Assignment 5 Background Subtraction in Video Streams

Ming Lou

Abstract:

This assignment will mainly focus on using Dynamic Mode Decomposition (DMD) method to separate the video stream to both the foreground (sparse) video and a background (low rank) for two video clips. We will use singular value decomposition (SVD) to find low-rank of data.

Section I. Introduction and Overview :

Dynamic Mode Decomposition (DMD) is a method to perform background subtraction. The aim of the method is to take advantage of low-dimensionality in experimental data without having to rely on a given set of governing equations. In this assignment, we are going to perform this DMD method on the two given videos, which are ski and Monte_carlo. The Dynamic Mode Decomposition will create a low-rank approximation of the linear mapping. We can get the sparce when we subtract the low rank approximation from the original data.

Section II. Theoretical Background:

1. Singular Value Decomposition (SVD)

   Singular Value Decomposition is a matrix method for reducing a matrix to its constituent part in order to make certain subsequent matrix calculations simpler. It can be represented as

$$A = USV^T$$

where $U \in R^{m\times m}$and $V \in R^{n\times n}$are unitary matrices, and $s \in 12^{m\times n}$ is diagonal matrix with non-negative real numbers on the diagonal.

The diagonal entries σ_i of S are known as the singular values of S. The number of non-zero singular values is equal to the rank of S. The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of M. In this project, we use SVD to make the raw data simpler by lower the rank. With the data we kept, we can still see the main information we want.

2. Dynamic Mode Decomposition (DMD)
    1. The setup

    Assuming that we have snapshots of spatio-temporal data, we define:

N = number of spatial points saved per unit time snapshot

M = number of snapshots taken

Then we construct

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

Where $$U(x, tm) = \begin{bmatrix} U(x_1, t_m) \\ U(x_2\ , t_m) \\ . \\ . \\ . \\ U(x_{n,}\ t_m) \end{bmatrix}$$

2. Dynamic Mode Decomposition

    We firstly represent the matrix

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

By

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

applying to the Koopman operator. Now the columns are formed by applying the powers of A to the vector x_1, so we can write the above in the matrix form to get

$$x_2^M = Ax_1^{M-1} + re_{M-1}^T$$

Where e_{M-1} is the vector with all zeros except a 1 at the (M-1)st component.

Then we can use the SVD to rewrite the x_1^{M-1} in order to solve for A. The above form becomes

$$x_2^M = AU\Sigma V^* + re_{M-1}^T$$

We want to choose A in such as way that the columns in x_2^M can be written as linear combinations of the columns of U, which is the same as written them as linear combinations of the POD modes. Multiplying the above equation through by U^* on the left gives:

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

Then we isolate

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

The right hand side of this equation is $\tilde{S}$, which has the same eigen value with A. The eigenvector/eigenvalue pairs of $\tilde{S}$ can be written as

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

Thus giving the eigenvectors of A, called the DMD modes, by

$$\psi_k = Uy_k$$

Until here, we already have all we need to describe continual multiplications by A. We can expand in out eigen basis to get

$$x_{DMD}(t) = \sum_{k=1}^{k} b_k \psi\, k^{e^{wkt}} = \psi diag(e^{wkt})b$$

Next, we are going to talk about how to use DMD to separate the video stream to both the foreground and background.

The DMD spectrum of frequencies can be used to subtract background modes. We denote it as

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

$b_p \varphi_p e^{wpt}$ represents background video and $\sum_{j \neq p} b_j \varphi_j e^{wpt}$ represent foreground video.

Since both term are complex, this will pose a problem when we want real valued output. Thus, we calculate the DMD's approximate low-rank reconstruction according to

$$X_{DMD}^{Low-rank} = b_p \varphi_p e^{wpt}$$

Then the DMD's approximate sparce reconstruction

$$X_{DMD}^{sparse} = \sum_{j \neq p} b_j \varphi_j e^{wpt}$$

Can be calculated with real-valued elements only as follows

$$X_{DMD}^{sparse} = X - |X_{DMD}^{Low-rank}|$$

This way the magnitude of the complex values from the DMD reconstruction are accounted for, while maintaining the importance constraints that

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

Section III. Algorithm Implementation and Development

Main steps

1. Loading the video and using SVD to see the singular values
2. Initializing DMD

3. DMD reconstruction
4. Using DMD to separate the foreground and background

Firstly, I Loaded the two video clips and obtain the length of the video. Then I used SVD command to check the significant modes. After the spectrum was plotted, we check the energy of these singular values and capture the proper modes. With the modes determined, I truncate the diagonal matrix Σ and UV. With the modified matrices, I am able to initialize the DMD by constructing $\tilde{S}$ to get its eigen values Φ and ω. Then we use Φ to get the initial conditions y0 and calculate DMD solutions by u_modes(:,iter) = y0.*exp(omega*t(iter)).

After we have the low-rank of approximation, we subtract the absolute value of the of the approximation from the original data to get the sparse DMD matrix which is the foreground video.

Section IV. Computational Results

1. Monte Carlo

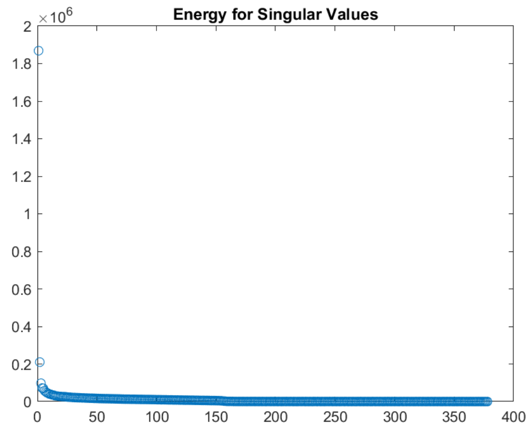

Figure1 energy for singular values for monte

From the graph, we saw two modes are significant here. Thus, we truncate the SVD with only two modes

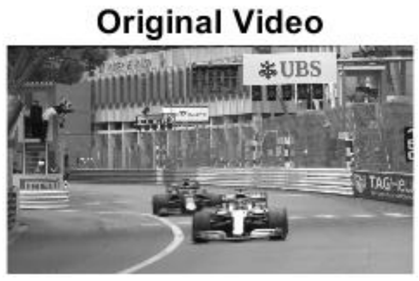

Figure2. Original video

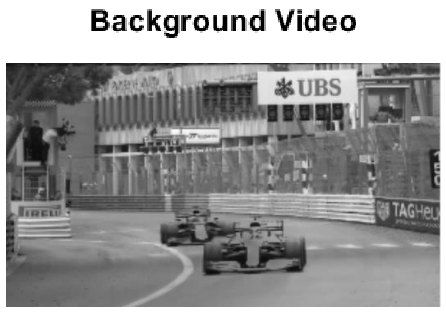

Figure2. background video

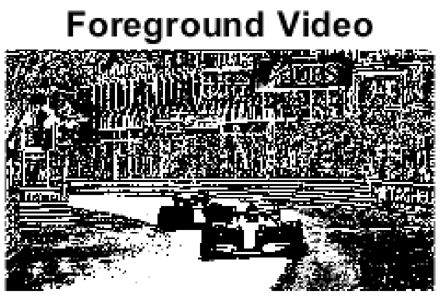

Figure3. Foreground Video

Observing the three graphs, we saw the Background race cars are more visible while the foreground one is not so clear.

2. Ski

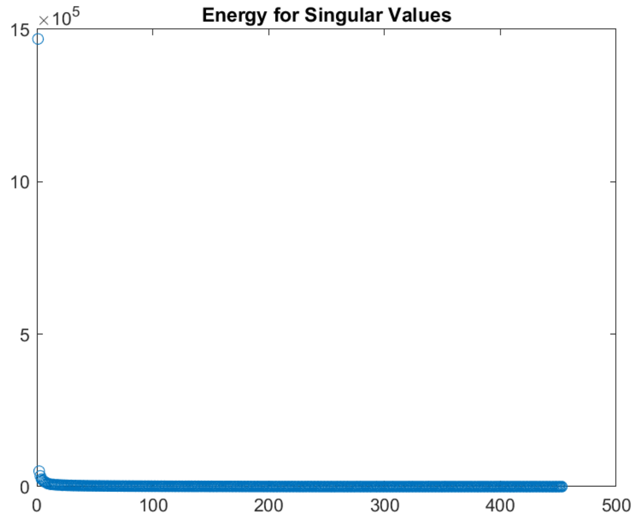

Figure5. energy for singular values for ski

From the energy of singular valued by Ski, we also use two modes although the second largest value is not so significant, it’s still obviously larger than any other modes.

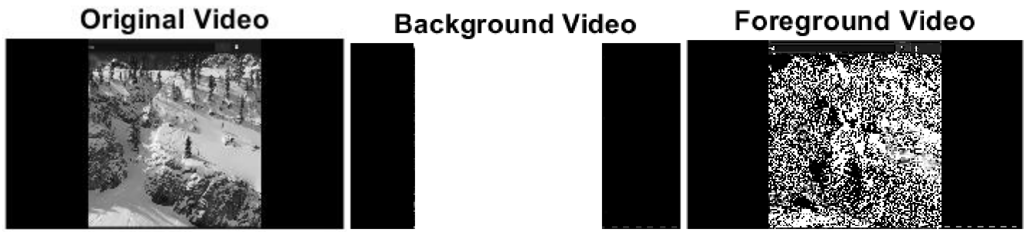

Figure6. videos for Ski

This time, both the background and foreground are both not clear as the original video maybe due to the wrong modes I used at the beginning.

Section V. Summary and Conclusions

I applied the Dynamic Mode Decomposition (DMD) method to both video. However, the monte Carlo one is doing better when we see the final result. The ski one are not good both for the foreground and background.

Appendix A. MATLAB functions used and brief implementation explanation

VideoReader(filename): Use a VideoReader object to read files containing video data

B = reshape(A,sz) reshapes A using the size vector, sz, to define size(B).

[U,S,V] = svd(A) performs a singular value decomposition of matrix A

Appendix B. MATLAB codes

```
%%
cd('C:\Users\75638\OneDrive - UW\Desktop');
ski = VideoReader('ski_drop_low.mp4')
monte = VideoReader('monte_carlo_low.mp4')

dt = 1/ski.Framerate;
dt = 1/monte.Framerate;
%%
ski1 = [];
while hasFrame(ski)
 frame_ski = readFrame(ski);
 frame_ski = rgb2gray(frame_ski);
 frame_ski = reshape (frame_ski, [], 1);
 ski1 = [ski1, frame_ski];
end
ski_length = size(ski1,2);
%%
monte1 = [];
while hasFrame(monte)
 frame_monte = readFrame(monte);
 frame_monte = rgb2gray(frame_monte);
 frame_monte = reshape (frame_monte, [], 1);
 monte1 = [monte1, frame_monte];
end
monte_length = size(monte1,2);

%%
monte1 = double(monte1);
X1 = monte1(:,1:end-1);
X2 = monte1(:,2:end);
[U, Sigma, V] = svd(X1,'econ');

%%
sv = diag(Sigma);
plot(linspace(1,length(sv),length(sv)),sv,'o')
title('Energy for Singular Values')
print('-dpng','svd.png')


%% only two modes are significant
U = U(:,1:2);
Sigma = Sigma(1:2,1:2);
V = V(:,1:2);
%%
S = U'*X2*V*diag(1./diag(Sigma));
```

```
[eV, D] = eig(S);
mu = diag(D);
omega = log(mu)/dt;
Phi = U*eV;
%%
y0 = Phi\X1(:,1);
t = 0:dt:monte.Duration;
u_modes = zeros(length(y0),length(t)-1);
for iter = 1:(length(t)-1)
    u_modes(:,iter) = y0.*exp(omega*t(iter));
end
u_dmd = Phi*u_modes;

%%
X_sparse = X1-abs(u_dmd);
R = X_sparse.*(X_sparse<0);
foreground = X_sparse - R;
background = R + abs(u_dmd);
X_r = foreground+background;

%%
recon = reshape(X_r, [540,960,378]);
 background = reshape(u_dmd, [540,960,378]);
 foreground = reshape(X_sparse, [540,960,378]);
 original = reshape(X1, [540,960,378]);
%%
subplot(3,1,1)
imshow(uint8(original(:,:,50)));
title("Original Video");

subplot(3,1,2)
imshow(background(:,:,50))
title('Background Video')

subplot(3,1,3)
imshow(foreground(:,:,50))
title('Foreground Video')
```