Background Subtraction in Video Streams

Yiling Sun

AMATH 482 HW01 Winter 2019

abstract

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

1. Introduction and Overview

Dynamic Mode Decomposition is a powerful technique for the discovery of the low-dimensional spatio-temporal coherent structures from high-dimensional data. It primarily enable the tasks of diagnostics of current data, future state prediction and control. In this report, the basics application on data diagnostics is explored in the case of separating the background and foreground in video Streams. We demonstrates its performance with three examples of video, videos of dogs, human walking in the hallway and the tennis balls game. In the video of dogs, three white dogs run across the grass. The whole scene is bright and the dog is the brightest spot. In the videos of human walking in hallway, the scene is relatively dark and the walking girl is the darkest spot. In the video of tennis ball game, this scene is still bright. We have two sets of moving object, player in white clothes and in yellow. White is brighter than yellow.

2. Theoretical Background

Dynamic Mode Decomposition

Dynamic Mode Decomposition(DMD) is essential a ideal combination of spatial dimensionality-reduction techniques, such as the proper orthogonal decomposition and Fourier Transforms in time. Applying DMD on data x, it algorithmically is a regression of complex dynamic onto the linear system

$$\frac{dx}{dt} = \mathcal{A}x, \tag{1}$$

or in discrete-time system

$$x_{k+1} = Ax_k \tag{2}$$

where x is the original data with each column representing data from one time frame, while in x_{k+1}, as the definition of derivative, each column is replaced by the column at its right, as time moving forward. Their relation is given by $A = exp(\mathcal{A}\Delta t)$. We take a low-rank eigen-decomposition of matrix A that fits the data with minimal $||x_{k+1} - Ax_k||_2$. An important feature of DMD is that we usually does not directly work on matrix A, but on its projection on POD mode given by SVD. Since the dimension of data would be much higher than the number of columns(data set), so is the matrix A. Hence, it is hard 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 A, and b_k are the coordinates of x(0) in the eigenvector basis. It contributes to the construction of original data and the prediction of future state. We call Φ = X'VΣ^{-1}W, the exact DMD mode, where X' is data in the second time frame, VΣ^{-1} is the transpose of data on POD mode and W is the eigenvectors of A. To translate from discrete to continuous, we take $\omega_k = ln(\lambda_k)/\Delta t$. Then, we can reconstruct data or make state prediction by plug the eigenvectors ϕ, eigenvalue ω and b_k back to the solution of (1),

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

Hence, the eigenvalue ω_k and the related $exp(\omega_k t)$ determine the frequency of the mode. If the frequency is 0, the mode is static.

Basically, DMD is a data-driven, equation-free method to provide sptio-temporal decomposition and make short-time future state prediction and control. Its has a relatively robust application on diagnostics than prediction and control.

3. Algorithm Implementation and Development

Prepare Data

Firstly, command **VideoReader()** is used to read in the mp4 format video. In case of too large video pixels in a frame, I may use **imresize()** to truncate it. Based on the number of frames, I set the time span that each frame represent one second and the Δt would be 1. Then, frame by frame, I read each frame in black and white and into a column with command **reshape()** in the shape :

$$X = \begin{bmatrix} X_1 & \cdots & X_n \end{bmatrix}. \tag{5}$$

To prepare for DMD, we translate matrix X into two matrices,

$$X1 = \begin{bmatrix} X_1 & \cdots & X_{n-1} \end{bmatrix}, X2 = \begin{bmatrix} X_2 & \cdots & X_n \end{bmatrix}. \tag{6}$$

SVD and DMD

Firstly, X1 need to be truncated with dominant mode given by SVD. The rank of truncation is determined by the singular values given by the command `svd(X,'econ')`. Then, we can get the more manageable data

$$X_{svd} = U\Sigma V^*. \tag{7}$$

By $X'_{svd} \approx AX_{svd}$, we can get $A = X'_{svd}X^{\dagger}_{svd}$, where $X^{\dagger}$ is the pseudo-inverse of X. Combining it with (6), we get $A = X'_{svd}V\Sigma^{-1}U^*$. For the convenience, we project A onto the POD given by SVD,

$$\tilde{A} = U^*AU = U^*X'_{svd}V\Sigma^{-1} \tag{8}$$

We find the eigen-decomposition of Ã by command `[W,D]=eig(A)`. The W and D returned are the eigenvectors and eigenvalues of A. Then, as I state in Section 2, the exact DMD mode is given by $\Phi = X'_{svd}V\Sigma^{-1}W$. We translate the eigenvalues into continuous time domain by command `log()` and divide it by Δt, which is 1 in this case. With the ω, we can separate the high frequency foreground with larger ω and low frequency background with ω close to zero. Then, we can get the low-rank DMD background by adding up mode given by (4) with smaller ω. The foreground can be extracted by taking the low rank DMD background out from the original data. So, command `sort()` is used to rank the omega in increasing order. Then, we can choose how many omega to use to construct the background.

Handle the complex term in reconstructed data

After the above implement, original data can be represented as X= $\underbrace{b_p\varphi_p e^{\omega_p t}}_{background} + \underbrace{\sum_{j\neq p} b_j\varphi_j e^{\omega_j t}}_{foreground}$. Although these two terms would add up to be real value, individual is complex due to its complex terms of DMD construction. So, in the last step of extracting background, the only way to get real value foreground is $X_{fg} = X - |X_{bg}|$. However, it may result in negative value in X_{fg}. The negative pixels is not reasonable. In order to get rid of these negative residuals. We take the negative values from the foreground out and add them back to the background.

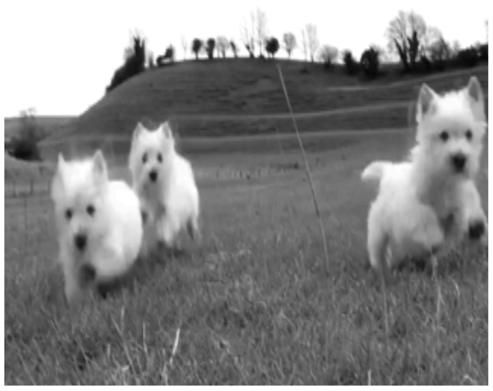
4. Computational Results

Case 1: Dog video

Figure 1 gives a snapshot of video at time 110 and the energy plot of modes from SVD. The first mode is prominent capturing 56% dynamic of original data and the 2nd and 3rd mode has around 4%, others are almost zero. So I tried the rank one, two and three truncation on the original data.

Figure 2 gives the result of rank one truncation. Because the result matrix A is 1x1, there is only one omega that associated with the low frequency background. The snapshot at left still contains the shape of background, while after the residuals processing, the snapshot at right gives the more accurate result that get rid of all the background.

In Figure 3, the two upper snapshots gives the result with rank two truncation. In the right snapshot, the background is given by the minimal omega. Although the background is black, there are still light spot.



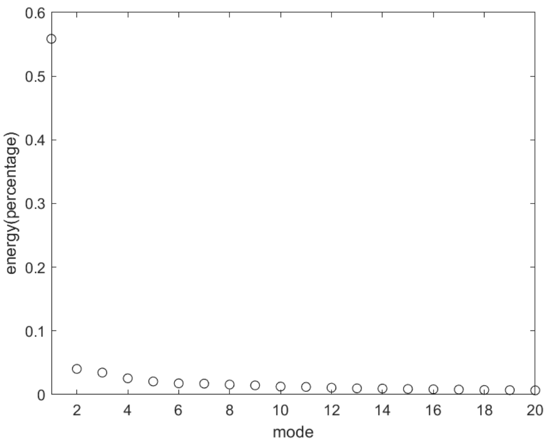


Figure 1: the snapshot at time 110 from the video of dogs and the energy plot of first 20 modes of data

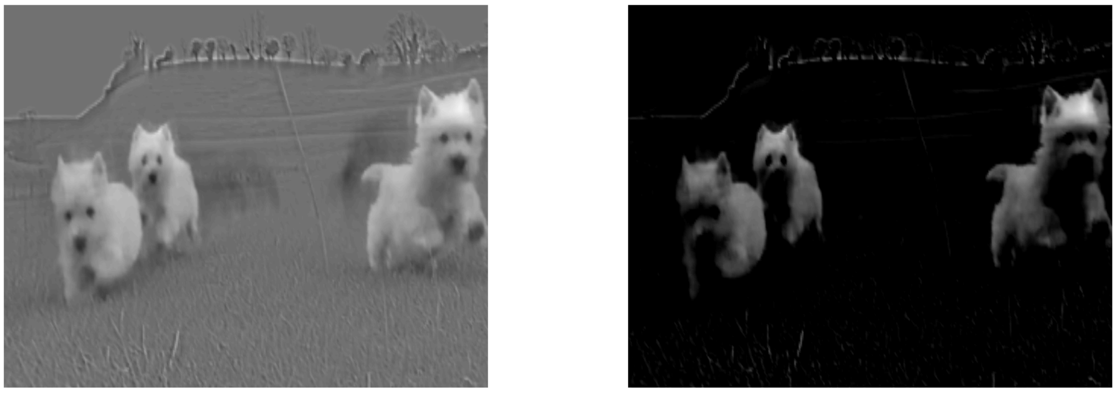


Figure 2: Two snapshot at time 110 from the foreground video without and with residuals processing(left to right). Both are the result from rank one truncation.

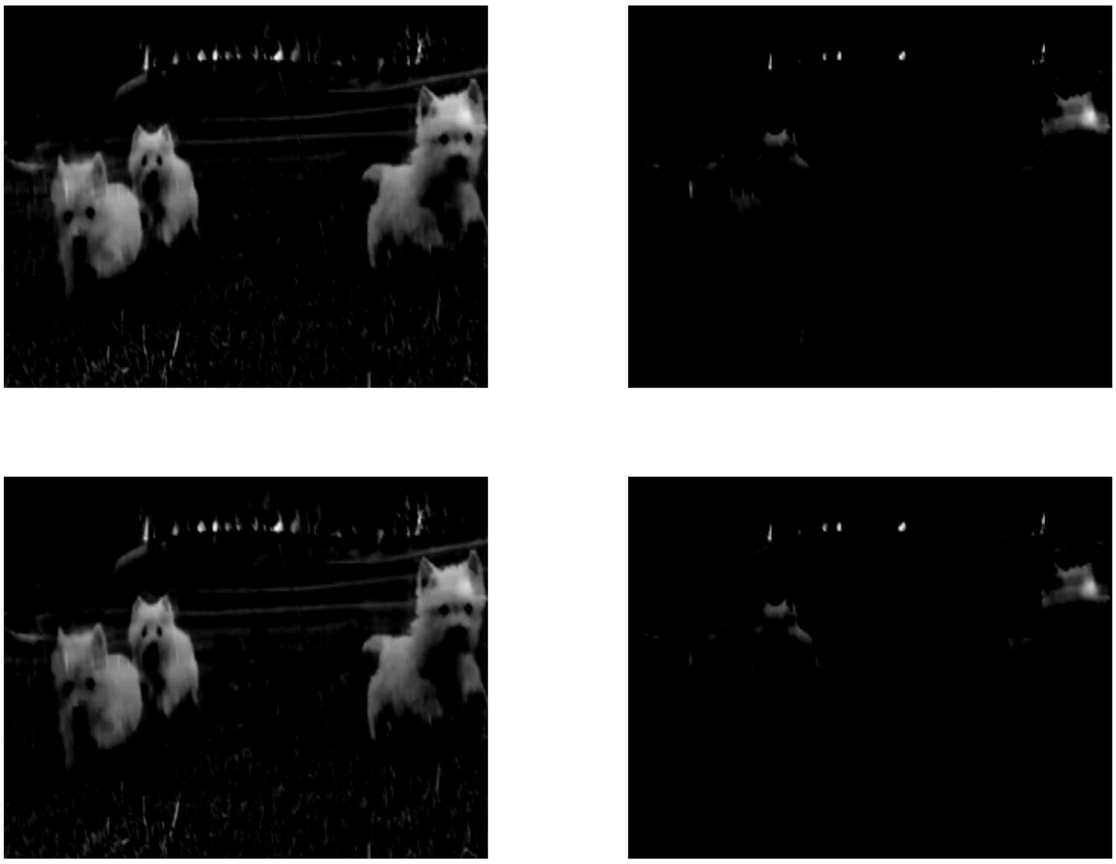
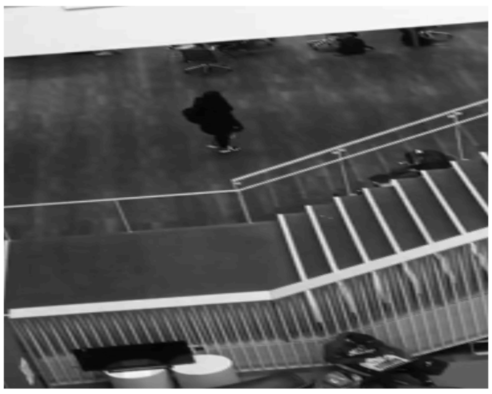


Figure 3: Four snapshot at time 110 from four different cases. Upper two snapshots are with rank two truncation and with one omega, two omega(left to right). Lower are with rank three truncation and with one omega, two omega(left to right). All cases eliminate the residuals.



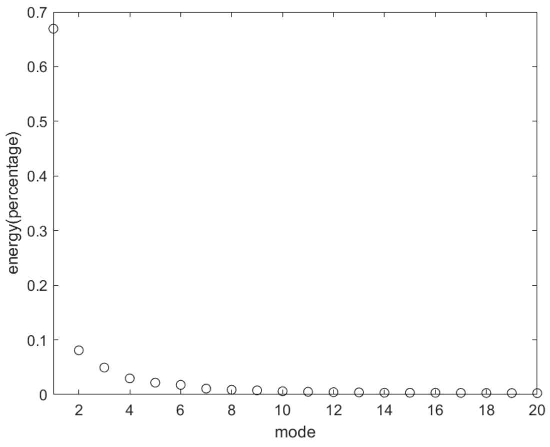


Figure 4: the snapshot at time 110 from the video of human walking in hallway and the energy plot of first 20 modes of data.

The left snapshot is given by the two omega together. It does not extract foreground successfully. It implies that data after rank two truncation still involves the background. The lower two snapshots basically give the same result with rank three truncation.
In general, the rank one truncation on matrix A with the only one omega gives the best result of extracting foreground.

Case 2: Human walking

Figure 4 gives a snapshot of video at time 65 and the energy plot of modes from SVD. The first mode is prominent capturing 68% dynamic of original data and the 2nd around 9%, 3rd around 5% and others are almost zero. So I tried the rank one and rank three truncation on the original data.

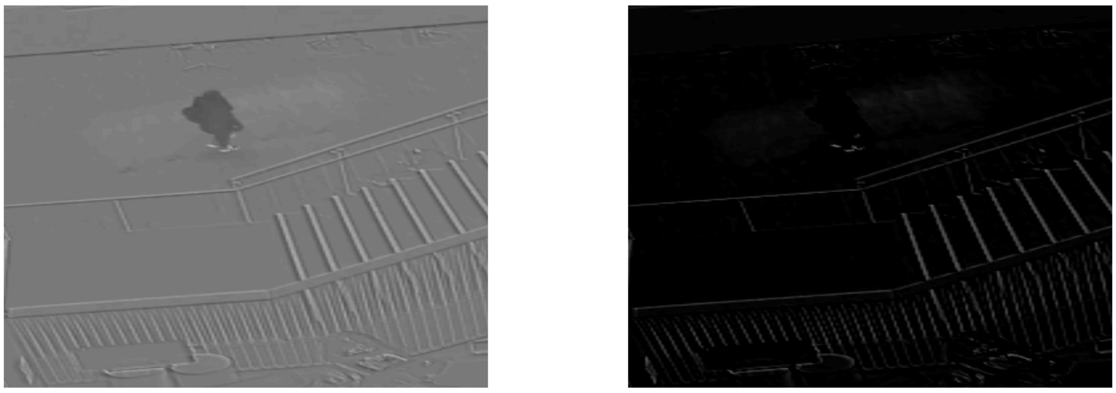


Figure 5: Two snapshot at time 65 from the foreground video without and with residuals processing(left to right). Both are the result from rank one truncation.

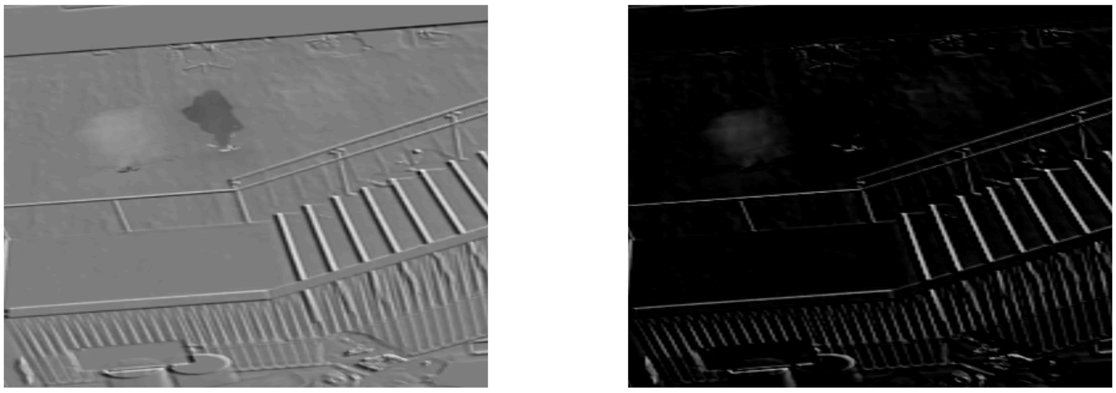


Figure 6: Two snapshot at time 65 from the foreground video with and without residuals processing. Both are the result from rank three truncation with one minimal omega.

Figure 5 gives the snapshot of foreground at time 65 with rank one truncation and one omega. The snapshot at left, even though still containing background, gives the motion of human walking, while there are basically nothing at the snapshot at left. The residuals processing does not work well in this case.

Figure 6 shows the case of rank three truncation. This case is even worse with more prominent background and bad foreground information.

In general, the DMD does not work well on separating background and foreground in this case.

Case 3: tennis balls game

Figure 7 gives the snapshot at time 100 from the video of tennis balls game and again the energy plot. We observe that this video has the same characteristics of the video of dogs, Light moving object in the

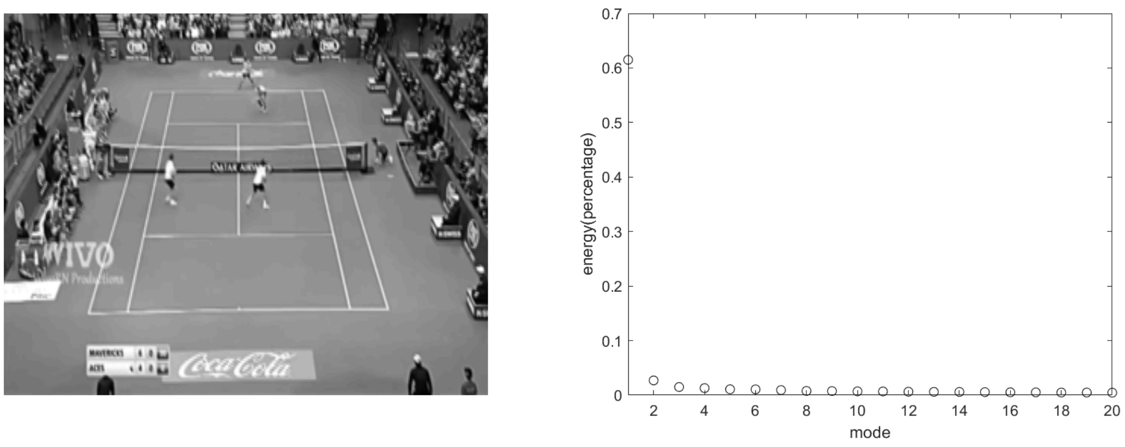


Figure 7: the snapshot at time 100 from the video of tennis balls game and the energy plot of first 20 modes of data.

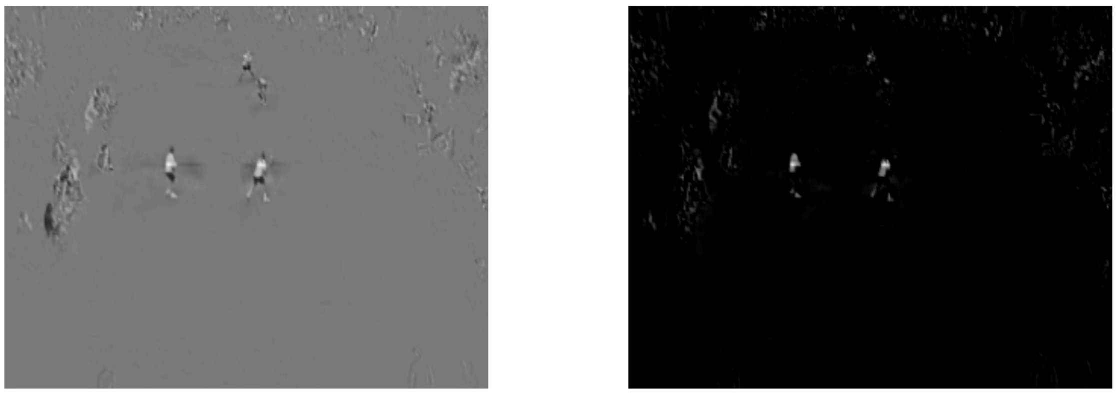


Figure 8: Two snapshot at time 100 from the foreground video of the tennis balls game without and with residuals processing. Both are the result from rank one truncation.

foreground and dark background. Also, it has the only one dominant mode capturing around 60% energy. So, I use the same parameters that gives the best result in case 1, rank one truncation. Although the moving tennis player is small, generally it gives a good result. Figure 8 shows the snapshot at time 100. We observe that when we eliminate the residuals, parts of the players in yellow clothes blend into the background.

5. Summary and Conclusions

Generally, with good data (no much noise), DMD can give a nice separation of background and foreground if the moving object is relatively brighter than the background. The brighter the object is, the better result of background subtraction we could get. Moreover, eliminating the negative pixels is a effective method to improve the quality of the background subtraction result.

6. Appendix A MATLAB functions used and brief implementation explanation

v = VideoReader(filename) creates object v to read video data from the file named filename.[1]

video = readFrame(v) reads the next available video frame from the file associated with v.[2]

B = imresize(A,scale) returns image B that is scale times the size of A. [3]

[**U,S,V**] = svd(A,'econ') [U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix A. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, S, along with the columns in either U or V that multiply those zeros in the expression A = U*S*V'. Removing these zeros and columns can improve execution time and reduce storage requirements without compromising the accuracy of the decomposition.[4]

x = diag(A) returns a column vector of the main diagonal elements of A.[5]

e = eig(A) returns a column vector containing the eigenvalues of square matrix A.[6]

Y = log(X) returns the natural logarithm ln(x) of each element in array X.[7]

B = sort(A) sorts the elements of A in ascending order.[8]

Y = abs(X) returns the absolute value of each element in array X. If X is complex, abs(X) returns the complex magnitude.[9]

[1]Description from MathWorks
[2]Description from MathWorks
[3]Description from MathWorks
[4]Description from MathWorks
[5]Description from MathWorks
[6]Description from MathWorks
[7]Description from MathWorks
[8]Description from MathWorks
[9]Description from MathWorks

7. Appendix B MATLAB codes

```
clear all; close all; clc;

video = VideoReader('gou.mp4');
data = [];
while hasFrame(video)
    ori_frame = readFrame(video);
    frame = imresize(ori_frame,0.5);
    bw = double(rgb2gray(frame));
%     pcolor(flipud(bw)), shading interp
%     colormap(gray); drawnow; hold on;
    frame_data = reshape(bw,[],1);
    data = [data frame_data];
end
%%
X1 = data(:,1:end-1);
X2 = data(:,2:end);
[u,s,v]=svd(X1,'econ');
energy = diag(s)/sum(diag(s));
plot(energy,'ko');
set(gca,'Xlim',[1 20]);
xlabel('mode');ylabel('energy(percentage)');
saveas(gcf,'wangqiuenergy.png');

%%
r=1;
ur = u(:,1:r);
sr = s(1:r,1:r);
vr = v(:,1:r);
Altilde = ur'*X2*vr\sr;
[W,D]=eig(Altilde);
phi = X2*vr/sr*W;
lamda = diag(D);
omega = log(lamda);
% figure()
% plot(omega,'.'),hold on;
% refline([0 0]);
% line([0 0],[2,-2]);
[a,ind]=sort(abs(omega));
```

```
%%
x1 = data(:,1);
b = phi\x1;
time_dynamics = zeros(r,size(data,2));
t = linspace(1,1,size(data,2));
n_omega = 1;
total_bg = 0;
for i = 1:n_omega
    for j = 1:size(data,2)
        time_dynamics(:,j) = (b.*exp(omega(ind(i))*t(j)));
    end
    bg_dmd = phi*time_dynamics;
    total_bg = total_bg + bg_dmd;
end
bg =  abs(total_bg);

fg = data - bg;
%%
resi = fg;
resi(resi>0) = 0;
fg_new = fg - resi;

%%
for i = 1:size(data,2)
    frame_dmd = data(:,i);
    frame_dmd_reshape = reshape(frame_dmd,size(bw,1),size(bw,2));
    pcolor(flipud(frame_dmd_reshape)), shading interp
    axis off;
    colormap(gray); drawnow; hold on;
    if i == 110
        saveas(gcf,'gou.png')
    end
end
```