Amath 482 Homework 5

Oscar Zhang

Mar.17th 2021

Abstract

In this assignment, we are going to apply the DMD method to separate two given videos into four new videos – two for foreground and two for background. We also check the energy captured by our single values and how good our foreground and background videos are based on our omega values.

1. Introduction

The two videos used in this assignment are provided by Professor Jason Bramburger. The basic idea for this assignment is to first convert our given videos into matrix form. Then we need to apply DMD to the matrix to convert it into a low-rank approximate reconstruction of our given. The new matrix will be the background of our videos. Then we subtract it from our original data to get one new matrix, which represents the sparse reconstruction – the foreground video. To make our foreground and background videos clearer and more accurate, we will subtract residual matrix to remove the negative pixels.

2. Theoretical Background

DMD, known as Dynamic Mode Decomposition, aims to take advantage of low-dimensionality in experimental data without having to rely on a given set of governing equations.

Suppose we have data that are both spatio-temporal. We define that:

N = number of spatial points saved per unit time snapshot

M = number of snapshots taken

Then we can get the snapshots matrix:

$$U(x, t_m) = \begin{bmatrix} U(x_1, t_m) \\ \vdots \\ U(x_n, t_m) \end{bmatrix}$$

Where m = 1, 2, …, M – 1and t is regularly spaced from each other.

Then we can get two new matrix based on our snapshots matrix:

$$X = [U(x, t_1)\ U(x, t_2)\ \dots\ U(x, t_M)]$$

$$X_j^k = [\big(x, t_j\big)\ U\big(x, t_{j+1}\big)\ \dots\ U(x, t_k)]$$

Then we need one Koopman operator A, a linear, time-independent operator that

$$X_{j+1} = AX_j$$

Then suppose we have a matrix of given data:

$$X_1^{M-1} = [x_1\ x_2\ \dots\ x_{M-1}]$$

And the above matrix can be presented by the following with the Koopman operator A:

$$X_1^{M-1} = [x_1\ Ax_1 \ldots A^{M-2}x_1]$$

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

Where $e_{M-1}$is a vector with all 0s and 1 at the component (M - 1) and r is residual vector to count for the error. So we need to find out A. Applying SVD to X_1^{M-1}, we get that :

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

Then multiplying U^*on both sides:

$$U^*X_2^M = U^*AU\Sigma V^*$$

$$U^*AU = U^*X_2^M V\Sigma^{-1} =: \tilde{S}$$

Since r is orthogonal to POD basis, $U * r = 0$

Then we can have that if y is an eigenvector of $\tilde{S}$, then Uy is an eigenvector of A, which leads to:

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

Then given the eigenvector of A, the DMD modes will be:

$$\psi_k = Uy_k$$

Therefore,

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

3. Algorithm Implementation and Development

In order to separate our videos into foreground and background, we have to make slight changes on the DMD formular above.

Assume that $\omega_p$where $p \in \{1, 2, \ldots, l\}$ satisfies that $\|\omega_p\| \approx 0$ and $\|\omega_l\| \forall j \neq p$ is bounded away from zero, then we can rewrite the DMD into the following:

$$X_{DMD} = b_p \varphi_p e^{\omega_p t} + \sum_{j \neq p} b_j \varphi_j e^{\omega_j t}$$

Where the first part is the background video and the first part is the foreground video. To approximate the low-rank reconstruction, we have that:

$$X_{DMD}^{low-rank} = b_p \varphi_p e^{\omega_p t}$$

And since:

$$X = X_{DMD}^{low-rank} + X_{DMD}^{sparse}$$

It’s easy to get:

$$X_{DMD}^{sparse} = \sum_{j \neq p} b_j \varphi_j e^{\omega_j t} = X - \left|X_{DMD}^{low-rank}\right|$$

However, in this way, we might get the sparse DMD some negative values. In order to count for that, we introduce a residual matrix R with dimensions $n \times m$. Therefore, our final equations will be:

$$X_{DMD}^{low-rank} = \left|X_{DMD}^{low-rank}\right| + R$$

$$X_{DMD}^{sparse} = X_{DMD}^{sparse} - R$$

After figuring out the formula for the DMD, we can construct our videos by matlab. We first import the video to matlab by X = VideoReader('filename'). Then I build up empty matrix and store the file data after covert the video into matrix. Then I set up the width and height for the video and use 1 / X.FrameRate to calculate the time between two frames. Also we set the video into two separate matrix as the X_1^{M-1} and X_2^M above.

After the basic setup, I apply the SVD to the first matrix and we can plot the single value energy plot. After applying the SVD, we can construct the Koopman operator A based on the U, S, and V we calculated. Then we find the eigenvalue and eigenvector of A and then we can reconstruct A based on the calculated eigenvalue and eigenvector. Then we get our DMD mode. Based on the $X_{DMD}^{low-rank}$ and X_{DMD}^{sparse} formulas above, we are able to get the foreground and background videos for our given videos.

4. Computational Results

For video "ski_drop"

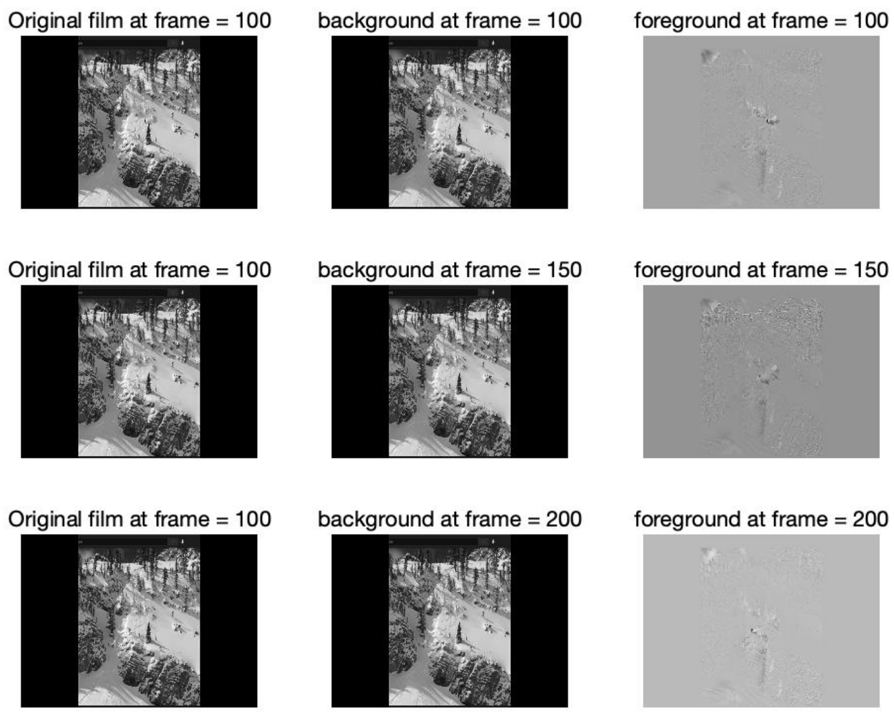

Original film at frame = 100
background at frame = 100
foreground at frame = 100
Original film at frame = 100
background at frame = 150
foreground at frame = 150
Original film at frame = 100
background at frame = 200
foreground at frame = 200

Figure 1: frames for ski_drop video

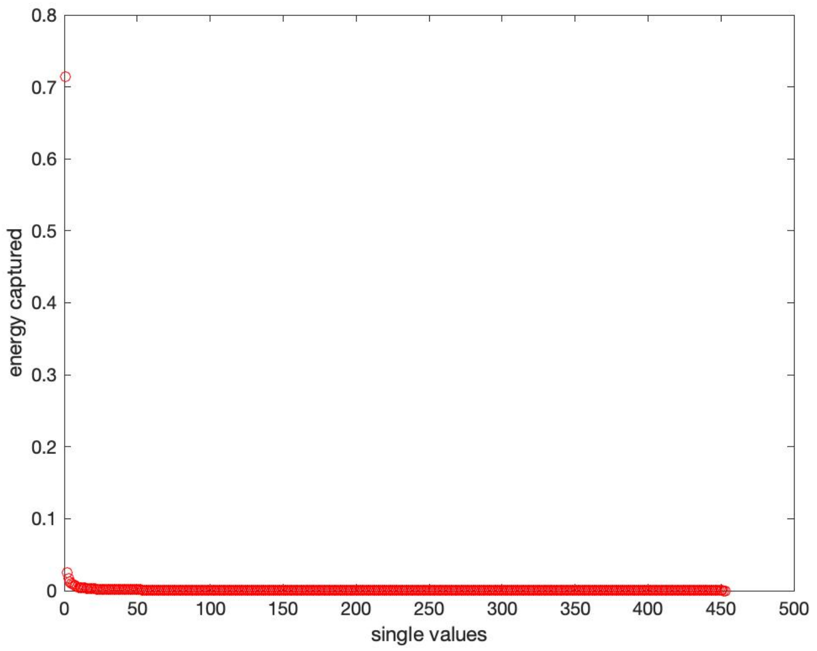
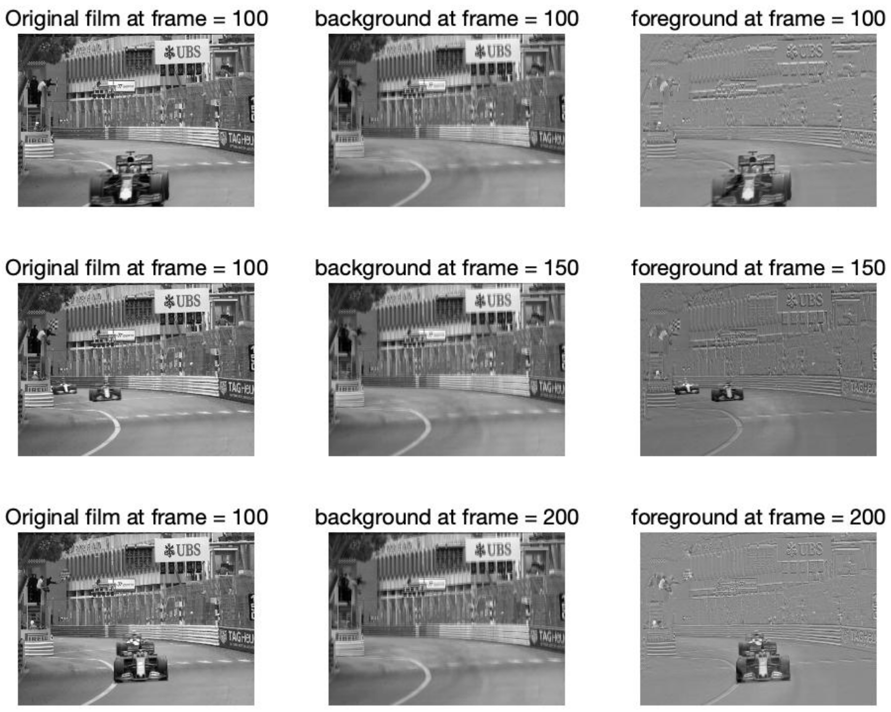


Figure2: energies of single values for video “ski_drop”

For video “monte_carlo”



Original film at frame = 100
background at frame = 100
foreground at frame = 100
Original film at frame = 100
background at frame = 150
foreground at frame = 150
Original film at frame = 100
background at frame = 200
foreground at frame = 200

Figure3: frames for monte_carlo video

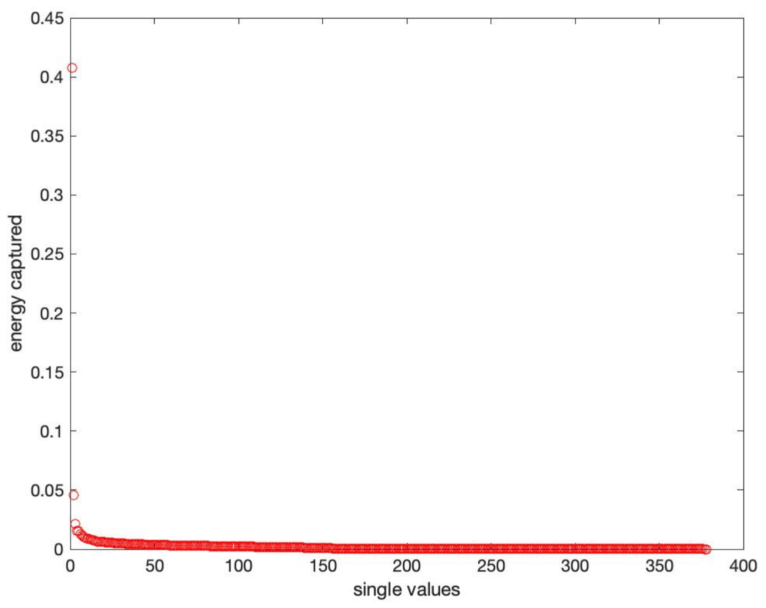


Figure4: energies of single values for video "monte_carlo"

For both videos, the results after the DMD are acceptable – we can easily see the foreground and background of the video. For the video "ski_drop", the foreground is a little bit hard to see since the athlete is quite small and in fact it is only a tiny dot on the frames. However, in total, the foreground of the moving athlete and cars and the background, the static background have been separated clearly from each other.

5.Conclusion

For this assignment, we have worked on separating the foreground and background of one video via the method Dynamic Mode Decomposition. Working on the tow given videos, my results are good – both are good for large objects as the foreground, the race car, and the small objects as the foreground, the ski athlete. DMD is quite a useful method since it can separate one matrix into two related to each other - $X_{DMD}^{low-rank}$ and X_{DMD}^{sparse}, one for the background, the static objects, one for the foreground, the moving objects.

Matlab Commands

X = VideoReader(x): import the video(x)
[U, S, V] = svd(x): apply SVD to the matrix and returns 3 matrices, U, S, and V
[W, D] = eig(x): find the eigenvalue and eigenvector of the matrix
Abs(x): take the absolute value of x
Double(x): convert the x into double type
Diag(x): return the diagonal values of x as a column vector

Matlab Codes

Since there are two videos and we only need to change the filename for the video, so I just post the code for “ski_drop” video.

```
%set up for and find the energy of single values
X = VideoReader('ski_drop_low.mp4');
video = [];
while hasFrame(X)
    data = readFrame(X);
    data = rgb2gray(data);
    data = data(:);
    video = [video data];
end
video = double(video);

dt = 1 / X.FrameRate;
height = X.height;
width = X.width;

X1 = video(:, 1:end-1);
X2 = video(:, 2:end);
frame = size(video, 2) - 1;
[U, S, V] = svd(X1, 'econ');
plot(diag(S)/sum(diag(S)), 'ro');
xlabel('single values')
ylabel('energy captured')
%% compute the DMD mode
r = 1;
Ur = U(:, 1:r);
Sr = S(1:r, 1:r);
Vr = V(:, 1:r);

Atilde = Ur'*X2*Vr / Sr;
[W, D] = eig(Atilde);
```

```matlab
Phi = X2*Vr/ Sr*W;
lambda = diag(D);
omega = log(lambda) / dt;
[val, ind] = sort(abs(omega));

omega = omega(ind(1));
%% based on the DMD mode, compute the Sparse DMD and low-rank DMD
t = linspace(0, X.Duration, size(video, 2));
x1 = video(:, 1);
b = Phi\x1;
time_dynamics = zeros(r,length(t));
for iter = 1:length(t)
    temp = (b.*(exp(omega*t(iter))).');
    time_dynamics(:,iter) = sum(temp, 2);
end

X_lowRank = Phi*time_dynamics;
X_sparse = video - abs(X_lowRank);

R = zeros(length(X_sparse), size(X_sparse, 2));

X_fg = X_sparse - R;
X_bg = abs(X_lowRank) + R;

video_bg = reshape(X_bg, [height, width, frame+1]);
video_fg = reshape(X_fg, [height, width, frame+1]);
%% plot results
i = 100;
figure(1)
subplot(3,3,1);
pcolor(flip(reshape(video(:,i), [height, width]))); shading interp;
colormap(gray);axis off;
title('Original film at frame = 100');
subplot(3,3,2);
pcolor(flip(video_bg(:,:,i))); shading interp; colormap(gray);axis
off;
title('background at frame = 100');
subplot(3,3,3);
pcolor(flip(video_fg(:,:,i))); shading interp; colormap(gray);axis
off;
title('foreground at frame = 100');
%%
i = 150;
subplot(3,3,4);
```

```
pcolor(flip(reshape(video(:,i), [height, width]))); shading interp;
colormap(gray);axis off;
title('Original film at frame = 100');
subplot(3,3,5);
pcolor(flip(video_bg(:,:,i))); shading interp; colormap(gray);axis
off;
title('background at frame = 150');
subplot(3,3,6);
pcolor(flip(video_fg(:,:,i))); shading interp; colormap(gray);axis
off;
title('foreground at frame = 150');
%%
i = 200;
subplot(3,3,7);
pcolor(flip(reshape(video(:,i), [height, width]))); shading interp;
colormap(gray);axis off;
title('Original film at frame = 100');
subplot(3,3,8);
pcolor(flip(video_bg(:,:,i))); shading interp; colormap(gray);axis
off;
title('background at frame = 200');
subplot(3,3,9);
pcolor(flip(video_fg(:,:,i))); shading interp; colormap(gray);axis
off;
title('foreground at frame = 200');
```