AMATH 582 Homework 5

Qiangqiang Guo

Due March 17, 2021

Abstract

In this assignment, we explore the dynamic mode decomposition (DMD) techniques by analyzing two videos. We use DMD to capture the dynamics of each videos stream and separate the background and foreground of each video stream using the DMD spectrum of frequencies.

1 Introduction and Overview

In generally, videos usually consist of backgrounds and foregrounds. For example, a video capturing the falling process of an apple contains the background of the static scenic and the foreground of the falling apple. Distinguishing the background and foreground of a video stream is an interesting topic for computer vision. In this assignment, we will use the dynamic mode decomposition (DMD) method to extract the background (or foreground) of two video streams.

In the following of this report, we will first introduce the theoretical background used in this assignment (Section 2). Then, we show how to design and implement the analysis algorithm (Section 3). We show the computation results in Section 4 and conclude this report in Section 5.

2 Theoretical Background

The key idea of DMD is to estimate the nonlinear dynamics of a system by linear, time-independent Koopman operator. Consider m snapshots of data collected at regularly spaced intervals of time $t_i, i = 1, 2, ...m$. Each snapshot $\mathbf{x}_i$ consists of n data points. We can aggregated the data to two $n \times (m-1)$ matrixs

$$\mathbf{X}_1^{m-1} = \left[\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_{m-1}\right] \tag{1}$$

$$\mathbf{X}_2^{m} = \left[\mathbf{x}_2, \mathbf{x}_3, ...\mathbf{x}_{m}\right] \tag{2}$$

The Koopman operator $\mathbf{A}$ maps the data at time i to time $i+1$: $\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i$. Apply it to Equation 1, we have

$$\mathbf{X}_1^{m-1} = \left[\mathbf{x}_1, \mathbf{A}\mathbf{x}_2, ...\mathbf{A}^{m-2}\mathbf{x}_{m-1}\right] \tag{3}$$

The columns of $\mathbf{X}_1^{m-1}$ form a Krylov space. The final data point $\mathbf{x}_m$ is represented in terms of this Krylov basis, thus

$$\mathbf{x}_m = \sum_{i=1}^{m-1} b_i \mathbf{x}_i + r \tag{4}$$

where b_i are the coefficients of the krylov space vectors and $\mathbf{r}$ is the residual that lies outside the Krylov space.

Generalizing $\mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i$ to the matrix form, we have

$$\mathbf{X}_2^{m} = \mathbf{A}\mathbf{X}_1^{m-1} \tag{5}$$

Considering Equation 4, the left hand side of Equation 5 can be written as

$$\mathbf{X}_2^{m} = \mathbf{X}_1^{m-1}\mathbf{S} + \mathbf{r}\mathbf{e}_{m-1}^{*} \tag{6}$$

where $\mathbf{e}_{m-1}$ is the $(m-1)$th unit vector and $[*]$ represents the conjugate transpose operator, and

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & b_1 \\ 1 & 0 & \cdots & 0 & 0 & b_2 \\ 0 & 1 & \ddots & 0 & 0 & b_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & b_{m-2} \\ 0 & 0 & \cdots & 0 & 1 & b_{m-1} \end{bmatrix} \tag{7}$$

In order to take the advantage of low-dimensional structures, we can use the singular value decomposition (SVD) to reduce the dimension of the matrix $\mathbf{X}_1^{m-1}$. The SVD of $\mathbf{X}_1^{m-1}$ is

$$\mathbf{X}_1^{m-1} = \mathbf{U\Sigma V}^* \tag{8}$$

where $U \in \mathbb{C}^{n\times n}$ is unitary where each column represents one new basis, i.e., U is the new feature space. $V \in \mathbb{C}^{m\times m}$ is unitary, which represents the projection of each original column onto the principle modes. $\Sigma \in \mathbb{R}^{n\times m}$ is diagonal where each diagonal element represents a singular value.

Then, we can reduce the rank of $\mathbf{X}_1^{m-1}$ by choosing a parameter r and approximate $\mathbf{X}_1^{m-1}$ by $\mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^*$ where $\mathbf{U}_r = U(:, 1:r)$, $\mathbf{\Sigma}_r = \mathbf{\Sigma}(1:r, 1:r)$, and $\mathbf{V}_r = \mathbf{V}(:, 1:r)$. The parameter r is chosen to reduce the rank as much as possible while maintaining the fundamental structure and dynamics. Assuming the residual error is small, i.e., $\|\mathbf{r}\| \ll 0$, from Equation 6, we can estimate

$$\mathbf{X}_2^m \approx \mathbf{X}_1^{m-1}\mathbf{S} \rightarrow \mathbf{X}_2^m \approx \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^*\mathbf{S} \rightarrow \mathbf{S} \approx \mathbf{V}_r\mathbf{\Sigma}_r^{-1}\mathbf{U}_r^*\mathbf{X}_2^m \tag{9}$$

Using the similarity transform of $\mathbf{V}_r\mathbf{\Sigma}_r^{-1}$, we have

$$\tilde{\mathbf{S}} \approx \mathbf{U}_r^*\mathbf{X}_2^m\mathbf{V}_r\mathbf{\Sigma}_r^{-1} \tag{10}$$

which is mathematically similar to the matrix $\mathbf{S}$.

By Equation 5, 6, 9, 10, we have $\mathbf{AX}_1^{m-1} = \mathbf{X}_2^m \approx \mathbf{X}_1^{m-1}\mathbf{S}$. Then, some of the eigenvalues of the matrix $\mathbf{S}$ approximate the eigenvalues of the Koopman operator $\mathbf{A}$. In addition, the eigenvalues of matrix $\tilde{\mathbf{S}}$ approximate those of $\mathbf{S}$. Consider the eigenvalue problem of $\tilde{\mathbf{S}}$:

$$\tilde{\mathbf{S}}\mathbf{y}_k = \mu_k\mathbf{y}_k, \quad k = 1, 2, ...r \tag{11}$$

The eigenvalues μ_k capture the time dynamics of the discrete Koopman operator $\mathbf{A}$ as a Δt step is taken forward in time. We can transform such eigenvectors back to those of $\mathbf{S}$ and $\mathbf{A}$:

$$\psi_k = \mathbf{U}\mathbf{y}_k \tag{12}$$

For the benefit of predicting future dynamics, also considering Koopman operator is linear, we can convert the eigenvalues to Fourier modes by defining

$$\omega_k = \frac{\ln \mu_k}{\Delta t} \tag{13}$$

Then, we can approximate the dynamics of the system at all future times, $\mathbf{x}_{DMD}(t)$ by

$$\mathbf{X}_{DMD}(t) = \sum_i^r b_i\psi_i e^{\omega_i t} = \mathbf{\Psi}\text{diag}(\exp(\omega t))\mathbf{b} \tag{14}$$

where $\mathbf{\Psi}$ is the matrix of which the columns are the eigenvectors ψ_k and $\mathbf{b}$ is the initial conditions with $\mathbf{b} \approx \mathbf{\Psi}^{-1}\mathbf{x}_1$.

Considering the video stream, it can be shown that a part of the first frame that does not change or changes very slowly in time should have an associated Fourier mode ω_k that is located near the origin in the complex space, $\|\omega_k\| \approx 0$. Therefore, we can separate the background and foreground information using the

DMD reconstruction of the video. The low-rank information represents the background information since the background is usually static, while the sparse information can represent the foreground information.

As discussed in the assignment description, assume that ω_p where $p \in 1, 2, ..., r$, satisfies $\|\omega_p\| \approx 0$ and that $\|\omega_j\|, j \neq p$ is bounded away from zero. The DMD reconstruction can be divided to the background and foreground

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

where $\mathbf{X}_{DMD}^{Low-Rank} = \sum_p b_p \psi_p e^{\omega_p \mathbf{t}}$ representing the background information and $\mathbf{X}_{DMD}^{Sparse} = \sum_{j \neq p} b_j \psi_j e^{\omega_j \mathbf{t}}$ representing the foreground information. In addition, we use the method described in the assignment description to handle the complex elements.

3 Algorithm Development and Implementation

3.1 Read the video data and setup

We first read in the video data and convert it to matrix format. Specifically, each video frame is originally a $a \times b \times 3$ (rgb) matrix, where a is the height of each frame and b is the width. We firstly transform it to gray format ($a \times b$) and then reshape it to a $ab \times 1$ vector. Finally, we aggregate all the frames to a $n \times m$ matrix, where $n = ab$ and m is the number of frames in the video. We also construct $\mathbf{X}_1^{m-1}$ and $\mathbf{X}_2^m$ based on this matrix.

3.2 SVD decomposition of $\mathbf{X}_1^{m-1}$

We then conduct the SVD on matrix $\mathbf{X}_1^{m-1}$, and plot the singular value spectrum. Based on this spectrum and the experiment of re-constructing the images, we determine the number of modes that are necessary for good image reconstruction, i.e., the rank r of the digit space.

3.3 Koopman operator approximation

We approximate the matrix $\tilde{\mathbf{S}}$ using Equation 10. Then, we calculate the eigenvalues and eigenvectors of $\tilde{\mathbf{S}}$ (i.e., Equation 11). After that, we approximate the eigenvectors of the koopman operator by Equation 12. Meanwhile, we convert the eigenvalues to Fourier modes using Equation 13. Based on those Fourier modes, we select those modes ω_p that are located near the origin, i.e., $\|\omega_p\| \approx 0$ as the modes of the low-rank DMD approximation. Note that by $\|\omega_p\| \approx 0$, we mean $\|\omega_p\| < \epsilon$ in numerical evaluation. We set $\epsilon = 0.01$ after pre-experiments. The transformed eigenvectors with the same indexes as ω_p are extracted as the Ψ_p.

3.4 Background and foreground extraction

We firstly calculate the initial condition of each video frame by $\mathbf{b}_p = \Psi_p^{-1}\mathbf{x}_1$. Then, we reconstruct the low-rank (i.e., background) DMD dynamics by $\mathbf{X}_{DMD}^{Low-Rank} = \sum_p b_p \psi_p e^{\omega_p \mathbf{t}}$. Then, as instructed in the the assignment descriptions, we reconstruct the initial sparse approximation by $\mathbf{x}_{DMD}^{Sparse} = \mathbf{x} - |\mathbf{x}_{DMD}^{Low-Rank}|$. Then, we find the residual negative matrix $\mathbf{R}$ and adjust the low-rank and sparse approximation as$\mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \leftarrow \mathbf{R} + \mid \mathbf{X}_{\text{DMD}}^{\text{Low-Rank}} \mid$ and $\mathbf{X}_{\text{DMD}}^{\text{Sparse}} \leftarrow \mathbf{X}_{\text{DMD}}^{\text{Sparse}} - \mathbf{R}$.

4 Computation results

4.1 SVD analysis

The singular value spectrum of $\mathbf{X}_1^{m-1}$ of the ski video is shown in Figure 1 (The racing video has similar result, which is omitted here). It is shown that the first singular value is the most dominant one (76.32%). We show the accumulated total proportion of the singular values in Figure 2 It is calculated that the first 100 singular values account for 93.67% of total singular values. Therefore, we think $r = 100$ is a good choice for the rank. Note that $r = 150$ is selected for the racing video after similar analyzing process.

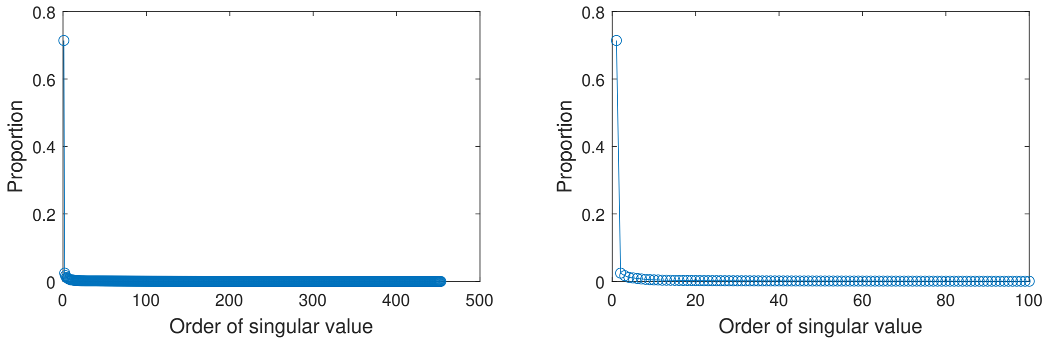

Figure 1: Singular value spectrum. Left figure shows all singular values, right figure shows the first 100 singular values

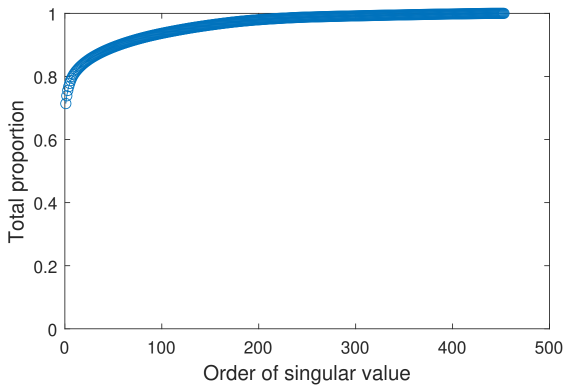

Figure 2: Accumulated proportion of the singular values

4.2 Background and foreground extraction of the ski video

For the ski video, there is only one eigenvalue satisfying the low-rank condition $\|\omega_p\| < \epsilon = 0.01$. The background and foreground extraction results are shown in Figure 3. There are two observations: First, the background is quite similar to the original video, which indicates that the rank that we selected is sufficient to represent the original information. However, the skier is also shown in the background. This is due to the fact that we added the negative part of the sparse matrix back to the low-rank matrix, which reconstruct the skier. If the skier is white color and the background is black, we believe there will be no such a reconstruction. Second, the foreground is almost dark, with several black spots. This is caused by the lack of contrast. To solve these two problems, we directly plot the low-rank reconstruction without adding back the negative sparse matrix $\mathbf{R}$. Meanwhile, we do not subtract the sparse matrix by the negative part. Instead, we directly add some positive values to every element of the sparse matrix calculated by $\mathbf{x}_{DMD}^{Sparse} = \mathbf{x} - |\mathbf{x}_{DMD}^{Low-Rank}|$, e.g., $\mathbf{x}_{DMD}^{Sparse} \leftarrow \mathbf{x}_{DMD}^{Sparse} + 150$. Although it is not guaranteed that the low-rank matrix and sparse matrix calculated in this way can be summed up to the original video matrix, the experiment result shows that this method can extract background and foreground properly. The results are shown in Figure 4. It can be shown that the skier has been eliminated from the background, while the skier is clearly shown in the foreground (as indicated in the red rectangle). It will be more obvious if one plot the stream of the video, please check the Github MATLAB to plot.

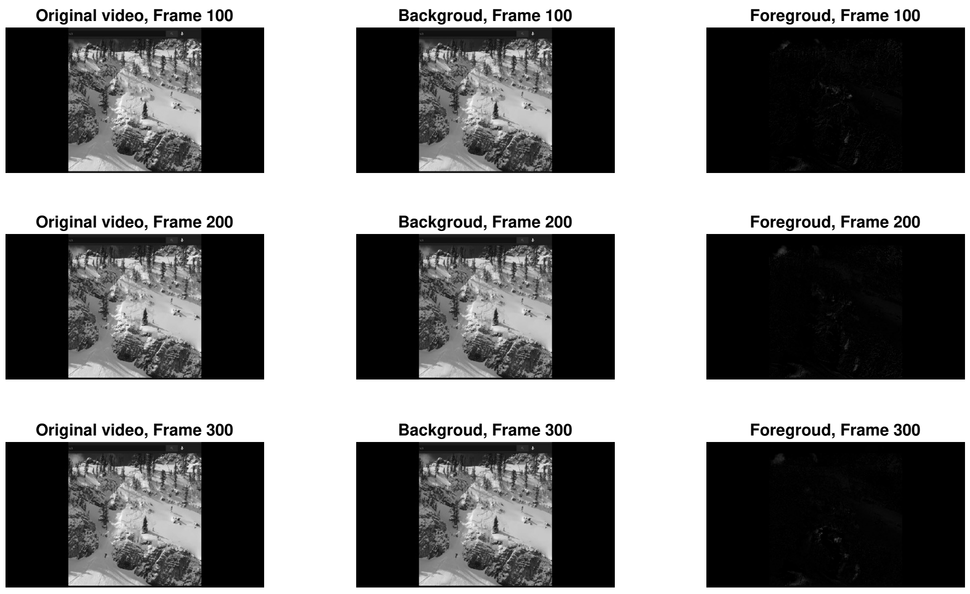

Figure 3: Original video, background, and foreground extraction using the instruction method

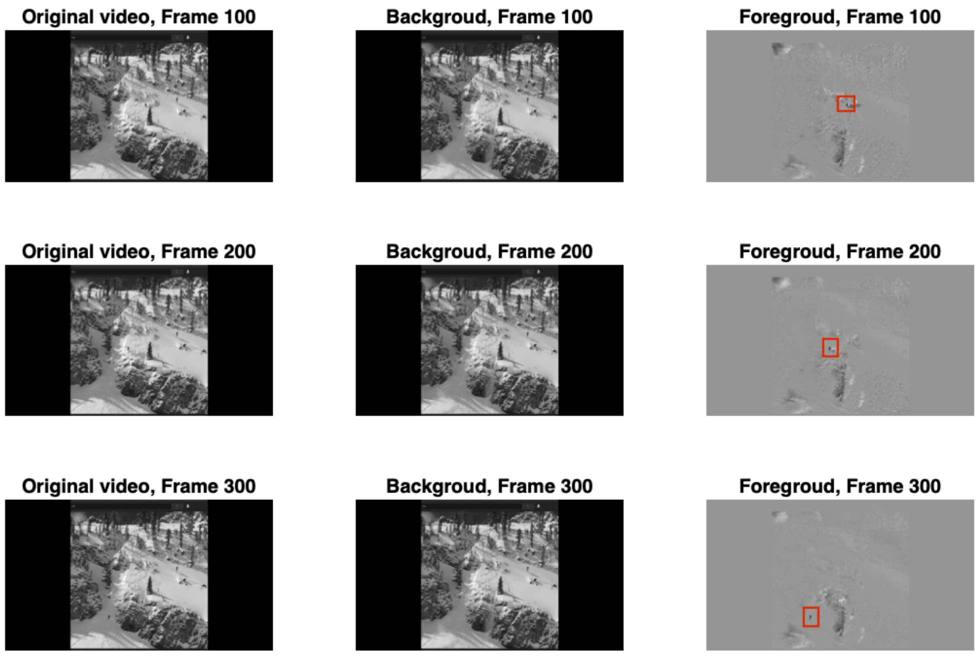

Figure 4: Original video, background, and foreground extraction add positive values to the sparse matrix

4.3 Background and foreground extraction of the racing video

Following the same process, we show the background and foreground extraction result in Figure 5. Note this figure shows the result calculated by the second method described in the above section. Please refer to the Github to plot the result calculated by the original method (i.e., the method described in the assignment instruction). It is shown that the background and foreground (the race car) are properly extracted.

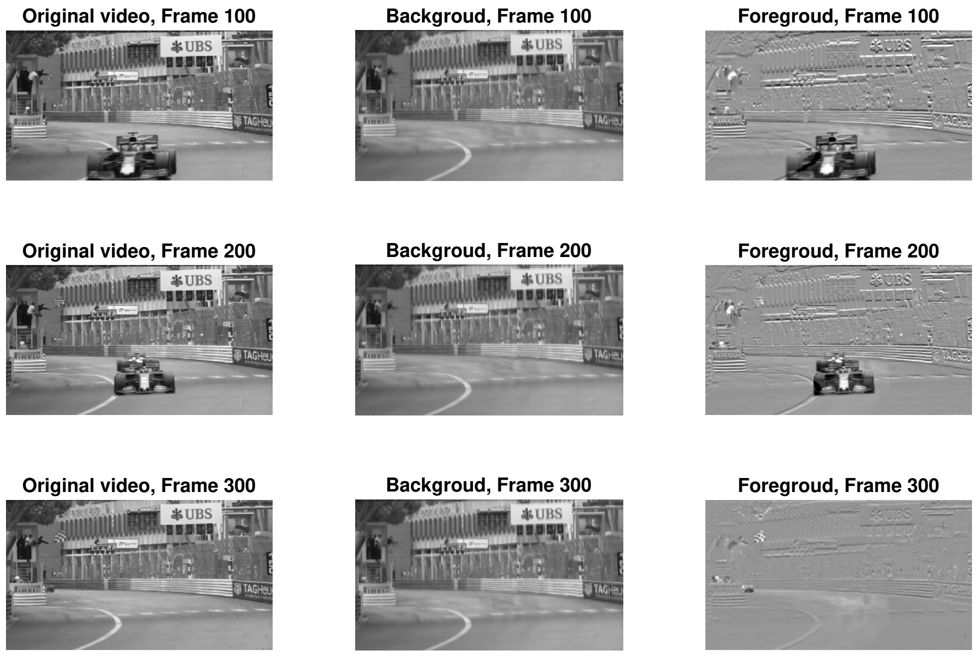

Figure 5: Original video, background, and foreground extraction add positive values to the sparse matrix

5 Summary and Conclusions

In this assignment, we designed algorithms to explore the DMD techniques on two video streams. We reshaped and constructed the time-dependent matrix of each video data. Then, we analyzed the SVD of each video data and designed the DMD algorithm to reconstruct the low-rank and sparse approximation, which was used to distinguish the background (low-rank) and the foreground (sparse) of the video. The numerical results showed that DMD could be well designed to well distinguish the background and foreground of dynamic videos. During this assignment, we have learned the power of DMD in reducing the dimension of video data, reconstructing the video data, extracting the background, and distinguishing the background and foreground of videos.

Appendix A GitHub repository

`https://github.com/Guoqq17/UW-AMATH-582`

Appendix B Key MATLAB Functions

- `B = reshape(A,sz1,...,szN)` reshapes `A` into a `sz1-by-...-by-szN` array where `sz1,...,szN` indicates the size of each dimension.
- `D = diag(v)` returns a square diagonal matrix with the elements of vector `v` on the main diagonal.
- `[U,S,V] = svd(A,'econ')` produces an economy-size decomposition of textttm-by-n matrix texttttA.
- `imshow(I)` displays the grayscale image `I` in a figure. `imshow` uses the default display range for the image data type and optimizes figure, axes, and image object properties for image display.
- `v = VideoReader(filename)` creates object `v` to read video data from the file named `filename`.
- `I = rgb2gray(RGB)` converts the truecolor image `RGB` to the grayscale image `I`. The `rgb2gray` function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.
- `[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`.
- `[y,Fs] = [row,col] = find(X>a)` returns the row and column subscripts of each element in array `X` which is larger than `a`.
- `Y = abs(X)` returns the absolute value of each element in array `X`. If `X` is complex, `abs(X)` returns the complex magnitude.

Appendix C MATLAB Codes

```matlab
close all; clear; clc;

%% import data and set up
% v=VideoReader('ski_drop_low.mp4');
v=VideoReader('monte_carlo_low.mp4');

% convert data to matrix format
width=v.Width;
height=v.Height;
images=zeros(width*height, v.NumberOfFrames);
for i=1:1:v.NumberOfFrames
    cur_img_temp=rgb2gray(read(v,i));
%      cur_img=reshape(cur_img_temp(:,233:728),width*height,1);
    cur_img=reshape(cur_img_temp,width*height,1);
    images(:,i)=double(cur_img);
end

% create time series
t_temp=linspace(0,v.CurrentTime, v.NumberOfFrames+1);
t=t_temp(1:end-1);
dt=t(2)-t(1);

%% DMD
% SVD
X1=images(:,1:end-1);
X2=images(:,2:end);
[U,Sigma,V]=svd(X1,'econ');
figure(1)
subplot(1,2,1)
plot(diag(Sigma)/sum(diag(Sigma)), '-o')
xlabel('Order of singular value','Fontsize',12)
ylabel('Proportion','Fontsize',12)

subplot(1,2,2)
```

```
plot(diag(Sigma(1:100, 1:100))/sum(diag(Sigma)), '-o')
xlabel('Order of singular value','Fontsize',12)
ylabel('Proportion','Fontsize',12)

temp=diag(Sigma)/sum(diag(Sigma));
acc_p=zeros(1,length(temp));
for i=1:1:length(temp)
    acc_p(i)=sum(temp(1:i));
end

figure(2)
plot(acc_p, '-o')
xlabel('Order of singular value','Fontsize',12)
ylabel('Total proportion','Fontsize',12)
ylim([0,1])
% low-rank approximation
r=100; % define the rank based on the singular value spectrum
U2=U(:,1:r);
Sigma2=Sigma(1:r,1:r);
V2=V(:,1:r);
S=U2'*X2*V2/Sigma2; % low rank
[eV,D]=eig(S);
% Phi=X2*V2/Sigma2*eV; % DMD modes
Phi=U2*eV; % DMD modes

lambda=diag(D);
omega=log(lambda)/dt;
bg = find(abs(omega)<1e-2);
omega_bg = omega(bg); % background eigenvalues
Phi_bg = Phi(:,bg); % DMD background mode

% reconstruct low-rank DMD
x1=X1(:,1);
y0=Phi_bg\x1;
x_modes=zeros(length(omega_bg),length(t));
for iter=1:length(t)
    x_modes(:,iter)=(y0.*exp(omega_bg*t(iter)));
end
X_dmd=Phi_bg*x_modes;

%% sparse method 1
% X_sparse=images-abs(X_dmd);
% R=X_sparse.*(X_sparse<0);
% X_dmd=R+abs(X_dmd);
% X_s_dmd=X_sparse-R;

%% sparse method 2
X_sparse=images-abs(X_dmd);
X_s_dmd=X_sparse + 150;
%% show results
% for report (3 snapshots)
snaps=[100,200,300];
figure(3)
for i=1:1:length(snaps)
    subplot(3,3,(i-1)*3+1)
    img1=reshape(uint8(images(:,snaps(i))),height,width);
    imshow(img1)
    title(['Original video, Frame ', num2str(snaps(i))])
    subplot(3,3,(i-1)*3+2)
    img2=reshape(uint8(X_dmd(:,snaps(i))),height,width);
    imshow(img2)
    title(['Backgroud, Frame ',num2str(snaps(i))])
    subplot(3,3,(i-1)*3+3)
    img3=reshape(uint8(X_s_dmd(:,snaps(i))),height,width);
    imshow(img3)
    title(['Foregroud, Frame ', num2str(snaps(i))])
end

```

```
% for viewing (all snapshots)
% figure(4)
% for ind_show=1:v.numberOfFrames
%     subplot(1,3,1)
%     img1=reshape(uint8(images(:,ind_show)),height,width);
%     imshow(img1)
%     title('Original video')
%     subplot(1,3,2)
%     img2=reshape(uint8(X_dmd(:,ind_show)),height,width);
%     imshow(img2)
%     title('Backgroud extracted by Low-rank DMD')
%     subplot(1,3,3)
%     img3=reshape(uint8(X_s_dmd(:,ind_show)),height,width);
%     imshow(histeq(img3))
%     title('Foregroud extracted by sparse DMD')
%     drawnow
% end
```