[W,D] = eig(Atilde);
Phi = X2*v_r/s_r*W;
mu = diag(D);
omega = log(mu)/dt;

% low-rank and sparse
y0 = Phi\X1(:,1);
X_modes = zeros(length(y0),length(t)-1);
for iter = 1:(length(t)-1)
    X_modes(:,iter) = y0.*exp(omega*t(iter));
end
X_dmd = Phi*X_modes;

X_s = X1-abs(X_dmd);
R = X_s.*(X_s<0);
X_sparse = X_s-R;
X_sparse_2 = X_sparse+200;

% picture
```

```
original = reshape(uint8(video(:,200)), ski.Height, ski.Width);
background = reshape(uint8(X_dmd(:,200)), ski.Height,
ski.Width);
foreground = reshape(uint8(X_sparse(:,200)), ski.Height,
ski.Width);
foreground2 = reshape(uint8(X_sparse_2(:,200)), ski.Height,
ski.Width);

figure()
subplot(2,2,1);
imshow(original);
title('Original Video');

subplot(2,2,2);
imshow(background);
title('Background');

subplot(2,2,3);
imshow(foreground);
title('Foreground');

subplot(2,2,4);
imshow(foreground2);
title('Foreground (Brighter)');
saveas(gcf,'ski.png');
```