Assignment3: PCA and a Spring-Mass System

Ming Lou

Abstract:

The main purpose of the project is to use PCA to analyze the movie files created from three different cameras. We are provided four different four cases which are ideal case, noisy case, horizontal displacement and horizontal displacement and rotation. For each case, we applied the Principal Component Analysis to compare the oscillations under different situation. The main programming tool for this project will be MATLAB.

Section I. Introduction and Overview :

We are going to do the Principal Component Analysis to four cases created by three different cameras. The first case is the ideal case, which the entire motion is in the z direction with simple harmonic motion being observed. The second case is the noisy case. For this case, the camera shakes which create noise. The third one is called Horizontal Displacement where we need to consider the motion on both x-y plane and z direction. The last one is Horizontal Displacement and Rotation. In this case the mass is released off-center.

There will be some oversampled redundancy with the three cameras; thus we are going to apply the Singular Value Decomposition to make sure where the most important principal component is and refine the system.

Section II. Theoretical Background

Singular Value Decomposition: $A = U\Sigma V$ where $U \in \mathbb{R}^{m\times m}$ and $v \in \mathbb{R}^{n\times n}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m\chi n}$ is diagonal.

The way to calculate the SVD is as following which is from class notes:

$$A^*A = (U\Sigma V^*)^*(U\Sigma V^*)$$

$$= V\Sigma U^* U\Sigma V^*$$

$$= V\Sigma^2 V^*$$

Multiplying on the right V gives

$$A^* AV = V\Sigma^2$$

The V here is exactly is eigenvector of $A^* A$

Similarly,

$$AA^* = U\Sigma^2 U^*$$

Multiplying U gives

$$AA^* U = U\Sigma^2$$

Which can be solved for U.

Singular Value Decomposition can help us to make sure the principal component. According to the value in the diagonal matrix, we can make sure how many principal components are making significant effects. In MATLAB, we can use function [U,S,V]= svd(A)to generate the output simply.

Section III. Algorithm Implementation and Development

Main steps:

1. Load the dataset to MATLAB and play the videos.
2. Deal with the ideal case first, use a for loop to generate the data for each camera
3. Collect all three dataset from three cameras to one matrix, minus the mean
4. Apply the SVD
5. Create graphs for the energy and the displacement on x-y plane and z direction
6. Conduct the same steps for the other three cases

Specifically, I loaded the data for the ideal case at first, then I use the size function to find the number of frames. Using a for loop, I converted the data to gray and double by rgb2gray and double command. With the simplified version, we are able to analyze. I use a filter to remove

the irrelevant information. Then I track the object by the light by finding the max value of the matrix. I set a threshold of 200 and use the find and ind2sub command to locate these values. To construct the collection of matrix with data combined by the three cameras, I simply cut the longer data to the shortest one among the three videos. With the collected-data matrix, I minus the mean value and divide the result by square root of n-1. With these steps done, I used the SVD to see the result of diagonal variance and the principle components projection. Finally, I plot the graphs. The four cases shares very similar algorithm, the only difference is the energy points. We need to capture those significant energy point according to the output.

Section IV. Computational Results

Figure 1 below is the output for the ideal case. According to the variance and energy grapgh, I dound we only have one principal component which is around 91% energy capaturing. All other points are very low. This is right for the ideal case since it's a single buket oscilllating

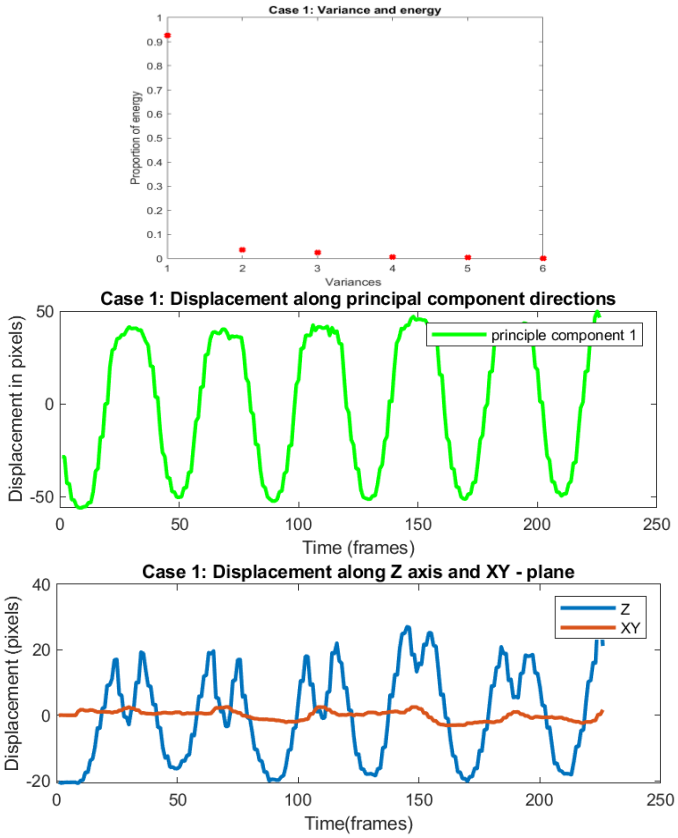

Figure1

The figure2 below is the output for the noisy case. With the cameras shaking, we have more noise. I found two points in the energy graph are large, one is around 71% and the other one is around 15%. Thus we have two principal components. Comparing to the case 1 which is the ideal case, there are more oscillation due to the noise.

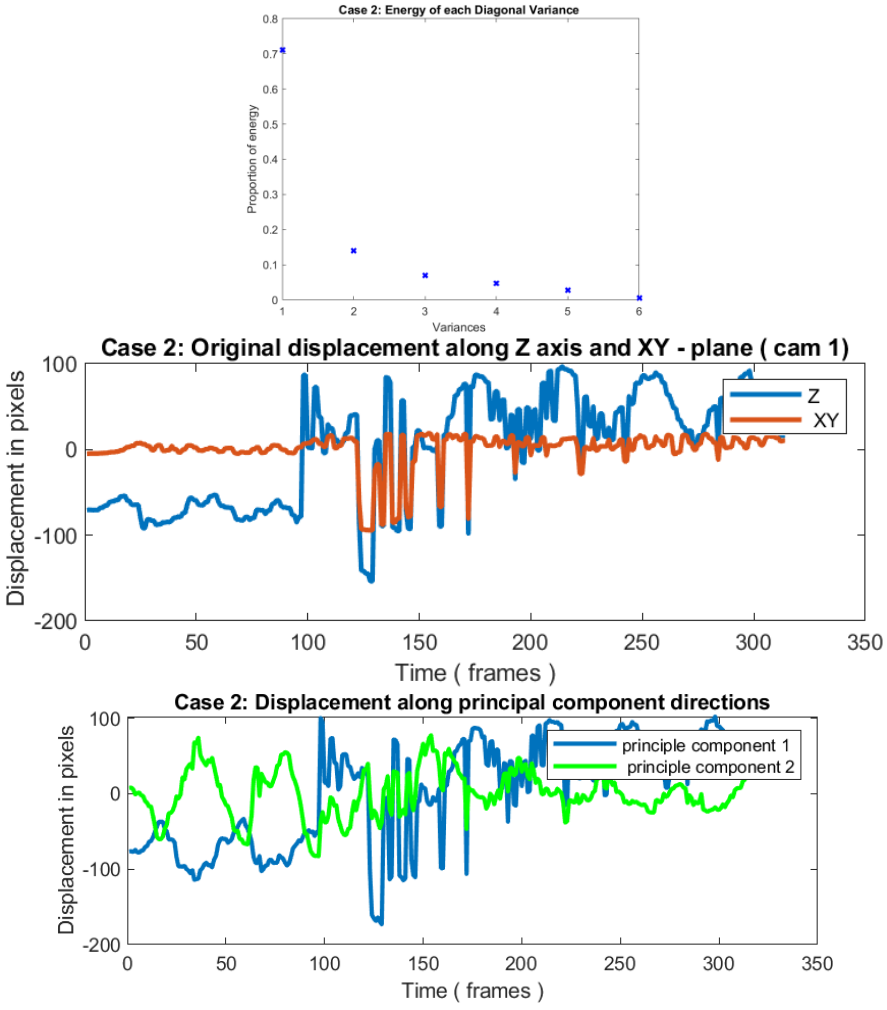

Figure2

The figure 3 below reflects the case with horizontal displacement. We saw there are three principal components which are around 49%, 39%, 14%. With the horizontal displacement, the oscillation is not change so much although we have more principal component.

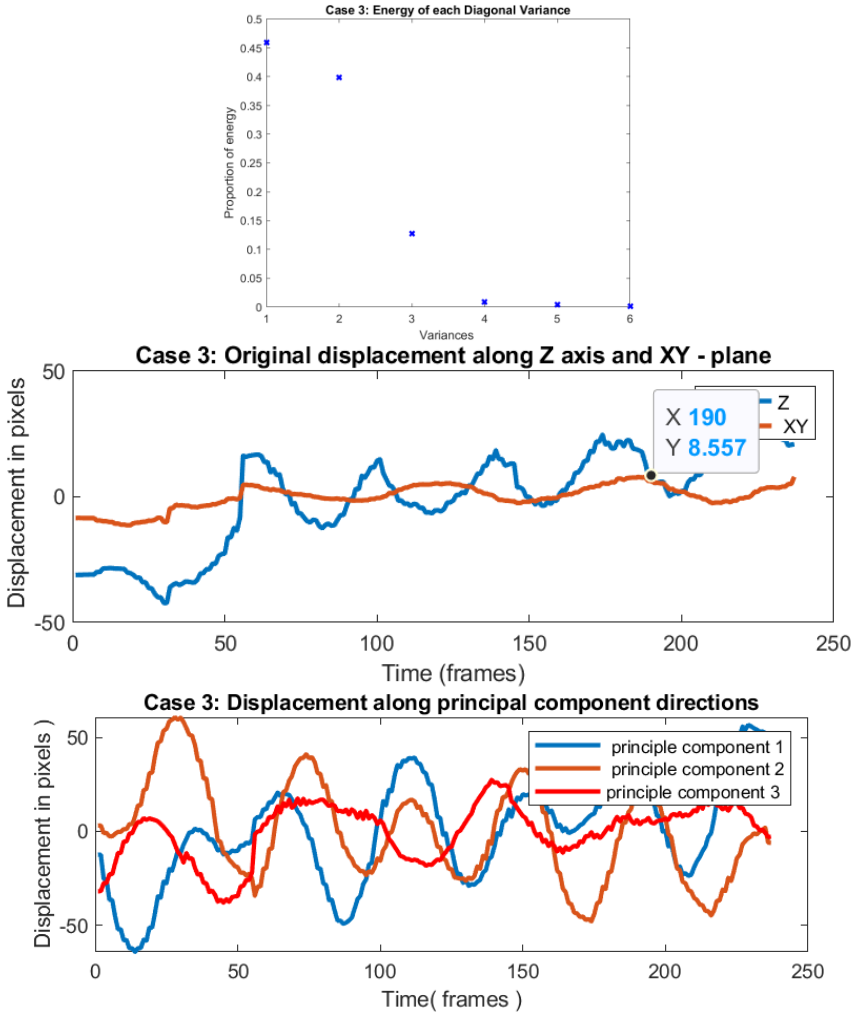

Figure3

The last case is the horizontal displacement and rotation. We found three principal component which are around 78%, 10% and 9%. Under this situation, we have both oscillatory motion and spinning motion.

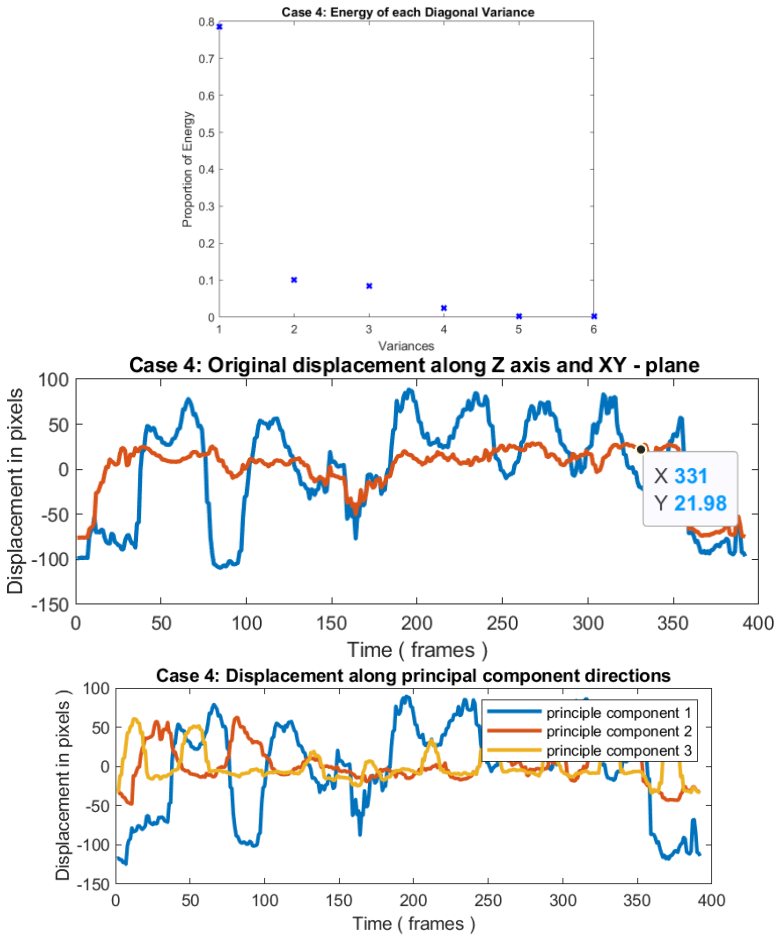

Figure 4

Section V. Summary and Conclusions

By applying the singular value decomposition and PCA, we are able to make sure how many principal components there are under different situations. These algorithms are helpful in analyzing the motion. After learning PCA, we are able to eliminate the redundancy in image and video processing.

Appendix A. MATLAB functions used and brief implementation explanation:

Rgb2gray(x): converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

Diag(x): returns a square **diagonal** matrix with the elements of vector v on the main **diagonal**.

Find(x): Find indices and values of nonzero elements

[i,j,k] = ind2sub(sz,ind): Used together with the find function to locate the bright spots.

Appendix B. MATLAB codes

```
%%
clear all ; close all ; clc ;
load ('cam1_1.mat')
load ('cam2_1.mat')
load ('cam3_1.mat')

%%
numFrames11 = size(vidFrames1_1,4);
for k = 1:numFrames11
   X11(k).cdata = vidFrames1_1(:,:,:,k);
   X11(k).colormap = [];
end
%%
filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data1 = [];
for j=1:numFrames11
   X=frame2im(X11(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
   thresh = Xf > 200;
   indices = find(thresh);
   [Y, X] = ind2sub(size(thresh),indices);
   data1 = [data1; mean(X), mean(Y)];
end

%%
numFrames21 = size(vidFrames2_1,4);
for k = 1:numFrames21
   X21(k).cdata = vidFrames2_1(:,:,:,k);
   X21(k).colormap = [];
end
```

```
filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data2 = [];
for j=1:numFrames21
   X=frame2im(X21(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
   thresh = Xf > 200;
   indices = find(thresh);
   [Y, X] = ind2sub(size(thresh),indices);
   data2 = [data2; mean(X), mean(Y)];
end
data2= data2(1:226,:);
%%
numFrames31 = size(vidFrames3_1,4);
for k = 1:numFrames31
   X31(k).cdata = vidFrames3_1(:,:,:,k);
   X31(k).colormap = [];
end

filter = zeros(480,640);
filter(200:400, 300:430) = 1;

data3 = [];
for j=1:numFrames31
   X=frame2im(X31(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
   thresh = Xf > 200;
   indices = find(thresh);
   [Y, X] = ind2sub(size(thresh),indices);
   data3 = [data3; mean(X), mean(Y)];
end
data3= data3(1:226,:);
%%
collected_data = [data1'; data2'; data3'];
[m,n]= size(collected_data);
mn = mean (collected_data ,2) ;
collected_data = collected_data - repmat(mn,1,n ) ;
[U,S,V]= svd(collected_data'/sqrt(n-1));
lambda = diag(S).^2;
Y = collected_data' * V;


%%
figure(1)
plot(1:6, lambda/sum(lambda), 'rx', 'Linewidth', 3);
title("Case 1: Variance and energy");
xlabel("Variances "); ylabel("Proportion of energy");

figure(2)
```

```
subplot(2,1,1)
plot(1:226, collected_data(2,:), 1:226, collected_data(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time(frames)");
title("Case 1: Displacement along Z axis and XY - plane");
legend("Z", "XY")
subplot(2,1,2)
plot(1:226, Y(:,1),'g','Linewidth', 2)
ylabel("Displacement in pixels)"); xlabel("Time (frames)");
title("Case 1: Displacement along principal component directions");
legend("principle component 1")

%% 2
clear all;close all;clc
load ('cam1_2.mat')
load ('cam2_2.mat')
load ('cam3_2.mat')
%%
numFrames12 = size(vidFrames1_2,4);
for k = 1:numFrames12
   X12(k).cdata = vidFrames1_2(:,:,:,k);
   X12(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data1 = [];
for j=1:numFrames12
   X=frame2im(X12(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
   thresh = Xf > 200;
   indices = find(thresh);
   [Y, X] = ind2sub(size(thresh),indices);
   data1 = [data1; mean(X), mean(Y)];
end

%%
numFrames22 = size(vidFrames2_2,4);
for k = 1:numFrames22
   X22(k).cdata = vidFrames2_2(:,:,:,k);
   X22(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data2 = [];
for j=1:numFrames22
   X=frame2im(X22(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
```

```
    thresh = Xf > 200;
    indices = find(thresh);
    [Y, X] = ind2sub(size(thresh),indices);
    data2 = [data2; mean(X), mean(Y)];
end
data2= data2(1:314,:);

%%
numFrames32 = size(vidFrames3_2,4);
for k = 1:numFrames32
    X32(k).cdata = vidFrames3_2(:,:,:,k);
    X32(k).colormap = [];
end

filter = zeros(480,640);
filter(200:400, 300:430) = 1;

data3 = [];
for j=1:numFrames32
    X=frame2im(X32(j));
    X2 = double(X);
    Xg = rgb2gray(X);
    Xg2 = double(Xg);
    Xf = Xg2.*filter;
    thresh = Xf > 200;
    indices = find(thresh);
    [Y, X] = ind2sub(size(thresh),indices);
    data3 = [data3; mean(X), mean(Y)];
end
data3= data3(1:314,:);
%%
collected_data = [data1'; data2'; data3'];
[m,n]= size(collected_data);
mn = mean (collected_data ,2) ;
collected_data = collected_data - repmat ( mn ,1 , n ) ;
[U,S,V]= svd(collected_data'/sqrt(n-1));
lambda = diag(S).^2;
Y = collected_data' * V;
%%
figure (3)
plot (1:6,lambda/sum(lambda),'bx','Linewidth', 2) ;
title("Case 2: Energy of each Diagonal Variance ") ;
xlabel("Variances ") ; ylabel("Proportion of energy") ;
figure(4)
subplot(2 ,1 ,1)
plot(1:314, collected_data(2,:), 1:314, collected_data(1,:), 'Linewidth', 2)
ylabel(" Displacement in pixels"); xlabel ("Time ( frames )") ;
legend("Z" , " XY ")
title("Case 2: Original displacement along Z axis and XY - plane ( cam 1)") ;
subplot(2,1,2)
plot(1:314 , Y (:,1) ,1:314 , Y(:,2) ,'g','Linewidth', 2)
ylabel("Displacement in pixels") ; xlabel("Time ( frames )") ;
title("Case 2: Displacement along principal component directions") ;
legend("principle component 1 " , " principle component 2")
%% 3
clear all ; close all ; clc ;
```

```
load ('cam1_3.mat')
load ('cam2_3.mat')
load ('cam3_3.mat')
%%
numFrames13 = size(vidFrames1_3,4);
for k = 1:numFrames13
   X13(k).cdata = vidFrames1_3(:,:,:,k);
   X13(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data1 = [];
for j=1:numFrames13
   X=frame2im(X13(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
   thresh = Xf > 200;
   indices = find(thresh);
   [Y, X] = ind2sub(size(thresh),indices);
   data1 = [data1; mean(X), mean(Y)];
end
data1= data1(1:237,:);
%%
numFrames23 = size(vidFrames2_3,4);
for k = 1:numFrames23
   X23(k).cdata = vidFrames2_3(:,:,:,k);
   X23(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data2 = [];
for j=1:numFrames23
   X=frame2im(X23(j));
   X2 = double(X);
   Xg = rgb2gray(X);
   Xg2 = double(Xg);
   Xf = Xg2.*filter;
   thresh = Xf > 200;
   indices = find(thresh);
   [Y, X] = ind2sub(size(thresh),indices);
   data2 = [data2; mean(X), mean(Y)];
end
data2= data2(1:237,:);

%%
numFrames33 = size(vidFrames3_3,4);
for k = 1:numFrames33
   X33(k).cdata = vidFrames3_3(:,:,:,k);
   X33(k).colormap = [];
end
```

```
filter = zeros(480,640);
filter(200:400, 300:430) = 1;

data3 = [];
for j=1:numFrames33
    X=frame2im(X33(j));
    X2 = double(X);
    Xg = rgb2gray(X);
    Xg2 = double(Xg);
    Xf = Xg2.*filter;
    thresh = Xf > 200;
    indices = find(thresh);
    [Y, X] = ind2sub(size(thresh),indices);
    data3 = [data3; mean(X), mean(Y)];
end
%%
collected_data = [data1'; data2'; data3'];
[m,n]= size(collected_data);
mn = mean (collected_data ,2) ;
collected_data = collected_data - repmat ( mn ,1 , n ) ;
[U,S,V]= svd(collected_data'/sqrt(n-1));
lambda = diag(S).^2;
Y = collected_data' * V;
%%
figure (5)
plot(1:6 , lambda / sum ( lambda ) , 'bx', 'Linewidth', 2) ;
title("Case 3: Energy of each Diagonal Variance") ;
xlabel("Variances ") ; ylabel ("Proportion of energy") ;
figure (6)
subplot (2 ,1 ,1)
plot (1:237 , collected_data(2 ,:), 1:237, collected_data(1,:),'Linewidth',2)
ylabel (" Displacement in pixels") ; xlabel("Time (frames)") ;
legend("Z" , " XY ")
title (" Case 3: Original displacement along Z axis and XY - plane ") ;
subplot (2 ,1 ,2)
plot (1:237 , Y(: ,1) , 1:237 , Y(: ,2) , 1:237 , Y(: ,3) ,'r','Linewidth', 2)
ylabel(" Displacement in pixels ) ") ; xlabel("Time( frames )") ;
title(" Case 3: Displacement along principal component directions ") ;
legend(" principle component 1 " , " principle component 2 " , "principle component 3 ")

%%
clear all ; close all ; clc ;
load ('cam1_4.mat')
load ('cam2_4.mat')
load ('cam3_4.mat')
%%
numFrames14 = size(vidFrames1_4,4);
for k = 1:numFrames14
    X14(k).cdata = vidFrames1_4(:,:,:,k);
    X14(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;
```

```
data1 = [];
for j=1:numFrames14
    X=frame2im(X14(j));
    X2 = double(X);
    Xg = rgb2gray(X);
    Xg2 = double(Xg);
    Xf = Xg2.*filter;
    thresh = Xf > 200;
    indices = find(thresh);
    [Y, X] = ind2sub(size(thresh),indices);
    data1 = [data1; mean(X), mean(Y)];
end
%%
numFrames24 = size(vidFrames2_4,4);
for k = 1:numFrames24
    X24(k).cdata = vidFrames2_4(:,:,:,k);
    X24(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;

data2 = [];
for j=1:numFrames24
    X=frame2im(X24(j));
    X2 = double(X);
    Xg = rgb2gray(X);
    Xg2 = double(Xg);
    Xf = Xg2.*filter;
    thresh = Xf > 200;
    indices = find(thresh);
    [Y, X] = ind2sub(size(thresh),indices);
    data2 = [data2; mean(X), mean(Y)];
end
data2= data2(1:392,:);
%%
numFrames34 = size(vidFrames3_4,4);
for k = 1:numFrames34
    X34(k).cdata = vidFrames3_4(:,:,:,k);
    X34(k).colormap = [];
end

filter = zeros(480,640);
filter(90:400, 230:360) = 1;


data3 = [];
for j=1:numFrames34
    X=frame2im(X34(j));
    X2 = double(X);
    Xg = rgb2gray(X);
    Xg2 = double(Xg);
    Xf = Xg2.*filter;
    thresh = Xf > 200;
    indices = find(thresh);
    [Y, X] = ind2sub(size(thresh),indices);
```

```
    data3 = [data3; mean(X), mean(Y)];
end
data3= data3(1:392,:);

%%
collected_data = [data1'; data2'; data3'];
[m,n]= size(collected_data);
mn = mean (collected_data ,2) ;
collected_data = collected_data - repmat ( mn ,1 , n ) ;
[U,S,V]= svd(collected_data'/sqrt(n-1));
lambda = diag(S).^2;
Y = collected_data' * V;
%%
figure(7)
plot(1:6 , lambda / sum(lambda) , 'bx', 'Linewidth', 2) ;
title(" Case 4: Energy of each Diagonal Variance") ;
xlabel("Variances") ; ylabel ("Proportion of Energy") ;
figure(8)
subplot(2 ,1 ,1)
plot(1:392 , collected_data(2 ,:) , 1:392, collected_data(1,:),'Linewidth',2)
ylabel("Displacement in pixels") ; xlabel("Time ( frames )") ;
title("Case 4: Original displacement along Z axis and XY - plane ") ;
subplot(2 ,1 ,2)
plot(1:392 , Y(: ,1) , 1:392 , Y(: ,2) , 1:392 , Y(: ,3) , 'Linewidth', 2)
ylabel("Displacement in pixels )") ; xlabel("Time ( frames )") ;
title("Case 4: Displacement along principal component directions") ;
legend("principle component 1" , "principle component 2" , "principle component 3")
```