Background Subtraction in Video Streams

Haoran Sun

March 16, 2021

Abstract

In this assignment,we will be separating the background and foreground of two videos with the application of Dynamic Mode Decomposition(DMD). In the foreground video, only moving objects will appear with the steady background being subtracted. We successfully separate the foreground and background videos in both videos.

1 Introduction and Overview

In this task, we are provided two videos. In the first video, there is a steady background that has no time-dynamic, and a skier, which keeps moving with respect to the time frame. And in the second video, the background also keeps steady, with a motion of racing cars in the foreground. We would like to separate the background and foreground using the Dynamic Mode Decomposition and separate their corresponding dynamic modes from the overall dynamic modes.
Since the background is steady, we would expect the mode does not change with respect to time, which we will be looking for to get the background isolated. Subtracting the isolated background from the original data will therefore give us the foreground.

2 Theoretical Background

2.1 Dynamic Mode Decomposition

The purpose of Dynamic Mode Decomposition(DMD) is to predict the time dynamics of the given data without relying on any equations. It aims to find a basis of spatial modes, in which the time dynamics are just exponential functions. The DMD method approximates the spatial modes of the **Koopman operator A**, which satisfies:

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

j is the specific time of the data and x_j is the data collected at time j.Therefore, $\mathbf{A}$ is the linear operator that maps the data to the data of a later time. Even the dynamics of the system are nonlinear in most cases, we can use a linear mapping to predict it. The goal is to find this linear operator $\mathbf{A}$. To do so, we will first consider the matrix

$$\mathbf{X}_1^{M-1} = [x_1, x_2, x_3, ...x_{M-1}] = [x_1, \mathbf{A}x_1, \mathbf{A}^2x_1, ...\mathbf{A}^{M-2}x_1] \tag{2}$$

Where $\mathbf{X}_1^{M-1}$ is the data for first M-1 snapshots. The linear operator $\mathbf{A}$ relates every snapshots with the first snapshot x_1. It will give us

$$\mathbf{X}_2^M = \mathbf{A}\mathbf{X}_1^{M-1} + \mathbf{r}e_{M-1}^T \tag{3}$$

The last term accounts for the residual because the final point X_M was not included when we maps $\mathbf{X}_1^{M-1}$ to $\mathbf{X}_2^M$. If we apply SVD on matrix $\mathbf{X}_1^{M-1}$, from Equation 3, we obtain:

$$\mathbf{X}_2^M = \mathbf{A}\mathbf{U}\Sigma\mathbf{V}^* + \mathbf{r}e_{M-1}^T \tag{4}$$

We want $\mathbf{X}_2^M$ to be written as a linear combinations of the columns of $\mathbf{U}$. Therefore, the residual term will be orthogonal to $\mathbf{U}$ columns, which gives $\mathbf{U} * \mathbf{r} = 0$. Multiply both sides of Equation 4 by $\mathbf{U}^*$, we have

$$\mathbf{U}^*\mathbf{X}_2^M = \mathbf{U}^*\mathbf{A}\mathbf{U}\Sigma\mathbf{V}^* \tag{5}$$

then we isolate $\mathbf{U}^*\mathbf{A}\mathbf{U}$, which gives

$$\mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1} \tag{6}$$

We name the RHS of the equation $\tilde{\mathbf{S}}$, which can be calculated directly from the input data. We notice $\tilde{\mathbf{S}}$ and $\mathbf{A}$ are similar matrix, meaning they have the same eigenvalues, and the eigenvectors of $\tilde{\mathbf{S}}$ is $\mathbf{U}y$, where y is the eigenvector of $\mathbf{A}$. This gives the **DMD modes**, which is the eigenvectors of $\mathbf{A}$, to be:

$$\psi_k = \mathbf{U}\mathbf{y}_k \tag{7}$$

Therefore, we have our continual multiplications expanded in eigenbasis:

$$\mathbf{x}_{DMD}(t) = \sum_{k=1}^{K} b_k\psi_k e^{\omega_k t} = \Psi diag(e^{\omega_k t})\mathbf{b} \tag{8}$$

Ψ has eigenvectors ψ_k as its columns. And $\omega_k = \ln(\mu_k)/\Delta t$, coming from the convention to convert discrete time linear solution into continuous time solution.
Notice that we don't have to use the full matrix Σ to apply this method. Instead, we can use low-rank approximation for DMD for dimensionality reduction.

2.2 Background Subtraction

Since the background is steady, we would expect it does not evolve in time and its corresponding mode has $||\omega_p||$ very close to 0. From Equation 8, we have

$$\mathbf{X}_{DMD}(t) = b_p\psi_p e^{\omega_p t} + \sum_{j\neq p} b_j\psi_j e^{\omega_j t} \tag{9}$$

Where the first term on RHS is the mode for background video, and the rest is for foreground video. The mode for background can be calculated by DMD's low-rank reconstruction:

$$\mathbf{X}_{DMD}^{Low-Rank} = b_p\psi_p e^{\omega_p t} \tag{10}$$

And the modes for foreground video can be obtained by simply subtract the low-rank DMD reconstruction from the original data.

3 Algorithm Implementation and Development

For both videos, we will first read the video and turn it into gray-scale. Then we will reshape the data matrix into the matrix where the columns record the information of pixels, and different columns represent pixels information for different frames. This matrix $\mathbf{X}$ will be our data. Firstly, we will apply the low-rank DMD algorithm on the data to get our background mode.

1. Construct $\mathbf{X}_1^{M-1}$ and $\mathbf{X}_2^M$ from $\mathbf{X}$
2. Compute the SVD of $\mathbf{X}_1^{M-1}$
3. Calculate $\tilde{\mathbf{S}} = \mathbf{U}^*\mathbf{X}_2^M\mathbf{V}\Sigma^{-1}$ and find its eigenvalues and eigenvectors. The $\mathbf{U}$, Σ and $\mathbf{V}$ in this case are the low-rank approximation of the original $\mathbf{U}$, Σ and $\mathbf{V}$.
4. find the coefficients b_p using the initial condition $\mathbf{x}_1$ and the pseudoinverse Ψ.
5. Compute the background mode according to Equation 10.

After obtained the DMD mode for background, we can obtain our foreground video modes(Sparse DMD) by simply subtract the background mode(Low-rank DMD) from the original data, as elaborated in section 2.2. However, the terms in both approximate low-rank and sparse reconstructions are complex, while we desire real-valued outputs.

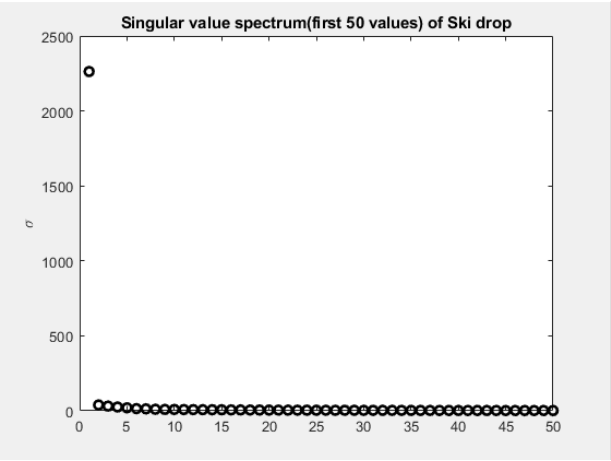


Figure 1: first 50 singular value of $\mathbf{X}_1^{M-1}$ (Left) of Ski video

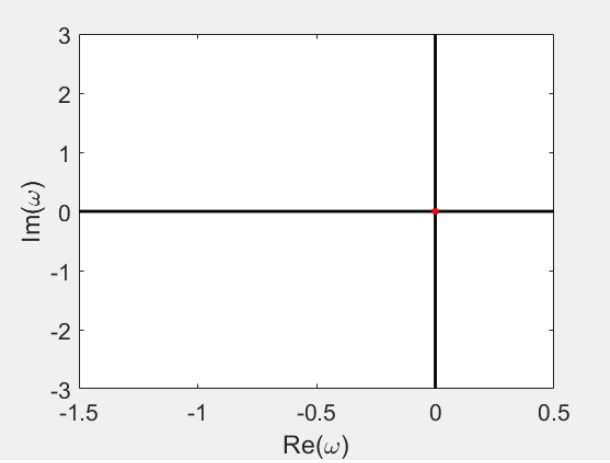


Figure 2: ω of low-rank DMD mode (right) of Ski video

By achieving this, we will subtract the modulus of low-rank DMD mode from the original data, which results the Sparse DMD reconstruction modes to have real values. However, another problems arises, that some elements in the sparse DMD reconstructions are negative, which does not make sense to have negative pixel intensities. The approach I used in this case is simply add a uniform number to all elements, which is large enough to bring negative elements into positive. This approach turns out to be working very well.

4 Computational Results

4.1 Skier Video

The first 50 singular value of $\mathbf{X}_1^{M-1}$ matrix of the data matrix in the Skier video is shown in Figure 1. It is obvious that the first singular value dominates the whole system. With this being said, we will only use one mode for our approximate low-rank DMD reconstructions. The ω of this one-rank DMD reconstruction mode is shown in Figure 2. The value of $||\omega||$ can be seen to be 0, or so close to 0 that cannot be distinguished visually. It means this low-rank DMD mode will not evolve in time, corresponding to the steady background in the video. After executing the algorithm described in Algorithm Implementation section, we successfully obtained the separated foreground and background video from the original video. The comparison of one frame are shown in Figure 3. In the original frame, we can see the skier rushing down from the mountain. The foreground in the middle highlights the skier(the black figure in the middle of the figure) with background being wiped out. And in the background frame, the skier figure is eliminated while the steady background mountain are kept perfectly. The foreground videos are obtained by adding a value of 0.4 to every elements in the Sparse DMD reconstruction modes, which eliminates negative values.

4.2 Monte Carlo video

Similar to the Ski video case in last section, we first analyze the singular value of $\mathbf{X}_1^{M-1}$ matrix of the data matrix in the Skier video to decide how many modes we use for our Low-rank approximate DMD reconstruction. The first 100 singular values of $\mathbf{X}_1^{M-1}$ are shown in Figure 4. Although the first singular value still dominates most of the system, the second one also has a large value compared to the rest. Hence, we choose to use 2 modes for Low-rank DMD reconstruction. The values of ω of the modes are shown in Figure 5. We have two red dots on the Figure because we used two modes. It is clear that both ω has the value very close to 0, which is good for reconstructing the steady background.

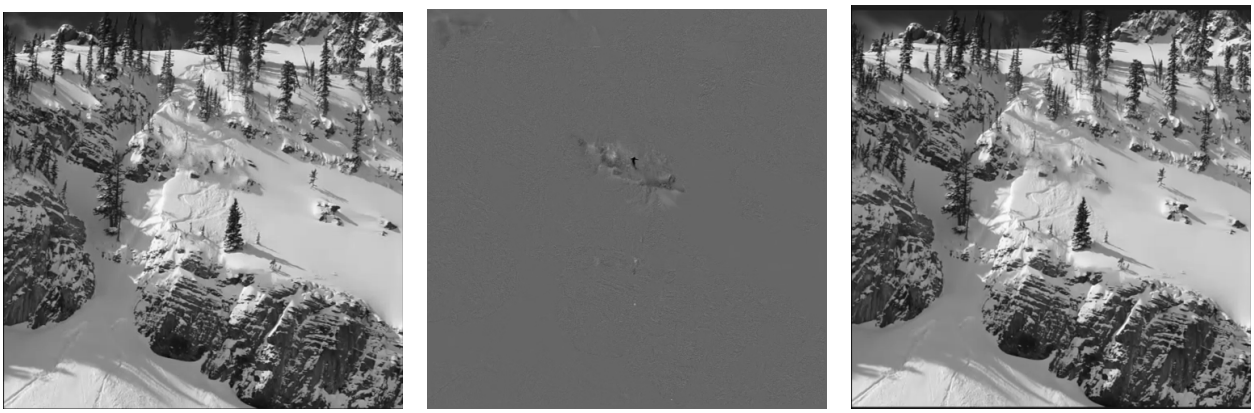
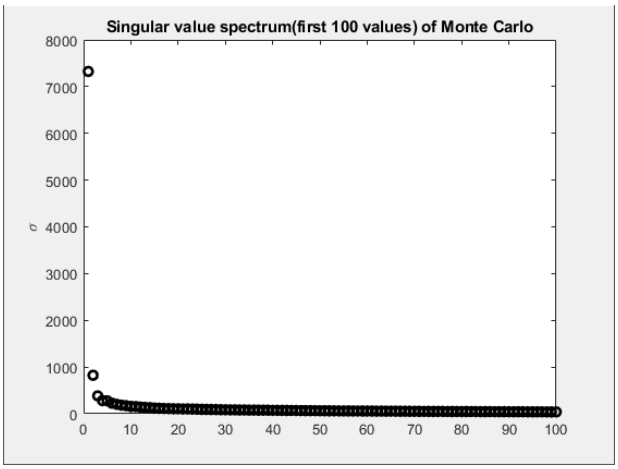


Figure 3: Original(Left), Foreground(Middle), and Background(Right) of frame 30 in Ski video



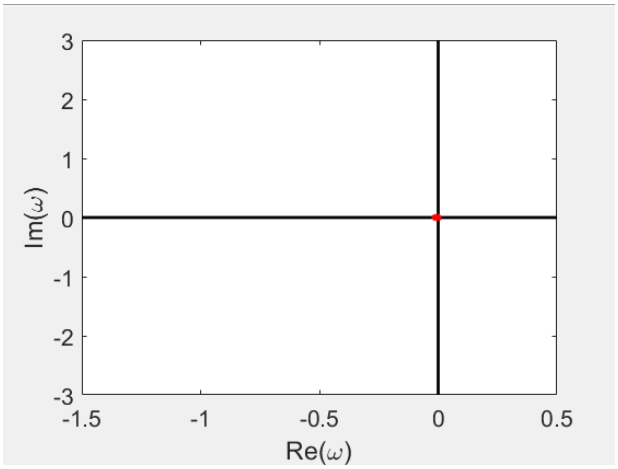


Figure 4: first 100 singular value of $\mathbf{X}_1^{M-1}$ of Monte Carlo video(Left)

Figure 5: ω of low-rank DMD mode of Monte Carlo video(right)

One frame of the separated foreground video, background video and the original video are shown in Figure 6. In the original frame, we can see the racing cars, shaking flag and background. In the Foreground figure, the racing cars and the flag are highlighted. We can still observe the vestige of the background in the foreground figure, which is due to the camera shaking in the original video that gives the background a small time-dynamic as well. In the background figure, the background is maintained perfectly with the racing car and flag erased. Same as the Ski video case, we add a value of 0.4 to every elements in the Sparse DMD reconstruction modes.

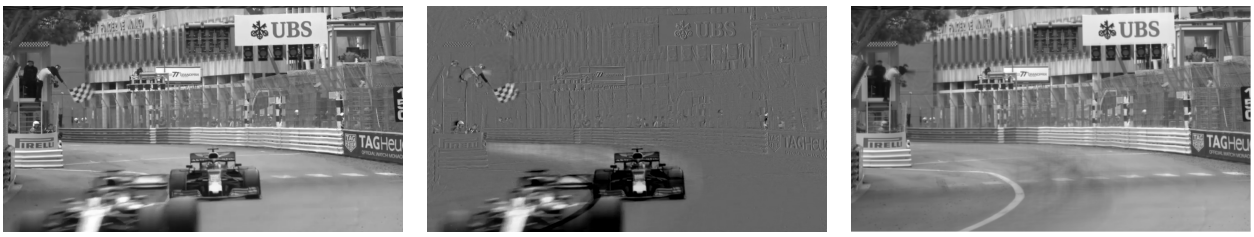


Figure 6: Original(Left), Foreground(Middle), and Background(Right) of frame 40 in Monte Calor video

5 Summary and Conclusions

Using DMD and background subtraction, we obtained very good results for separating foreground part and background part of both videos. In the Monte Carlo case, the foreground is a little noisy with some background appears, but it is mainly due to the camera shaking in the original video and does not affect the highlight of the moving foreground objects. The method we used to eliminate negative elements in our sparse DMD matrix is by adding a constant to every elements, which works well in this case. Potentially, there might be better ways to deal with the negative elements that we could be looking into.

Appendix A MATLAB Functions

- `VideoReader` Create a multimedia reader object.
- `im2double` Convert image to double precision.
- `reshape(X,M,N)` Returns the M-by-N matrix whose elements are taken columnwise from X.
- `[U,S,V] = svd(X)` produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that X = U*S*V'.
- `diag(V)` puts V on the main diagonal
- `histogram(X)` plots a histogram of X.
- `[V,D] = eig(A)` produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that A*V = V*D.

Appendix B MATLAB Code

```
clear; close all; clc

%% Read video
v = VideoReader('monte_carlo_low_Trim.mp4');
frames = read(v,[1 Inf]);
[height,width, rgb, num_frames] = size(frames);
time = v.CurrentTime;
dt = time / (num_frames - 1);
t = 0:dt:time;

%% Turn video into grayscale, make DMD Matrices
pixel = height*width;
X = zeros(pixel, num_frames);
for j=1:num_frames
    X_frame = frames(:,:,:,j);
    X_frame = im2double(rgb2gray(X_frame));
    X_frame_rs = reshape(X_frame,pixel,1);
    X(:,j) = X_frame_rs;
end


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

%% SVD of X1

[U, Sigma, V] = svd(X1,'econ');
%% Low-rank approximation
mode = 2;
U_low = U(:,1:mode);
Sigma_low = Sigma(1:mode,1:mode);
V_low = V(:,1:mode);
diag_sigma = diag(Sigma);
plot(diag_sigma(1:50),'ko','Linewidth',2);
title('Singular value spectrum(first 50 values) of Monte Carlo');
ylabel('\sigma')
%% Computation of ~S
S = U_low'*X2*V_low*diag(1./diag(Sigma_low));
[eV, D] = eig(S); % compute eigenvalues + eigenvectors
mu = diag(D); % extract eigenvalues
%% phi & omega
omega = log(mu)/dt;
Phi = U_low*eV;

%% Create DMD Solution

y0 = Phi\X1(:,1); % pseudoinverse to get initial conditions

u_modes = zeros(length(y0),num_frames);
for iter = 1:num_frames
   u_modes(:,iter) = y0.*exp(omega*t(iter));
end
```

```
X_dmd_low = Phi*u_modes;

%% Plotting Eigenvalues (omega)

% make axis lines
line = -15:15;

plot(zeros(length(line),1),line,'k','Linewidth',2) % imaginary axis
hold on
plot(line,zeros(length(line),1),'k','Linewidth',2) % real axis
plot(real(omega)*dt,imag(omega)*dt,'r.','Markersize',15)
xlabel('Re(\omega)')
ylabel('Im(\omega)')
set(gca,'FontSize',16,'Xlim',[-1.5 0.5],'Ylim',[-3 3])

%% video of low-rank DMD;
X_dmd_low_video = reshape(X_dmd_low,height,width,num_frames);
for j=1:num_frames
    X_play=X_dmd_low_video(:,:,j);
    %X_play = abs(X_play);
    imshow(X_play); drawnow
end

%% Sparse DMD
x_dmd_low_abs = abs(X_dmd_low);
X_sparse = X - x_dmd_low_abs;

%% Foreground
X_fore_video = reshape(X_sparse,height,width,num_frames);
for j=1:num_frames
    X_play=X_fore_video(:,:,j);
    X_play = X_play + 0.4;
    %X_play = abs(X_play);
    imshow(X_play); drawnow
end


%% find residual matrix R
X_sparse_negative_logical = X_sparse < 0;
R = X_sparse.*X_sparse_negative_logical;
%% Find X_sparse and X_lowRank with R;
X_sparse_R = X_sparse - R ;
X_low_R = R + abs(X_dmd_low);
%% foreground with R
X_sparse_R_video = reshape(X_sparse_R,height,width,num_frames);
for j=1:num_frames
    X_play=X_sparse_R_video(:,:,j);
%     X_play = abs(X_play);
    imshow(X_play); drawnow
end

%% background with R
X_low_R_video = reshape(X_low_R,height,width,num_frames);
for j=1:num_frames
```

```
    X_play=X_low_R_video(:,:,j);
%     X_play = abs(X_play);
    imshow(X_play); drawnow
end
```

Code for one video