Principal Component Analysis

Yiling Sun

AMATH 482 HW03 Winter 2019

abstract

Principle Component Analysis(PCA) is a widely used tool in modern data analysis. In this report, we introduce the basic mechanism of PCA and how it relate to the Singular Value Decomposition(SVD). We also explore the performance of PCA with redundancy and noise case in a paint-releasing experiments.

1. Introduction and Overview

Theoretically, Principle Component Analysis can extract important information from unknown, but potentially low-dimensional data. In this report, we test its performance on removing the redundancy and noise by an paint-releasing experiment.

Data is given by a paint released experiment. There are totally four cases and for each case, there are three cameras record the motion. The first test is the ideal case, the paint is released vertically, so the entire motion is in the z direction with simple harmonic motion. The second test is the same experiment in first test but with noise given by the shaking camera. In the third test, the paint is released off center. So there is not only motion in the z direction but also in x-y plane. The forth case is more complex that it involves rotation of the paint and also pendulum motion.

2. Theoretical Background

2.1 Principle Component Analysis

Principal Component Analysis is widely used non-parameter method to deal with unknown, but potentially low-dimensional data. It can reduce the complex data set to lower dimension to reveal the dynamic underlie it. To begin, the collected data should be standardized and normalized before hand and arranged in this

form:

$$X = \begin{bmatrix} X_a \\ Y_a \\ X_b \\ Y_b \\ X_c \\ Y_c \end{bmatrix} \tag{1}$$

where each row represents one measurements. Data can be viewed as a basis and the number of measurements is the dimension of this basis. Then, we find the covariance between all possible pairs of measurements by

$$C_x = \frac{1}{n-1} X X^T, \tag{2}$$

where C_x is a square and symmetric matrix. We apply method of diagonalization on C_x, eigenvalue decomposition or singular value decomposition usually used. Then, we can find a new orthogonal basis, which is a linear combination of the original basis, that can best present our data. each vector in this basis is called principle component or proper orthogonal mode(POD) due to the orthogonality of the basis. We define the energy of each POD by the proportion of its corresponding singular value divided by the sum of all singular value. We assume that the direction with the largest energy is the most dominant direction, which contains the most important dynamic of the system.

Notably, by doing the PCA, we have made assumptions on the linearity of the system, the statistical sufficiency of the mean and variance.

2.2 SVD

Single Value decomposition(SVD) is a tool usually carried out when we do the Principle Component Analysis. SVD can diagonal all matrices and provide their eigenvalues and orthonormal bases.

The basics of SVD is matrix transformation. It basically rotates and stretches, or compresses, a given set of vectors, x, by the matrix A, which often defined as **Ax=b**. For example, in 2 dimensional, the rotation matrix is given by

$$A = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}, \tag{3}$$

where θ is the angle to rotate. Also, the rotation matrix A is a unitary matrix, $A^{-1} = A^T$. The scaling matrix is given by

$$A = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}, \tag{4}$$

which is a diagonal matrix and α donates the scalar. The combination of the above two matrices control the rotation and scaling in a two-dimensional vector space.

Based on above, Singular value decomposition can factorize matrix into a number of constitutive components all of which have a specific meaning. More concretely, SVD of a function A is

$$A_{m*n} = U_{m*m}\Sigma_{m*n}V_{n*n}, \tag{5}$$

where U and V are orthonormal bases and Σ is diagnal matrix, of which the diagonal entries are nonnegative and ordered from largest to smallest.

When we apply the SVD(equation 5) to the covariance matrix (equation 2), we can get

$$C_x = \frac{1}{n-1}\Sigma^2, \tag{6}$$

where Σ^2 is our desired principle component.

3. Algorithm Implementation and Development

I use basically the same strategy to collect and prepare data and to do the Principle Component Analysis for all cases.

Prepare Data

I load the given data of video in matrix and go over the video by **pcolor** and in **colormap(gray)**. To track the paint, firstly, I turn dark the environment to get rid of distraction from other light distraction by set environment to be zero, since we work on gray colormap. Then, I find the value of the brightest point on the frame and set a threshold to extract the area with certain brightness below the largest(the extracted area would be within the paint). After that, I take the average position of the extracted area as the location of the paint. I can get the trajectory of the paint by applying this strategy on each frame. The threshold may vary for different case to get the best trajectory.

For the first case and first camera, the matrix size is 480*640*3*226, which means there are 226 frames. So

we get the location of the light spots in every frames in **forloop** and this is the trajectory of the paint. After we get the data of trajectory from three camera in one case, since the video may begin at different time, I align three sets of data by align the last oscillation. Then I enforce them into same length, the minimum length among three. Finally, I get three sets of data of a motion almost happened at the same time and with same length in time.

Principle Component Analysis

To do PCA, I put data into the form we mentioned in section 2.1 (1) and subtract the means from each row to normailze data. Then I diagonalize data with **[u,s,v]=svd(M,'econ')** command to the reduced SVD. The returned matrix **s** is the diagonal matrix that contains the singular values. We can calculate its energy by dividing the singular value by the sum of singular values. At last, I take product of matrix of singular value and the orthonormal bases v to see the final result from PCA analysis for the motion in each cases.

4. Computational Results

Ideal Case

Figure 1 is the plot of the vertical and horizontal information in time span. There is no significant displacement in horizontal direction. The small fluctuation should come from the measurement error. The displacements on vertical direction represent the motion of simple harmonic oscillation.

Figure 2 is the energy plot of each mode in the new basis. The clear dominance of the first mode is easily deduced. The first mode captures 76% of the dynamic, not as high as we expected for the simple motion oscillation in one dimension. The measurement error in the horizontal direction should explain to this reduc-

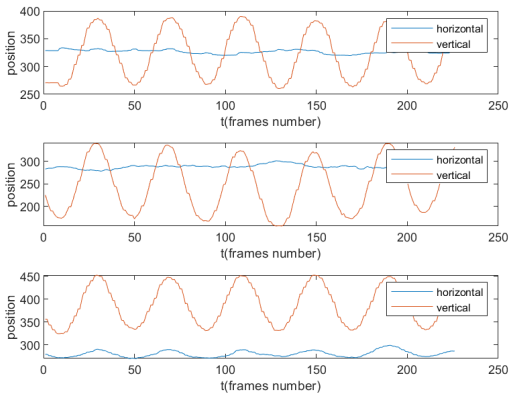

Figure 1: Data collected from case1

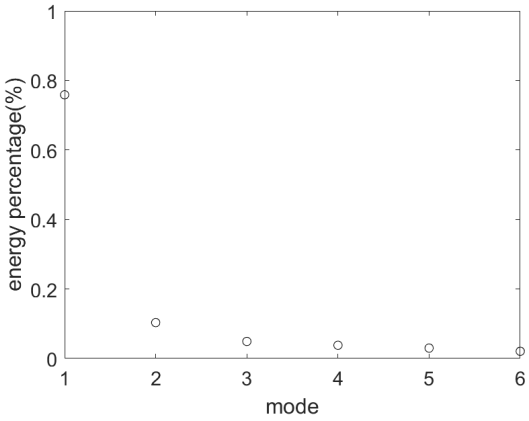

Figure 2: The energy in percentage of each mode in case1

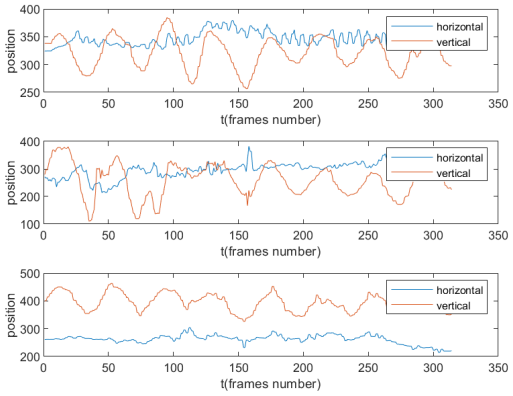

Figure 3: Data collected from case2

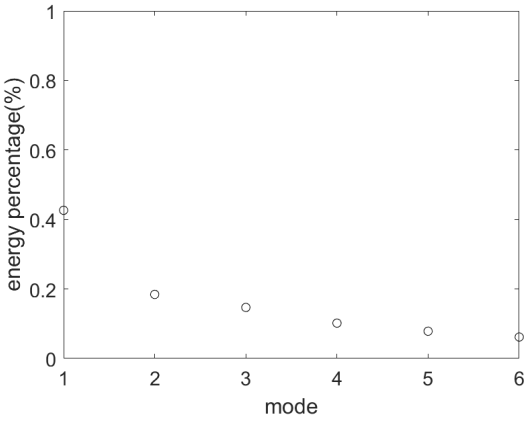

Figure 4: The energy in percentage of each mode in case2

tion and the energy in the second mode. Thus, the first mode tells us the most interesting dynamic, others redundancies. In general, the Principle Component Analysis do a good job in this case.

Noise Case

Though the cameras still record the simple harmonic oscillation in vertical direction, the data contains lots of noisy information in horizontal and vertical direction, as we see in Figure 3.

In the energy plot, Figure 4, the energy of the first mode is still higher than others, but not as prominent as that in Figure 2. The first modes captures only 52% of the dynamic. Consequently, it need to take four modes to capture 90% of dynamic.

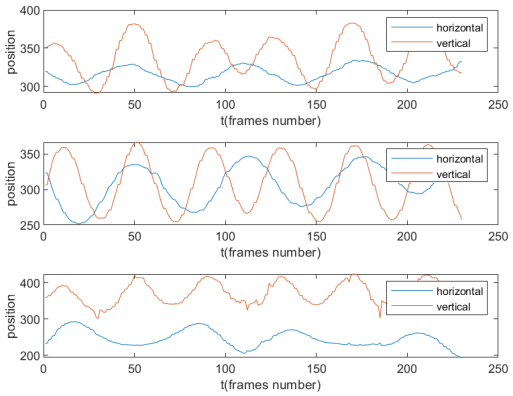

Figure 5: Data collected from case3

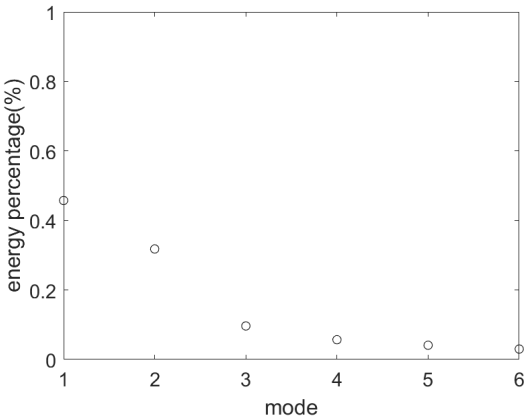

Figure 6: The energy in percentage of each mode in case3

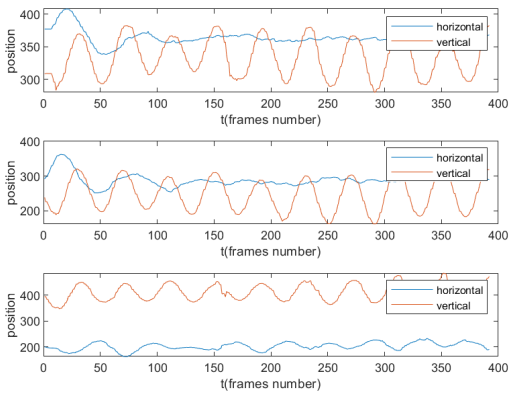

Figure 7: Data collected from case4

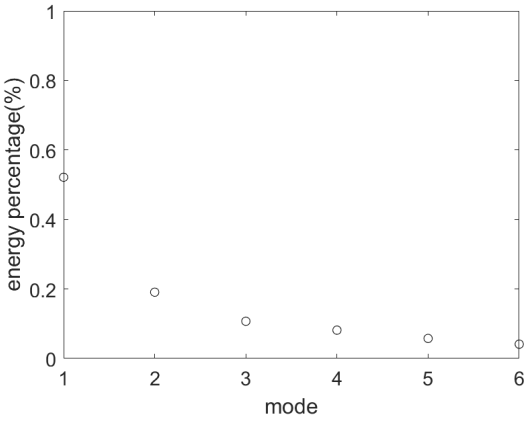

Figure 8: The energy in percentage of each mode in case4

Horizontal Case

In Figure 5, we get the good oscillation in both vertical and horizontal direction. We observe the different pattern of the horizontal displacement from three data set, possibly because the camera recorded from different angles. Also, there are still some unavoidable noise due to the error of measurements.

From Figure 6, we can find the first two modes have relatively large energy, 46% for the first mode and 32% for the second mode. It reasonable because we have motion in two dimension. The difference in the energy of these two modes is possibly due to the difference between the range of the motion in vertical and horizontal. And, there is still a little bit energy in third mode due to the noise. We can basically treat the last four modes as redundancies. In general, the Principle Component Analysis do a good job in this case.

Rotation Case

In Figure 7, though there is rotation of the paint, we cannot find any evidence of rotation in data set. still the displacement in vertical direction gives only oscillation on plot. For the horizontal direction, there is only prominent displacement at the beginning of the experiment. Though it was released off center, the pendulum motion was reduced due to the rotation of the paint. So, there is no significant data after the first oscillation in horizontal direction. Nevertheless, the third camera may record from aside, so the third data set has only small oscillation.

In Figure 8, the first mode captures 52% of the dynamic while the second capture 19%. The energy of second mode comes from the pendulum motion, not so significant. And also, PCA does not give us any information on the rotation. Thus, it does not work well in this case.

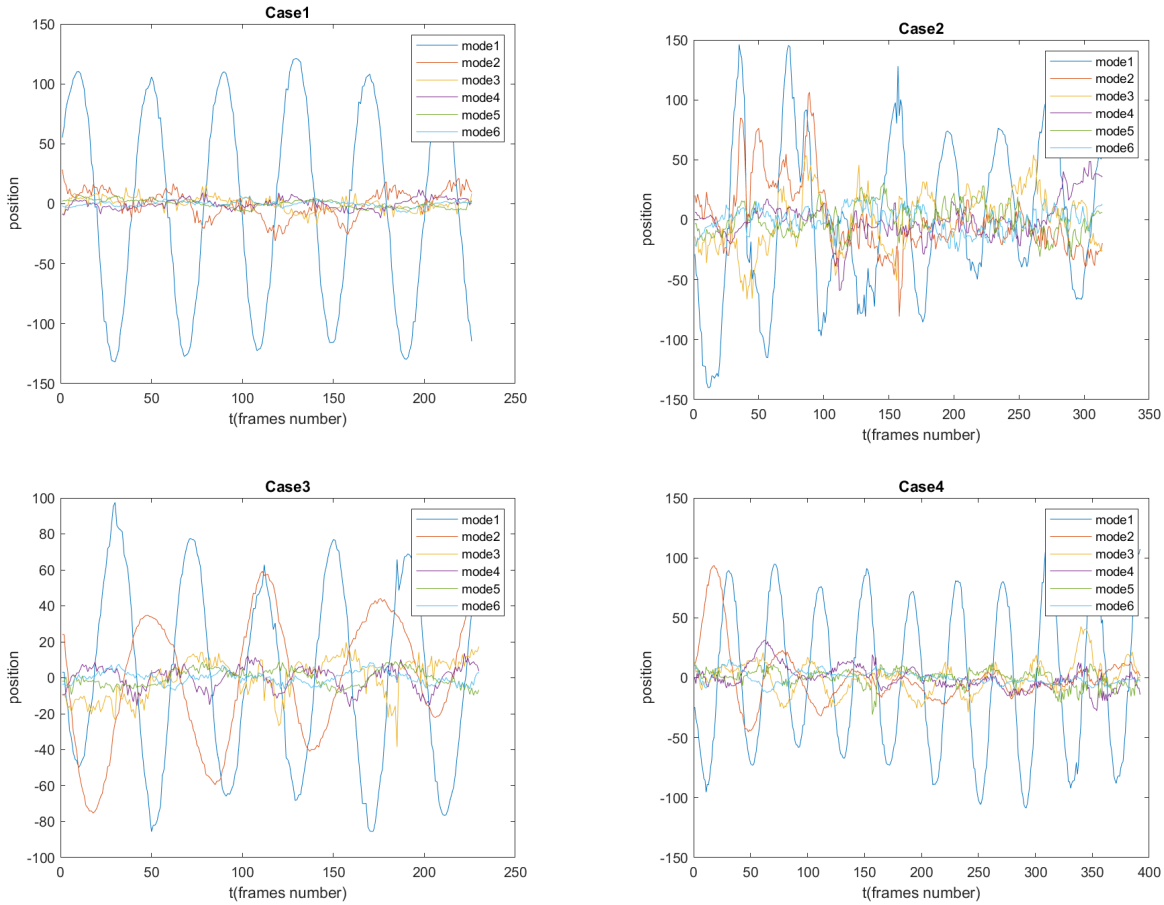

Figure 9: Dynamic captured by each mode in four cases

Final Result of Experiments

Figure 9 gives plots of dynamic captured by each mode for four cases. All the results fit our expectation in above analysis.

5. Summary and Conclusions

The ideal case and horizontal case show the ability of PCA on dimension reduction. It well extract the important information from redundant data with little noise due to the error of measurement. We almost can get the equation of the displacement to describe the motion in video. While in the noisy case, the performance of PCA goes down. We conclude that the noise negatively influences the performance of PCA. Thus, we should try to reduce the noise in data to get better result from PCA. The forth case brings out the weakness of non-parametric method. Blindly applying PCA causes the missing of complex but significant motion, in this case the rotation of the paint. This problem surely should to be taken into consideration before we do the Principle Component Analysis.

6. Appendix A MATLAB functions used and brief implementation explanation

[**sz1,...,szN**] = size(A) returns the length of each dimension of A.[1]

M = max(A) returns the maximum elements in vector A.[2]

M = min(A) returns the minimum elements in vector A.[3]

Y = double(X) converts the values in X to double precision.[4]

I = rgb2gray(RGB) converts the truecolor image RGB to the grayscale intensity image I. [5]

k = find(X) returns a vector containing the linear indices of each nonzero element in array X.[6]

M = mean(A) returns the mean of the elements in A.[7]

[**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.[8]

x = diag(A) returns a column vector of the main diagonal elements of A.[9]

S = sum(A) returns the sum of the elements of A. [10]

[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
[10]Description from MathWorks

7. Appendix B MATLAB codes

```
clear all; close all; clc;
for i = 1:3
  load(['cam' num2str(i) '_1.mat'])
end
%% Ideal Case
%1
[xlength,ylength,c,length] = size(vidFrames1_1);
loc1_1 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames1_1(:,:,:,j)));
    frame(1:480,[1:250 450:680])=0;
    frame(1:150, 250:450)=0;
    M = max(frame(:));
    [x1,y1] = find(frame ≥ M * 9 / 10);
    loc1_1(j,1)=mean(y1);
    loc1_1(j,2)=mean(x1);
    imshow(vidFrames1_1(:,:,:,j)); hold on;
    plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;
%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end

%%
[xlength,ylength,c,length] = size(vidFrames2_1);
loc2_1 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames2_1(:,:,:,j)));
    frame(1:480,[1:250 400:ylength])=0;
    frame(1:100, 200:350)=0;
    M = max(frame(:));
    [x1,y1] = find(frame ≥ M * 19 / 20);
    loc2_1(j,1)=mean(y1);
    loc2_1(j,2)=mean(x1);
    imshow(vidFrames2_1(:,:,:,j)); hold on;
    plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;
%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end
```

```
39
40 %%
41 [xlength,ylength,c,length] = size(vidFrames3_1);
42 loc3_1 = zeros(length,2);
43 for j = 1:length
44     frame = double(rgb2gray(vidFrames3_1(:,:,:,j)));
45     frame(1:xlength,[1:250 500:ylength]) = 0;
46     frame([1:230 350:xlength],250:500) = 0;
47     M = max(frame(:));
48     [x1,y1] = find(frame ≥ M * 9 / 10);
49     loc3_1(j,1)=mean(x1);
50     loc3_1(j,2)=mean(y1);
51     imshow(vidFrames3_1(:,:,:,j)); hold on;
52     plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;
53 %     pcolor(frame), shading interp
54 %     colormap(gray); drawnow; hold on;
55 end
56 %%
57 x1=loc1_1(:,1);
58 x2=loc2_1(:,1);
59 x3=loc3_1(:,1);
60 y1=loc1_1(:,2);
61 y2=loc2_1(:,2);
62 y3=loc3_1(:,2);
63
64 % figure()
65 % plot(y1), hold on
66 % plot(y2)
67 % plot(y3)
68
69 x2 = x2(52:size(x2));
70 y2 = y2(52:size(y2));
71 % figure()
72 % plot(y1), hold on
73 % plot(y2)
74 % plot(y3)
75 siz = [size(y1,1) size(y2,1) size(y3,1)];
76 minisiz = min(siz);
77 M = [x1(1:minisiz) y1(1:minisiz) x2(1:minisiz) y2(1:minisiz) x3(1:minisiz) y3(1:minisiz)];
78 M = M';
79 %%
80 figure()
```

```
t=1:minisiz;
subplot(3,1,1)
plot(t,M(1,:),t,M(2,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,2)
plot(t,M(3,:),t,M(4,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,3)
plot(t,M(5,:),t,M(6,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')
saveas(gcf,'signal1.png')
%% Minus average
[m,n]=size(M); % compute data size
mn=mean(M,2); % compute mean for each row
M=M-repmat(mn,1,n); % subtract mean

figure()
plot(M(2,:)),hold on
plot(M(4,:))
plot(M(6,:))
xlabel('t(frames number)'); ylabel('position')
saveas(gcf,'avesig1.png')
%%
[u,s,v]=svd(M);
sig=diag(s)/sum(diag(s));
figure()
plot(linspace(1,6,6),sig,'ko')
set(gca,'Ylim',[0 1],'Fontsize',[14])
xlabel('mode'); ylabel('energy percentage(%)')
saveas(gcf,'diag1.png')

figure()
plot(t,s*v');
title('Case1');
xlabel('t(frames number)'); ylabel('position');
legend('mode1','mode2','mode3','mode4','mode5','mode6');
saveas(gcf,'result1.png')
```

```
clear all; close all; clc;
for i = 1:3
  load(['cam' num2str(i) '_2.mat'])
end

%% 1
[xlength,ylength,c,length] = size(vidFrames1_2);
loc1_2 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames1_2(:,:,:,j)));
    frame(1:450,[1:300 450:ylength])=0;
    frame(1:200,300:450)=0;
    M = max(frame(:));

    [x1,y1] = find(frame ≥ M * 9 / 10);
    loc1_2(j,1)=mean(y1);
    loc1_2(j,2)=mean(x1);

%     imshow(vidFrames1_2(:,:,:,j)); hold on;
%     plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;
%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end

%%
[xlength,ylength,c,length] = size(vidFrames2_2);
loc2_2 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames2_2(:,:,:,j)));
    frame(1:450,[1:180 440:ylength])=0;
    M = max(frame(:));

    [x1,y1] = find(frame ≥ M * 29 / 30);
    loc2_2(j,1)=mean(y1);
    loc2_2(j,2)=mean(x1);

%     imshow(vidFrames2_2(:,:,:,j)); hold on;
%     plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;

```

```
%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end

%%
[xlength,ylength,c,length] = size(vidFrames3_2);
loc3_2 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames3_2(:,:,:,j)));
    frame(1:xlength,[1:280 500:ylength])=0;
    frame([1:180 350:xlength],280:500)=0;
    M = max(frame(:));
    [x1,y1] = find(frame ≥ M * 9 / 10);
    loc3_2(j,1)=mean(x1);
    loc3_2(j,2)=mean(y1);

%     imshow(vidFrames3_2(:,:,:,j)); hold on;
%     plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;

%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end
%%
x1=loc1_2(:,1);
x2=loc2_2(:,1);
x3=loc3_2(:,1);
y1=loc1_2(:,2);
y2=loc2_2(:,2);
y3=loc3_2(:,2);

% figure()
% plot(y1,'r'), hold on
% plot(y2,'g')
% plot(y3,'b')
%
% x1=x1((20:size(x1)));
% y1=y1((20:size(y1)));

x2=x2((25:size(x2)));
y2=y2((25:size(y2)));

x3=x3((5:size(x3)));
```

```
y3=y3((5:size(y3)));
% figure()
% plot(y1,'r'), hold on
% plot(y2,'g')
% plot(y3,'b')
siz = [size(y1,1) size(y2,1) size(y3,1)];
minisiz = min(siz);
M = [x1(1:minisiz) y1(1:minisiz) x2(1:minisiz) y2(1:minisiz) x3(1:minisiz) y3(1:minisiz)];
M = M';
%%
figure()
t=1:minisiz;
subplot(3,1,1)
plot(t,M(1,:),t,M(2,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,2)
plot(t,M(3,:),t,M(4,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,3)
plot(t,M(5,:),t,M(6,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')
saveas(gcf,'signal2.png')
%% Minus average
[m,n]=size(M); % compute data size
mn=mean(M,2); % compute mean for each row
M=M-repmat(mn,1,n); % subtract mean

figure()
plot(M(2,:)),hold on
plot(M(4,:))
plot(M(6,:))
xlabel('t(frames number)'); ylabel('position')
saveas(gcf,'avesig2.png')
%%
[u,s,v]=svd(M,'econ');
sig=diag(s)/sum(diag(s));
figure()
```

```
plot(linspace(1,6,6),sig,'ko')
set(gca,'Ylim',[0 1],'Fontsize',[14])
xlabel('mode'); ylabel('energy percentage(%)')
saveas(gcf,'diag2.png')

figure()
plot(t,s*v');
title('Case2');
xlabel('t(frames number)'); ylabel('position');
legend('mode1','mode2','mode3','mode4','mode5','mode6');
saveas(gcf,'result2.png')
```

```
clear all; close all; clc;
for i = 1:3
  load(['cam' num2str(i) '_3.mat'])
end

%%
[xlength,ylength,c,length] = size(vidFrames1_3);
loc1_3 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames1_3(:,:,:,j)));
    frame(1:480,[1:280 400:ylength])=0;
    frame(1:220, 280:400)=0;
    M = max(frame(:));
    [x1,y1] = find(frame ≥ M * 49 / 50);
    loc1_3(j,1)=mean(y1);
    loc1_3(j,2)=mean(x1);

%     imshow(vidFrames1_3(:,:,:,j)); hold on;
%     plot(y1,x1, 'ro'); hold off; drawnow;

%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;

end
%%
[xlength,ylength,c,length] = size(vidFrames2_3);
loc2_3 = zeros(length,2);
for j = 1:length
```

```
    frame = double(rgb2gray(vidFrames2_3(:,:,:,j)));
    frame(1:xlength,[1:210 450:ylength])=0;
    frame(1:170, 210:450)=0;
    M = max(frame(:));

    [x1,y1] = find(frame ≥ M * 49 / 50);
    loc2_3(j,1)=mean(y1);
    loc2_3(j,2)=mean(x1);

%     imshow(vidFrames2_3(:,:,:,j)); hold on;
%     plot(y1,x1, 'ro'); hold off; drawnow;

%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end
%%
[xlength,ylength,c,length] = size(vidFrames3_3);
loc3_3 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames3_3(:,:,:,j)));
    frame(1:480,[1:200 480:ylength])=0;
    frame([1:100 350:xlength], 200:500)=0;
    M = max(frame(:));
    [x,y] = ind2sub([xlength ylength],I);
    [x1,y1] = find(frame ≥ M * 49 / 50);
    loc3_3(j,1)=mean(x1);
    loc3_3(j,2)=mean(y1);


%     imshow(vidFrames3_3(:,:,:,j)); hold on;
%     plot(y1,x1, 'ro'); hold off; drawnow;

%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end

%%
x1=loc1_3(:,1);
x2=loc2_3(:,1);
x3=loc3_3(:,1);
y1=loc1_3(:,2);
y2=loc2_3(:,2);
```

```
y3=loc3_3(:,2);
%
% figure()
% plot(y1,'r'), hold on
% plot(y2,'b')
% plot(y3,'g')

x1=x1((10:size(x1)));
y1=y1((10:size(y1)));

x2=x2((35:size(x2)));
y2=y2((35:size(y2)));
%
% figure()
% plot(y1,'r'), hold on
% plot(y2,'b')
% plot(y3,'g')

siz = [size(y1,1) size(y2,1) size(y3,1)];
minisiz = min(siz);
M = [x1(1:minisiz) y1(1:minisiz) x2(1:minisiz) y2(1:minisiz) x3(1:minisiz) y3(1:minisiz)];
M = M';
%%
figure()
t=1:minisiz;
subplot(3,1,1)
plot(t,M(1,:),t,M(2,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,2)
plot(t,M(3,:),t,M(4,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,3)
plot(t,M(5,:),t,M(6,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')
saveas(gcf,'signal3.png')
%% Minus average
[m,n]=size(M); % compute data size
```

```
mn=mean(M,2); % compute mean for each row
M=M-repmat(mn,1,n); % subtract mean

figure()
plot(M(2,:)),hold on
plot(M(4,:))
plot(M(6,:))
saveas(gcf,'avesig3.png')
%%
[u,s,v]=svd(M,'econ');
sig=diag(s)/sum(diag(s));
figure()
plot(linspace(1,6,6),sig,'ko')
set(gca,'Ylim',[0 1],'Fontsize',[14])
xlabel('mode'); ylabel('energy percentage(%)')
saveas(gcf,'diag3.png')

figure()
plot(t,s*v');
title('Case3');
xlabel('t(frames number)'); ylabel('position');
legend('mode1','mode2','mode3','mode4','mode5','mode6');
saveas(gcf,'result3.png')
```

```
clear all; close all; clc;
for i = 1:3
  load(['cam' num2str(i) '_4.mat'])
end

%%
[xlength,ylength,c,length] = size(vidFrames1_4);
loc1_4 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames1_4(:,:,:,j)));
    frame(1:480,[1:300 500:ylength])=0;
    frame(1:200, 300:500)=0;
    M = max(frame(:));

    [x1,y1] = find(frame ≥ M * 49 / 50);
    loc1_4(j,1)=mean(y1);
```

```
    loc1_4(j,2)=mean(x1);

%     imshow(vidFrames1_4(:,:,:,j)); hold on;
%     plot(y1,x1, 'ro'); hold off; drawnow;
% %
%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end

%%
[xlength,ylength,c,length] = size(vidFrames2_4);
loc2_4 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames2_4(:,:,:,j)));
    frame(1:480,[1:200 450:ylength])=0;
    frame([1:120 400:xlength] , 200:450)=0;
    M = max(frame(:));
    [x1,y1] = find(frame ≥ M * 49 / 50);
    loc2_4(j,1)=mean(y1);
    loc2_4(j,2)=mean(x1);

%     imshow(vidFrames2_4(:,:,:,j)); hold on;
%     plot(y1,x1, 'ro'); hold off; drawnow;

%     pcolor(frame), shading interp
%     colormap(gray); drawnow; hold on;
end

%%
[xlength,ylength,c,length] = size(vidFrames3_4);
loc3_4 = zeros(length,2);
for j = 1:length
    frame = double(rgb2gray(vidFrames3_4(:,:,:,j)));
    frame(1:xlength,[1:300 500:ylength])=0;
    frame([1:130 300:xlength], 300:500)=0;
    M = max(frame(:));
    [x1,y1] = find(frame ≥ M * 19 / 20);
    loc3_4(j,1)=mean(x1);
    loc3_4(j,2)=mean(y1);
%
%     imshow(vidFrames3_4(:,:,:,j)); hold on;
%     plot(mean(y1),mean(x1), 'ro'); hold off; drawnow;
```

```
59
60  %     pcolor(frame), shading interp
61  %     colormap(gray); drawnow; hold on;
62  end
63  %%
64  x1=loc1_4(:,1);
65  x2=loc2_4(:,1);
66  x3=loc3_4(:,1);
67  y1=loc1_4(:,2);
68  y2=loc2_4(:,2);
69  y3=loc3_4(:,2);
70
71  % figure()
72  % plot(y1), hold on
73  % plot(y2)
74  % plot(y3)
75
76  x2=x2((10:size(x2)));
77  y2=y2((10:size(y2)));
78  %
79  % figure()
80  % plot(y1), hold on
81  % plot(y2)
82  % plot(y3)
83
84  %%
85  siz = [size(y1,1) size(y2,1) size(y3,1)];
86  minisiz = min(siz);
87  M = [x1(1:minisiz) y1(1:minisiz) x2(1:minisiz) y2(1:minisiz) x3(1:minisiz) y3(1:minisiz)];
88  M = M';
89
90  %%
91  figure()
92  t=1:minisiz;
93  subplot(3,1,1)
94  plot(t,M(1,:),t,M(2,:))
95  xlabel('t(frames number)'); ylabel('position')
96  legend('horizontal','vertical','Location','NorthEast')
97
98  subplot(3,1,2)
99  plot(t,M(3,:),t,M(4,:))
100 xlabel('t(frames number)'); ylabel('position')
```

```
legend('horizontal','vertical','Location','NorthEast')

subplot(3,1,3)
plot(t,M(5,:),t,M(6,:))
xlabel('t(frames number)'); ylabel('position')
legend('horizontal','vertical','Location','NorthEast')
saveas(gcf,'signal4.png')
%% Minus average
[m,n]=size(M); % compute data size
mn=mean(M,2); % compute mean for each row
M=M-repmat(mn,1,n); % subtract mean

figure()
plot(M(2,:)),hold on
plot(M(4,:))
plot(M(6,:))
saveas(gcf,'avesig4.png')
%%
[u,s,v]=svd(M,'econ');
sig=diag(s)/sum(diag(s));
figure()
plot(linspace(1,6,6),sig,'ko')
set(gca,'Ylim',[0 1],'Fontsize',[14])
xlabel('mode'); ylabel('energy percentage(%)')
saveas(gcf,'diag4.png')

figure()
plot(t,s*v');
title('Case4');
xlabel('t(frames number)'); ylabel('position');
legend('mode1','mode2','mode3','mode4','mode5','mode6');
saveas(gcf,'result4.png')
```