Homework 3: Principal Component Analysis

Jiayi Wang

Abstract

In this report, we will see illustration of various aspects of the Principal Component Analysis (PCA) and its practical usefulness and the effects of noise on the PCA algorithms in a real-world problem by extracting the motion information of a paint can in different cases.

Introduction and Overview

Imagine we have a paint can hanging on a spring and we have four experiment cases here. 12 movie files are provided from three different cameras called **camN_M.mat** with N=1,2,3 indicates the three cameras and M=1,2,3,4 indicates the case number. In the first case, there is a small displacement of the mass in the z direction and the entire motion is in the z direction with simple harmonic motion. In the second case, we will repeat the first ideal case experiment; however, this time we will introduce camera shake into video recording. In the third case, the mass is released off-center to produce motion in the x-y plan as well as z direction; there is both a pendulum motion and simple harmonic oscillations. In the last case, the mass is released off-center and rotates to produce motion in the x-y plane, rotation, and motion in the z direction. We will performance PCA on removing the redundancy and noise and extract important information from data.

Theoretical Background

- **Principal component analysis (PCA)**
  PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. To check for covariance, the appropriate covariance matrix is then:

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T \tag{1}$$

where the matrix X contains the experimental data of the system. In particular, $X \in C^{m\times n}$ where m is the number of probes or measuring positions, and n is the number of experimental data points taken at each location.[1] Then we can find the new orthogonal basis, which is a linear combination of original basis. Each vector in this basis is called a proper orthogonal mode (POD). To calculate the energy each mode capture, it is simply just the square of its corresponding singular value divided by the sum of the squares of all singular values.

- **Singular Value Decomposition (SVD)**
  A second method for diagonalizing the covariance matrix is the SVD method. In this case, the SVD can diagonalize any matrix by working in the appropriate pair of bases U and V as outlined in the first lecture of this section. Thus, by defining the transformed variable:

$$\mathbf{Y} = \mathbf{U}^*\mathbf{X} \tag{2}$$

where U is the unitary transformation associated with the SVD: X = UΣV^*. Just as in the eigenvalue/eigenvector formulation, we then compute the variance in Y:[2]

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{\Sigma}^2 \tag{3}$$

Algorithm Implementation and Development

For case 1, first we would load **cam1_1.mat**, **cam2_1.mat**, **cam3_1.mat** to matlab using *load()*command. Then we will use *size()* command to find the frames numbers of each videos. For each video, we would turn each frame into grayscale by using *rgb2gray()* and *double()* command. In order to track the light on the paint can which is very bright, we would set the environment of frames to be zero in grayscale since there might be some other spots which are also very bright but nothing to do with the light on the paint can we need to track. Next, we will set a threshold for extracting the area that contains the brightest spot. Now, with the help of *max()* and *find()* command, we are able to track the location of

[1] Page 404 from course notes.
[2] Page 406 from course notes.

the paint can in each time. We then average these locations and save in the matrix we generated in advance.

After we obtained the three matrixes with location information, we will need to keep them in sync since the paint cans in three videos don't start to move at the same time. In order to do that, we would change the first frames to be the frame which corresponding with the lowest y coordinate. Next, we than truncate each matrix to have the same frame number of the shortest frame length. At last, we would add the x and y coordinates from each of the three matrixes to a new matrix, and we would like to transpose this matrix for the columns to represent each time frame. We would save this matrix in a variable called *bigX*. Then we are able to generate three plots of the position of pain can in each camera.

To apply PCA, we first would like to normalize the data by subtracting the means of each row from each entry. Then with the help of [U,S,V]=svd() command, we are able to apply the PCA on data. To obtain the singular value, we can use the command diag(S) and assign it to a variable called sig. Now, in order to illustrate the energy captured of each mode compare to the total energy, we simply use the command sig.^2/sum(sig.^2) to implement it.

For the other three cases, we will use the same strategy to apply PCA.

Computational Results

Case 1: From the plots of paint can's position from each camera in figure 1(left), we can clearly see that the paint can is doing simple harmonic motion in the z direction. There is no significant displacement in x-y plane. From the energy captured plot in figure 1(top right), we see that the first mode captures about 73% of the total energy; the first two modes capture about 95%. It thus suggests that the data can be easily represented by a two-mode expansion. This also can be proved by the displacement across principal component direction plot in figure 1(bottom right). However, the energy captured by the first mode is not as high as we would expect since the mass is doing a simple harmonic motion.

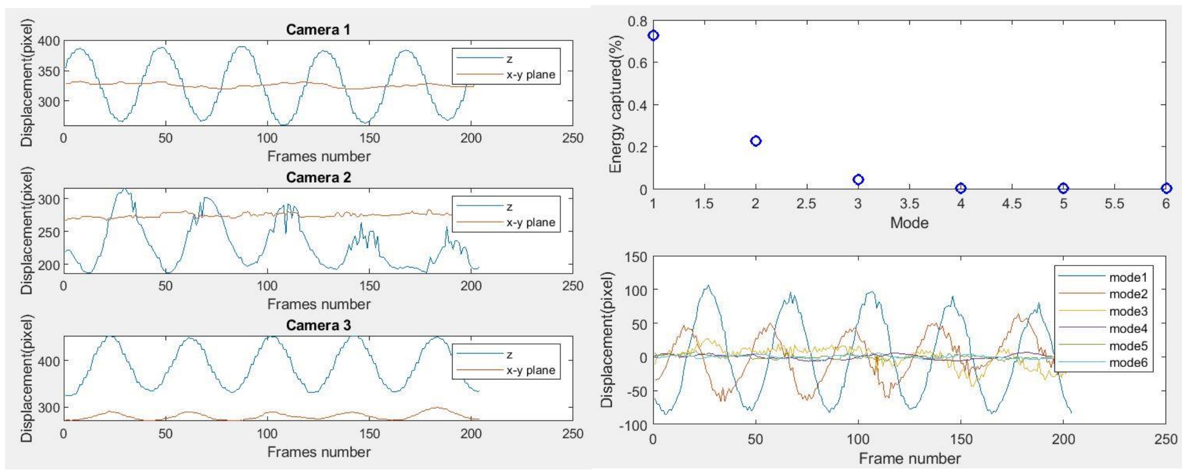

Figure 1: Paint can position which are captured by three cameras in case 1(left)

Energy captured in percentage of each mode in case 1(top right)

Displacement across principal component direction in case 1(bottom right)

<u>Case 2: </u>From the plots of paint can's position from each camera in figure 2(left), we can still see that the paint can is doing simple harmonic motion in the z direction especially under the first camera when shake is the least amount within these three cameras. In the x-y plan, we cannot see any significant movement but noise, which is mainly due to the camera shaking. This matches the description of the case. From the energy captured plot in figure 2(top right), we see that the first three modes capture about 90% of the total energy. It thus suggests that the data can be easily represented by a three-mode expansion. This also can be proved by the displacement across principal component direction plot in figure 2(bottom right). Although this is the same system as in the first case, but due to the significate noise, the energy captured of each mode are much lower than in the first case. Thus, we will need more modes expansion to accurately represent the data.

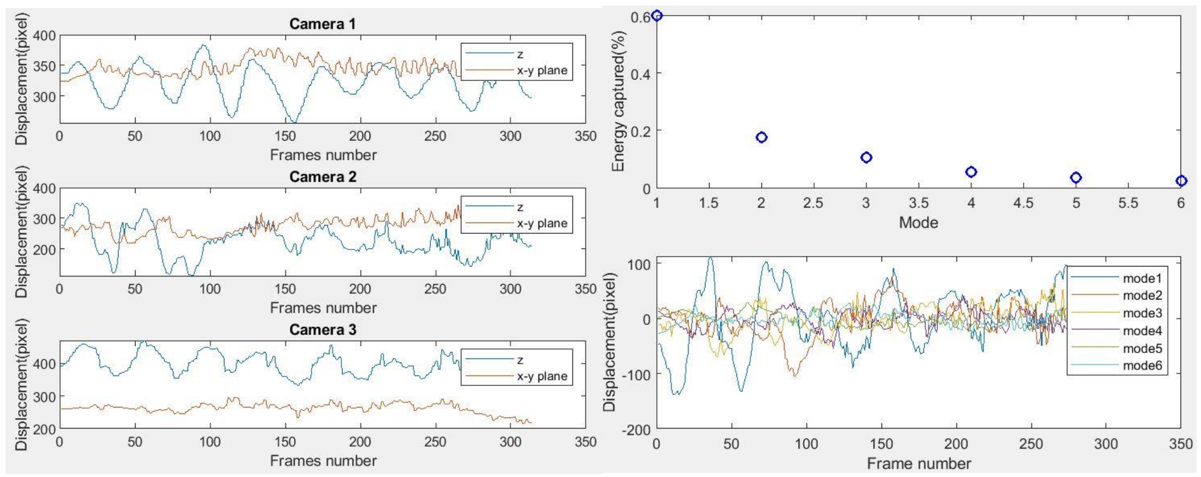

Figure 2: Paint can position which are captured by three cameras in case 2(left)

Energy captured in percentage of each mode in case 2(right)

Displacement across principal component direction in case 2 (bottom right)

Case 3: From the plots of paint can's position from each camera in figure 3(left), we can still see that the paint can is not only doing simple harmonic motion in the z direction, but also pendulum motion in x-y plane. This matches the description of the case. From the energy captured plot in figure 3(top right), we see that the first mode captures about 46%, first two modes capture 84% and first three modes capture 95.9% of the total energy, which means data can be represented by a simple three-mode expansion. This also can be proved by the displacement across principal component direction plot in figure 3(bottom right).

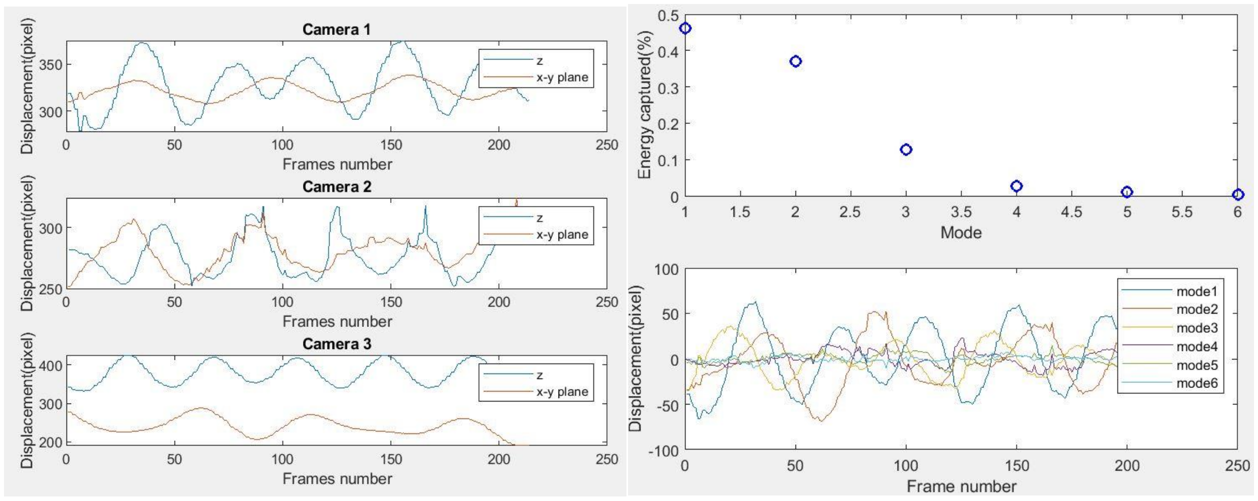

Figure 3: Paint can position which are captured by three cameras in case 3(left)
Energy captured in percentage of each mode in case 3(right)
Displacement across principal component direction in case3(bottom right)

Case 4: From the plots of paint can's position from each camera in figure 4(left), we can still see that the paint can is doing simple harmonic motion in the z direction. However, the motion in the x-y plane and the rotational motion cannot be observed. We believe it is due to the rotational motion reduces the pendulum motion and the flashlight cannot be tracked the whole time due to the rotation. Therefore, PCA does not work well in this case.

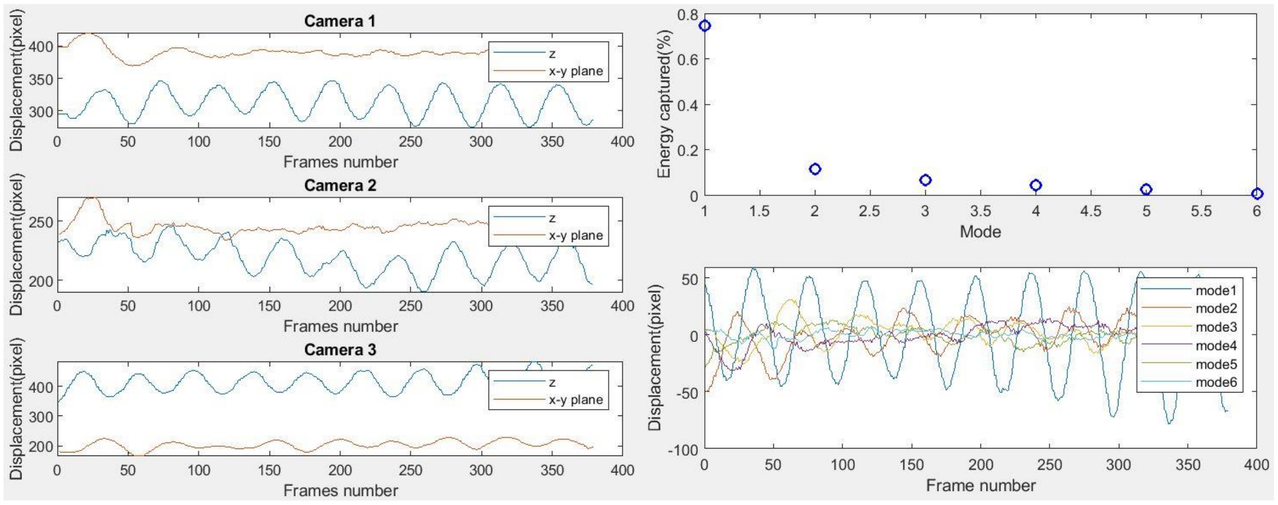

Figure 4: Paint can position which are captured by three cameras in case 4(left)

Energy captured in percentage of each mode in case 4(right)

Displacement across principal component direction in case 4(bottom right)

Summary and Conclusions

In this report, four different scenarios are being analyzed to extract the location information of a paint can. By performing PCA on our data, we are able to see how many significant, principal components exist. Plots of energy captured of each mode and displacement across principal component direction are accurate to some extent. Without knowing the system dynamics, we are able to determine the movement of the mass to some extent and the minimum number of modes needed to represent a system.

Appendix A. MATLAB functions used and brief implementation explanation

- *I = rgb2gray(RGB)* converts the true color image RGB to the grayscale image I.
- *k = find(X)* returns a vector containing the linear indices of each nonzero element in array X.
- *[U,S,V] = svd(A)* performs a singular value decomposition of matrix A, such that A = U*S*V'.
- x = diag(A) returns a column vector of the main diagonal elements of A.[3]

[3] Cited from MathWorks.com

Appendix B. MATLAB codes

```
%% Case 1
clear all;close all;clc
load('cam1_1.mat')
% implay(vidFrames1_1)
load('cam2_1.mat')
% implay(vidFrames2_1)
load('cam3_1.mat')
% implay(vidFrames3_1)

% for j = 1:numFrames
% X = vidFrames1_1(:,:,:,j);
% imshow(X); drawnow
% end
%[x,y,rgb,frame_num]=size(vidFrames1_1);
frame_num1=size(vidFrames1_1,4);
frame_num2=size(vidFrames2_1,4);
frame_num3=size(vidFrames3_1,4);
matrix1=[];
matrix2=[];
matrix3=[];
for i=1:frame_num1
    gray_imag=double(rgb2gray(vidFrames1_1(:,:,:,i)));
    gray_imag(:,1:240)=0;
    gray_imag(:,400:680)=0;
    gray_imag(1:180,:)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix1=[matrix1;mean(x),mean(y)];
end
for i=1:frame_num2
    gray_imag=double(rgb2gray(vidFrames2_1(:,:,:,i)));
    gray_imag(:,1:240)=0;
    gray_imag(:,400:680)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix2=[matrix2;mean(x),mean(y)];
end
for i=1:frame_num3
    gray_imag=double(rgb2gray(vidFrames3_1(:,:,:,i)));
    gray_imag(:,1:240)=0;
    gray_imag(:,500:680)=0;
    gray_imag(1:200,200:500)=0;
    gray_imag(330:480, 220:500)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix3=[matrix3;mean(x),mean(y)];
end

[M,I]=min(matrix1(1:25,2));
```

```
matrix1=matrix1(I:end,:);
[M,I]=min(matrix2(1:25,2));
matrix2=matrix2(I:end,:);
[M,I]=min(matrix3(1:25,2));
matrix3=matrix3(I:end,:);
size1=size(matrix1);
size2=size(matrix2);
size3=size(matrix3);
min_size=min([size1(1) size2(1) size3(1)]);
matrix1=matrix1(1:min_size,:);
matrix2=matrix2(1:min_size,:);
matrix3=matrix3(1:min_size,:);
x1=matrix1(:,1);
y1=matrix1(:,2);
x2=matrix2(:,1);
y2=matrix2(:,2);
x3=matrix3(:,1);
y3=matrix3(:,2);
bigX=[x1 y1 x2 y2 x3 y3];
bigX=bigX';
t=1:min_size;

subplot(3,1,1)
plot(t,bigX(1,:),t,bigX(2,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
title('Camera 1')
legend('z','x-y plane')
subplot(3,1,2)
plot(t,bigX(3,:),t,bigX(4,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 2')
subplot(3,1,3)
plot(t,bigX(6,:),t,bigX(5,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 3')

meandata=zeros(6,1);
for i=1:6
   meandata(i,1)=mean(bigX(i,:));
end
bigX=bigX-meandata;
subplot(2,1,1)
[U,S,V]=svd(bigX);
sig=diag(S);
plot(sig.^2/sum(sig.^2),'bo','Linewidth',1.5)
xlabel('Mode'); ylabel('Energy captured(%)')
subplot(2,1,2)
plot(t,S*V')
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('mode1','mode2','mode3','mode4','mode5','mode6')
```

```
%% Case 2
clear all;close all;clc
load('cam1_2.mat')
implay(vidFrames1_2)
load('cam2_2.mat')
implay(vidFrames2_2)
load('cam3_2.mat')
implay(vidFrames3_2)

frame_num1=size(vidFrames1_2,4);
frame_num2=size(vidFrames2_2,4);
frame_num3=size(vidFrames3_2,4);
matrix1=[];
matrix2=[];
matrix3=[];
for i=1:frame_num1
    gray_imag=double(rgb2gray(vidFrames1_2(:,:,:,i)));
    gray_imag(1:440,1:300)=0;
    gray_imag(1:440,470:680)=0;
    gray_imag(1:180,300:450)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix1=[matrix1;mean(x),mean(y)];
end
for i=1:frame_num2
    gray_imag=double(rgb2gray(vidFrames2_2(:,:,:,i)));
    gray_imag(1:440,1:180)=0;
    gray_imag(1:440,450:680)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix2=[matrix2;mean(x),mean(y)];
end
for i=1:frame_num3
    gray_imag=double(rgb2gray(vidFrames3_2(:,:,:,i)));
    gray_imag(:,1:280)=0;
    gray_imag(:,520:680)=0;
    gray_imag(1:170,290:500)=0;
    gray_imag(350:480, 290:500)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>240);
    matrix3=[matrix3;mean(x),mean(y)];
end

[M,I]=min(matrix1(1:25,2));
matrix1=matrix1(I:end,:);
[M,I]=min(matrix2(1:25,2));
matrix2=matrix2(I:end,:);
[M,I]=min(matrix3(1:25,2));
matrix3=matrix3(I:end,:);
size1=size(matrix1);
```

```
size2=size(matrix2);
size3=size(matrix3);
min_size=min([size1(1) size2(1) size3(1)]);
matrix1=matrix1(1:min_size,:);
matrix2=matrix2(1:min_size,:);
matrix3=matrix3(1:min_size,:);
x1=matrix1(:,1);
y1=matrix1(:,2);
x2=matrix2(:,1);
y2=matrix2(:,2);
x3=matrix3(:,1);
y3=matrix3(:,2);
bigX=[x1 y1 x2 y2 x3 y3];
bigX=bigX';
t=1:min_size;

subplot(3,1,1)
plot(t,bigX(1,:),t,bigX(2,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
title('Camera 1')
legend('z','x-y plane')
subplot(3,1,2)
plot(t,bigX(3,:),t,bigX(4,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 2')
subplot(3,1,3)
plot(t,bigX(6,:),t,bigX(5,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 3')

meandata=zeros(6,1);
for i=1:6
   meandata(i,1)=mean(bigX(i,:));
end
bigX=bigX-meandata;
subplot(2,1,1)
[U,S,V]=svd(bigX);
sig=diag(S);
plot(sig.^2/sum(sig.^2),'bo','Linewidth',1.5)
xlabel('Mode'); ylabel('Energy captured(%)')
subplot(2,1,2)
plot(t,S*V')
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('mode1','mode2','mode3','mode4','mode5','mode6')

%% Case 3
clear all;close all;clc
load('cam1_3.mat')
%implay(vidFrames1_3)
load('cam2_3.mat')
```

```
%implay(vidFrames2_3)
load('cam3_3.mat')
%implay(vidFrames3_3)
frame_num1=size(vidFrames1_3,4);
frame_num2=size(vidFrames2_3,4);
frame_num3=size(vidFrames3_3,4);
matrix1=[];
matrix2=[];
matrix3=[];
for i=1:frame_num1
    gray_imag=double(rgb2gray(vidFrames1_3(:,:,:,i)));
    gray_imag(:,1:260)=0;
    gray_imag(:,400:640)=0;
    gray_imag(1:200,270:380)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix1=[matrix1;mean(x),mean(y)];
end
for i=1:frame_num2
    gray_imag=double(rgb2gray(vidFrames2_3(:,:,:,i)));
    gray_imag(:,1:200)=0;
    gray_imag(:,450:680)=0;
    gray_imag(1:160,200:450)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix2=[matrix2;mean(x),mean(y)];
end
for i=1:frame_num3
    gray_imag=double(rgb2gray(vidFrames3_3(:,:,:,i)));
    gray_imag(:,1:200)=0;
    gray_imag(:,500:680)=0;
    gray_imag(1:100,200:500)=0;
    gray_imag(350:480, 200:500)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>240);
    matrix3=[matrix3;mean(x),mean(y)];
end

[M,I]=min(matrix1(1:25,2));
matrix1=matrix1(I:end,:);
[M,I]=min(matrix2(1:25,2));
matrix2=matrix2(I:end,:);
[M,I]=min(matrix3(1:25,2));
matrix3=matrix3(I:end,:);
size1=size(matrix1);
size2=size(matrix2);
size3=size(matrix3);
min_size=min([size1(1) size2(1) size3(1)]);
matrix1=matrix1(1:min_size,:);
matrix2=matrix2(1:min_size,:);
matrix3=matrix3(1:min_size,:);
x1=matrix1(:,1);
```

```
y1=matrix1(:,2);
x2=matrix2(:,1);
y2=matrix2(:,2);
x3=matrix3(:,1);
y3=matrix3(:,2);
bigX=[x1 y1 x2 y2 x3 y3];
bigX=bigX';
t=1:min_size;

subplot(3,1,1)
plot(t,bigX(1,:),t,bigX(2,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
title('Camera 1')
legend('z','x-y plane')
subplot(3,1,2)
plot(t,bigX(3,:),t,bigX(4,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 2')
subplot(3,1,3)
plot(t,bigX(6,:),t,bigX(5,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 3')

meandata=zeros(6,1);
for i=1:6
   meandata(i,1)=mean(bigX(i,:));
end
bigX=bigX-meandata;
subplot(2,1,1)
[U,S,V]=svd(bigX);
sig=diag(S);
plot(sig.^2/sum(sig.^2),'bo','Linewidth',1.5)
xlabel('Mode'); ylabel('Energy captured(%)')
subplot(2,1,2)
plot(t,S*V')
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('mode1','mode2','mode3','mode4','mode5','mode6')

%% Case 4
clear all;close all;clc
load('cam1_4.mat')
implay(vidFrames1_4)
load('cam2_4.mat')
implay(vidFrames2_4)
load('cam3_4.mat')
implay(vidFrames3_4)

frame_num1=size(vidFrames1_4,4);
frame_num2=size(vidFrames2_4,4);
frame_num3=size(vidFrames3_4,4);
```

```
matrix1=[];
matrix2=[];
matrix3=[];
for i=1:frame_num1
    gray_imag=double(rgb2gray(vidFrames1_4(:,:,:,i)));
    gray_imag(:,1:260)=0;
    gray_imag(:,500:680)=0;
    gray_imag(1:200,270:500)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.8*Max);
    matrix1=[matrix1;mean(x),mean(y)];
end
for i=1:frame_num2
    gray_imag=double(rgb2gray(vidFrames2_4(:,:,:,i)));
    gray_imag(:,1:200)=0;
    gray_imag(1:100,200:400)=0;
    gray_imag(:,450:680)=0;
    gray_imag(400:480,200:400)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>0.9*Max);
    matrix2=[matrix2;mean(x),mean(y)];
end
for i=1:frame_num3
    gray_imag=double(rgb2gray(vidFrames3_4(:,:,:,i)));
    gray_imag(:,1:200)=0;
    gray_imag(:,600:640)=0;
    gray_imag(1:100,:)=0;
    gray_imag(350:480, :)=0;
    Max=max(gray_imag(:));
    [x,y]=find(gray_imag>230);
    matrix3=[matrix3;mean(x),mean(y)];
end

[M,I]=min(matrix1(1:25,2));
matrix1=matrix1(I:end,:);
[M,I]=min(matrix2(1:25,2));
matrix2=matrix2(I:end,:);
[M,I]=min(matrix3(1:25,2));
matrix3=matrix3(I:end,:);
size1=size(matrix1);
size2=size(matrix2);
size3=size(matrix3);
min_size=min([size1(1) size2(1) size3(1)]);
matrix1=matrix1(1:min_size,:);
matrix2=matrix2(1:min_size,:);
matrix3=matrix3(1:min_size,:);
x1=matrix1(:,1);
y1=matrix1(:,2);
x2=matrix2(:,1);
y2=matrix2(:,2);
x3=matrix3(:,1);
y3=matrix3(:,2);
```

```
bigX=[x1 y1 x2 y2 x3 y3];
bigX=bigX';
t=1:min_size;

subplot(3,1,1)
plot(t,bigX(1,:),t,bigX(2,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
title('Camera 1')
legend('z','x-y plane')
subplot(3,1,2)
plot(t,bigX(3,:),t,bigX(4,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 2')
subplot(3,1,3)
plot(t,bigX(6,:),t,bigX(5,:))
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('z','x-y plane')
title('Camera 3')

meandata=zeros(6,1);
for i=1:6
   meandata(i,1)=mean(bigX(i,:));
end
bigX=bigX-meandata;
subplot(2,1,1)
[U,S,V]=svd(bigX);
sig=diag(S);
plot(sig.^2/sum(sig.^2),'bo','Linewidth',1.5)
xlabel('Mode'); ylabel('Energy captured(%)')
subplot(2,1,2)
plot(t,S*V')
xlabel('Frame number'); ylabel('Displacement(pixel)')
legend('mode1','mode2','mode3','mode4','mode5','mode6')
```