AMATH482 HW5
Background Subtraction in Video Streams

Winnie Shao

3/10/2021

Abstract

In this report, we are going to explore the basic mechanism of Dynamic Mode Decomposition. Specially, we will focus on its application on diagnostics of data associated with DMD mode and eigenvalues. This application is explored through its implementation on background subtraction in video streams. We will also examine the meaning of DMD modes and eigenvalues and how they influence the performance of DMD.

Section I: Introduction

Dynamic Mode Decomposition is a powerful technique to discover the low-dimensional spatial-temporal coherent structures from high dimensional data. It primarily enable the diagnostics of current data and future state prediction. In this report, the application is explored in the case of separating the background and foreground in video Streams. I demonstrate it with two example videos. The first video is a man skiing down the mountain. The second video is several cars running through the start-line.

Section II: Theoretical Background

Dynamic Mode Decomposition (DMD) is an ideal combination of spatial dimensional reduction techniques. When we apply DMD on data x, it is a regression of complex dynamic onto the linear system.

$$\frac{dx}{dt} = Ax \tag{1}$$

or in discrete-time system

$$x_{k+1} = A_{x_k} \tag{2}$$

where x is the original data with each column representing data from a frame of the video. While in x_{k+1}, each column is replaced by the column at its right as time moving forward. The relation is given by A = $e^{A\Delta t}$. We will take the low-rank eigen-decomposition of matrix A that fits the data with minimal $||x_{k+1} - A_{x_k}||_2$. An important thing to know is that, DMD usually work on the projection of A on POD mode given by SVD because the dimension of A usually will be too high to directly decompose or represent.

$$x(t) = \sum_{k=1}^{n} \phi_k \lambda_k b_k = \Phi \Lambda^k b \tag{3}$$

where ϕ_k and λ_k are the eigenvectors and eigenvalues of matrix , and b_k are the coordinates

$$\boldsymbol{\Phi = X' V \Sigma^{-1} W}$$

of x(0) in the eigenvector basis. We know where X' is the data in the second time frame, $V\Sigma^{-1}$ is the transpose data on POD mode, W is the eigenvectors of A. To make the equation from discrete to continuous, we will take $w_k = \frac{ln(\lambda_k)}{\Delta t}$. We can reconstruct the data by plug in the eigenvector, eigenvalue, and b_k

$$x(t) \approx \sum_{k=1}^{n} \phi_k e^{w_k t} b_k = \Phi e^{\Omega t} b \tag{4}$$

The eighenvalue w_k and the related $e^{w_k t}$ determine the frequency of the mode. If the frequency is 0, the mode is static.

Section III: Algorithm Implementation and Development

- Load the video
- Iterate through the frame of the video and convert the frame to black and white. Reshape it into a column vector and add it to the matrix X so that each column of matrix X is a frame of the video.
- Generate the staggered matrices X1, and X2 where column $x_j^1 = x_{j+1}^2$ from X1, X2 respectively. Take the SVD of X1.
- Plot the singular values and choose the low-rank approximation capturing 90 percent of the energy of the system.
- Use the low-rank approximation of X1 to calculate A, From A calculate the relevant Φ and frequency w and y_0.
- Find the minimum frequency so that the low frequency will be the stationary background.
- Calculate the low-rank DMD $X_{lowrank}$ and the sparse DMD X_{sparse}
- Replace negative entries in $X_{lowrank}$
- Plot the corresponding low-rank and sparse DMD of the data matrix

Section IV: Computational Results

Skiing Video:

The first video i tried is a short clip of a person skiing down the steep mountain. We can see from Figure 1 that there is a single dominant singular value. From our calculation, we only need the first singular value to capture 90 percent of the images. Therefore, I am using rank 1 approximation. In Figure 2, we can see the results of the DMD decomposition. It does a good job of separating the background and the moving person (a bit hard to see but it is the black dot on the second plot).

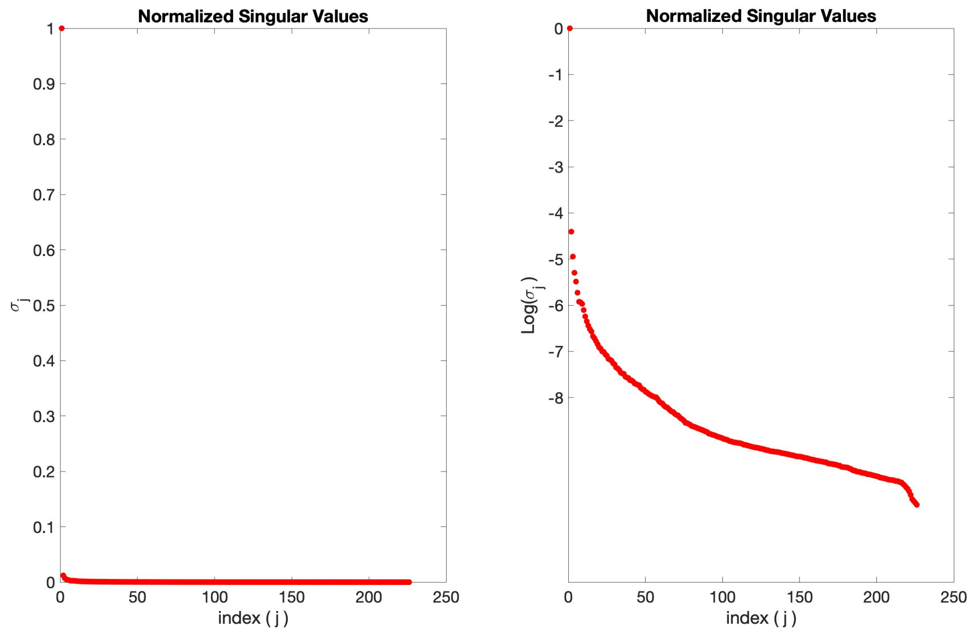

Figure 1: Singular values for the data matrix X storing each frame of the skiing video

F1 Racing Video:

The second video I tried is the F1 racing video. We can see from Figure 3 that there is a single dominant singular value and some other singular values may be important too. From out calculation, when rank = 89, we can capture 90 percent of energy in the image matrices. Therefore, we are going to use rank 98 approximation. As we can see in Figure 4, at this moment, the algorithm perfectly separate the moving cars and the background. However, we can see in Figure 5, the algorithm cannot separate the video stream for most of the time. I think it might due to the shaking camera and the way the moving cars move in the frame.

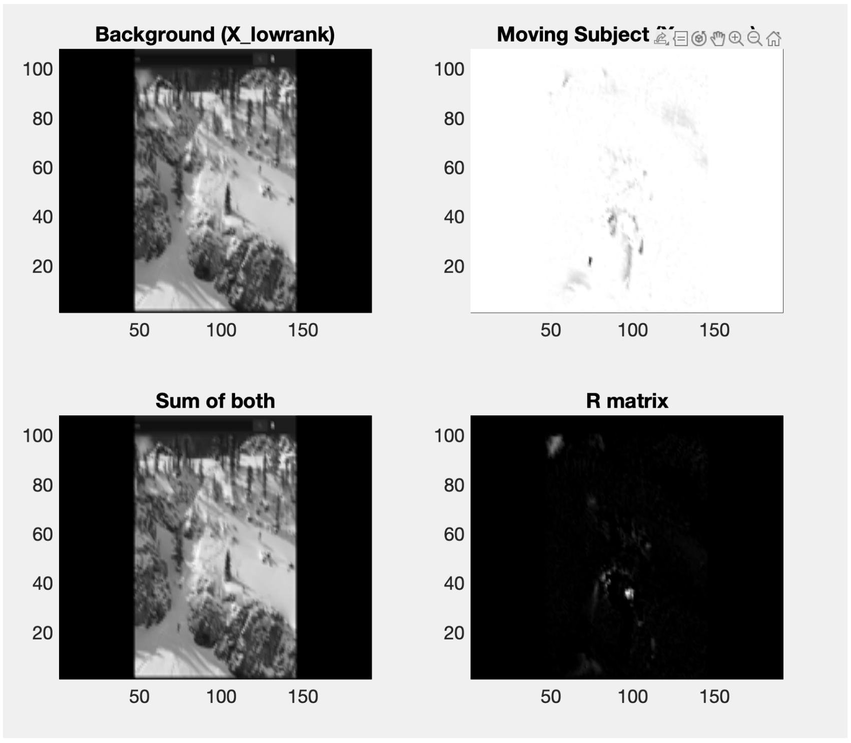

Figure 2: The screenshot of DMD decomposition of ski video. We have two parts: high frequency ares X sparse (moving objects) and low frequency areas X lowrank (background)

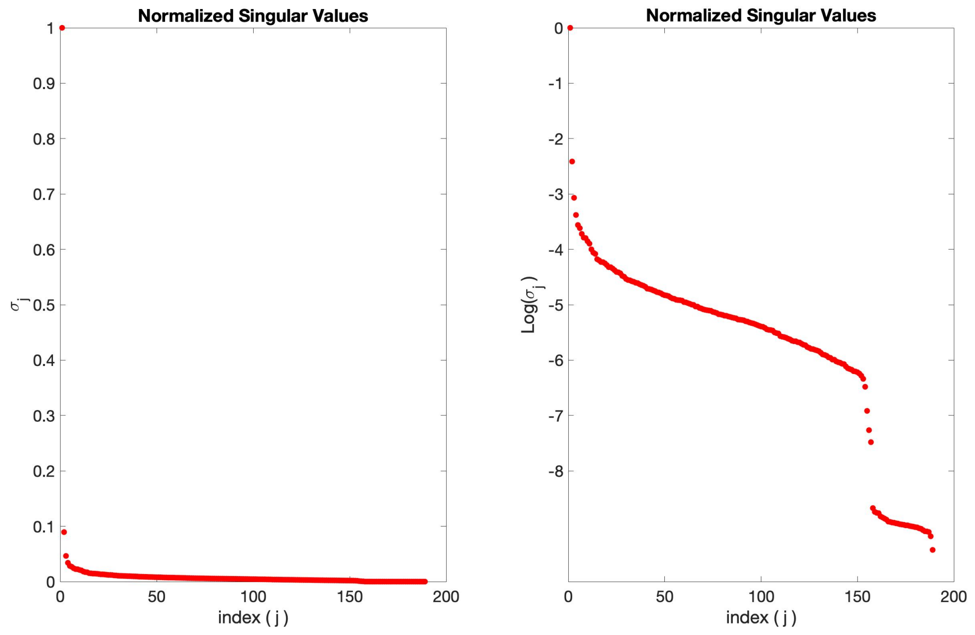

Figure 3: Singular values for the data matrix X storing each frame of the F1 racing video

Section V: Conclusion

DMD is a very powerful tool. By encoding each mode with specific frequency, it can capture spatio-temporal information what SVD would miss. However, like we see in the second

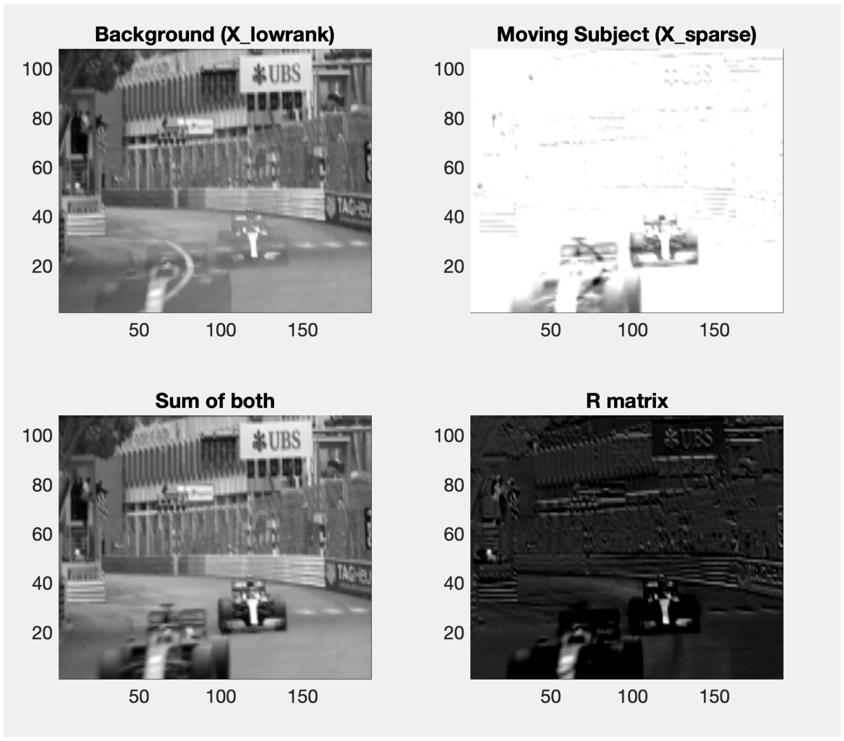

Figure 4: The screenshot of DMD decomposition of racing video. We have two parts: high frequency ares X sparse (moving objects) and low frequency areas X lowrank (background). This is a good representation.

example, DMD struggles with differentiating true movements from camera shake. To avoid this, there probably should be some pre-processing step performed before using DMD.

Appendix A: MATLAB Functions

- svd(X): Calculates the Singular Value Decomposition of the given matrix X, returning U, Σ, V.
- imcomplement(I): Given a black and white image matrix I, this returns a new image matrix with every pixel value inverted.

Appendix B: MATLAB Code

```
clear all; clc; close all;

video = VideoReader('ski_drop_low.mp4');
flip_contrast = 1;
frame_skip = 2;
resolution_reduction = 0.2; % reduce resolution by this factor.
imheight = video.Height*resolution_reduction;
```

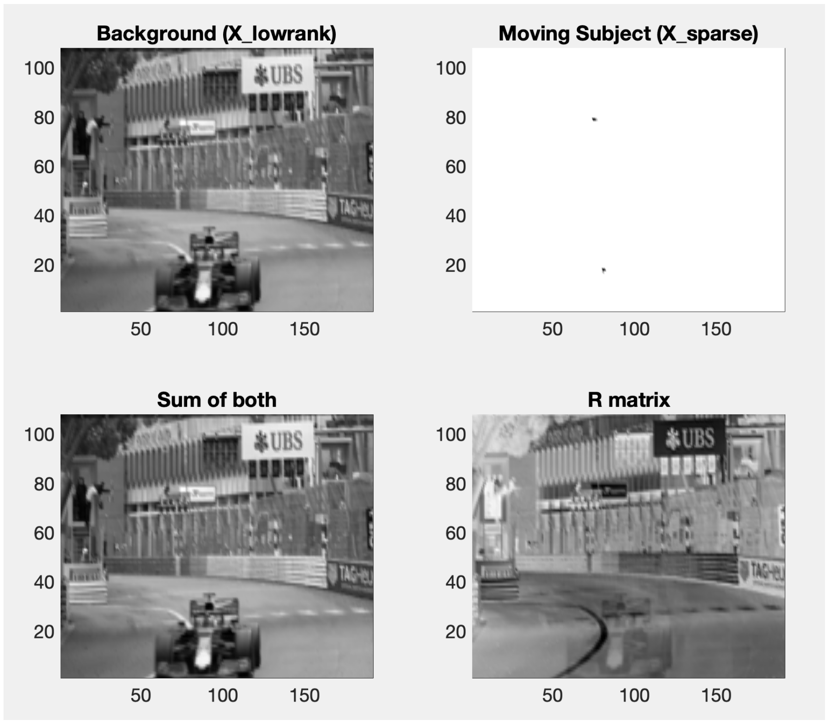

Figure 5: The screenshot of DMD decomposition of racing video. We have two parts: high frequency ares X sparse (moving objects) and low frequency areas X lowrank (background). This is a bad representation

```
imwidth = video.Width*resolution_reduction;
num_frames = ceil(video.Duration * video.FrameRate / frame_skip);
dt = frame_skip / video.FrameRate; % used later
% t = 0: dt :video.Duration;
dt = 1;
t = 1:num_frames;

v = zeros(num_frames, imheight, imwidth); % stores frames as rows of 2D ma
X = zeros(imheight*imwidth, num_frames); % stores frames as 1D columns

frame = 1;
index = 1;
while hasFrame(video)
    next_frame = readFrame(video);
    if mod(index, frame_skip) == 0
        x = imresize(next_frame, resolution_reduction);
        %imshow(x);
        if flip_contrast
            v(frame, :, :) = imcomplement(rgb2gray(x)); % imcomplement
        else
            v(frame, :, :) = rgb2gray(x); % dont imcomplement
        end
        X(:, frame) = reshape(v(frame,:,:), [imheight*imwidth, 1]);
        frame = frame + 1;
    end
    index = index + 1;
end


X1 = X(:,1:end-1); X2 = X(:,2:end);

[U2,Sigma2,V2] = svd(X1, 'econ');

threshold = 0.90;
r = find(cumsum(diag(Sigma2) ./ sum(diag(Sigma2))) > threshold   ,1);
%r = size(Sigma2, 1); % full rank
figure(1)
subplot(121)
plot(diag(Sigma2)./max(diag(Sigma2)), 'r.', 'markersize', 20);
title('Normalized Singular Values')
xlabel('index ( j )')
ylabel('\sigma_j')
yticks(0:0.1:1)
set(gca, 'fontsize', 20);
subplot(122);
plot(log(diag(Sigma2)./max(diag(Sigma2))), 'r.', 'markersize', 20);
title('Normalized Singular Values')
xlabel('index ( j )')
ylabel('Log(\sigma_j )')
yticks(-8:1:0)
set(gca, 'fontsize', 20);

U=U2(:,1:r); Sigma=Sigma2(1:r,1:r); V=V2(:,1:r);
Atilde = U'*X2*V/Sigma;
[W,D] = eig(Atilde);
Phi=X2*V/Sigma*W;

mu=diag(D);
omega=log(mu)/dt;

y0 = Phi\X(:, 1);  % pseudo-inverse initial conditions


[min_omega, min_index] = min(abs(omega));
foreground_modes_indices = find(abs(omega) > min_omega);

%%
```

```
clc; close all;

X_lowrank = y0(min_index).*Phi(:, min_index).*exp(omega(min_index).*t);
X_sparse = X - abs(X_lowrank);

R = X_sparse .* (X_sparse < 0); % places all negative entries in R



X_lowrank2 = abs(X_lowrank) + R;
X_sparse2 = X_sparse - R;

%% clc; close all;

figure(2)
set(gcf, 'Position', [100, 100, 1000, 1000])
for frame = 1:size(X_lowrank2, 2)
    if flip_contrast
       lowrank = imcomplement(reshape(X_lowrank2(:, frame), [imheight, imw
       sparse = imcomplement(reshape(X_sparse2(:, frame), [imheight, imwidt
       rj = imcomplement(reshape(R(:, frame), [imheight, imwidth]));
    else

       lowrank = reshape(X_lowrank2(:, frame), [imheight, imwidth]);
       sparse = reshape(X_sparse2(:, frame), [imheight, imwidth] );
       rj = reshape(R(:, frame), [imheight, imwidth]);
    end

   subplot(221)
   %imshow(uint8(lowrank));
   pcolor(flipud(lowrank)), shading interp, colormap(gray);
   title('Background (X\_lowrank)')
   set(gca, 'fontsize', 20);

   subplot(222)
   %imshow(uint8(sparse));
   pcolor(flipud(sparse)), shading interp, colormap(gray);
   title('Moving Subject (X\_sparse)')
   set(gca, 'fontsize', 20);

   subplot(223)
   %imshow(uint8(sparse + lowrank + rj));
   pcolor(flipud(sparse + lowrank)), shading interp, colormap(gray);
   title('Sum of both')
   set(gca, 'fontsize', 20);
```

```
    subplot(224)
    %imshow(uint8(abs(rj)));
    pcolor(flipud(rj)), shading interp, colormap(gray);
    title('R matrix')
    set(gca, 'fontsize', 20);

    drawnow;
end
```