AMATH 482 HW3: PCA

Tiffany Tang
Department of Applied Mathematics
University of Washington
leqitang@uw.edu

Abstract

In this paper, we look at four different sets of oscillation scenario. For each set, we have three videos recorded a same oscillation movement with three different shooting angles. With the oscillating data collected from three cameras, we performed principal component analysis on the four systems to convert the three sets of X and Y coordinates during the shooting time period into six principal components. The energy of each of these components are also calculated and plotted to examine the effects of the number of mode(rank). We then projected the data onto significant principal components basis sets and compared them to the original oscillation behaviors.

1 Introduction and Overview

Movie files with three shooting angles are taken within four different circumstances. One is the ideal case which we record a mass doing a simple hormonic motion with spring; the second is the case which we record the mass motion but camera shakes has been introduced; the third is the case that the mass is released off-center; the last circumstance was presented with the mass released off-center and rotated. We converted the data collected by three cameras at different tracking locations and focused on the time-space data on tracking the movement of a light on the moving mass. After acquiring the frame data matrix from three cameras, we would perform SVD and compare the four cases.

2 Theoretical Background

The singular valued decomposition is a method of decomposing a matrix into three other matrices:

$$A = USV^T$$

Where we have A as a m*n matrix. V as a n*n unitary matrix with v_1 and v_2 as its columns, U is m*m unitary matrix with u_1 and u_2 as its columns, and Σ is a m*n diagonal matrix with σ_j on its diagonals. The values σ_n on the diagonal matrix are the singular values of the matrix A. The singular values are nonnegative and ordered from largest to smallest.

We can compute the SVD by having

$$A^TA = (U\Sigma V^T) * (U\Sigma V^T) = V\Sigma V * U\Sigma V^T = V\Sigma^2 V^T$$

Multiplying on the right by V gives

$$A^TAV = V\Sigma^2$$

Similarly,

$$A^TA = U\Sigma^2 U^T$$

$A^TAU = U\Sigma^2$

By computing the normalized eigenvectors for above two equations, we have orthonormal basis vectors for U and V.

Principal component analysis (PCA) is a method that apply the SVD. The PCA break down a multidimensional data into a set of orthogonal 'modes' and corresponding scalar values. Covariance matrix is used to help find diagonal matrix of singular values. The covariance matrix is:$C_X = \frac{1}{n-1} XX^T = AA^T$

Then from the fomular of the SVD, we have

$$C_X = AA^T = U\Sigma^2 U^T$$

We define the transformed variable Y= U*X where U is the unitary transformation related to the SVD ($X = USV^T$). We then calculate the variance in Y:

$$C_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} (U * X)(U * X)^T$$

$$= \frac{1}{n-1} U * (XX^T) * U = \frac{1}{n-1} U^T U \Sigma^2 U * U^T$$

$$= \frac{1}{n-1} \Sigma^2$$

Since the off-diagonal elements of Σ are zero, it follows that the variables in Y are uncorrelated.

Proper Orthogonal Decomposition (POD) is the same as PCA, but is usually used in fluid dynamics. With the low-rand approximation from the SVD, we get the POD modes which determine how much energy from the full system is contained in each mode.

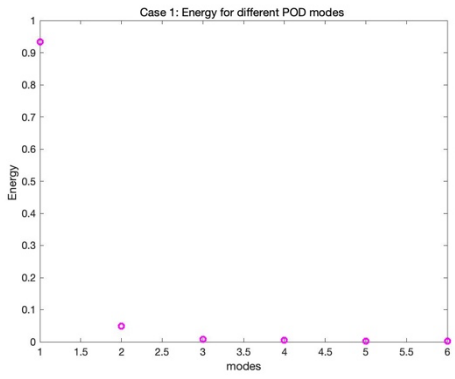
3 Algorithm Implementation and Development

In order to process the data from all four cases, we first filtered out irrelevant signal data in all the movies we captured. By doing this, we found the number of frames for each movie (size command). Then we converted every frame into grayscale (rgb2gray command). Since we need to keep track of a bright light that did the same motion as the mass, we made a filter that filtered out all the irrelevant signal data by converting these data to zeros which means black at grayscale. After applying the filter, we created a threshold matrix in order to find the signals which have the value greater than around 250. Then we found the indices of these points (find command) and acquired the coordinates of all the bright points(ind2sub command). Lastly, we averaged the coordinates (mean command) and added this to a data array which gave a matrix of size 2*the number of frames.

However, since the total number of frames is different from the three cameras recoding the same movement, we normalized them by changing the first frame in each of the videos as the lowest Y coordinates in first twenty frames. In this way, we set the movement of mass in three movies as synchronous. We then added three data matrixes as one large matrix which we has the size of 6* the lowest number of frames. The data matrix was centered by subtracted the mean of each row. Then we took the SVD decomposition of the data matrix (svd command) to get three matrixes: u, s, v. From diagonal matrix s, we got the singular values (diag command) and squared then to get variances for each principal component. Then we plooted the variance for the rand 1 through 6 to visualize the value of energy captured. Finally, the principal component projections were plotted.

The same process has been placed for all four cases of data.

4 Computational Results



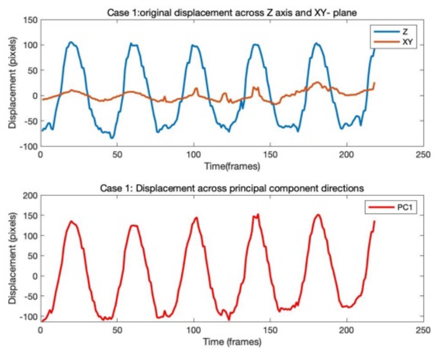
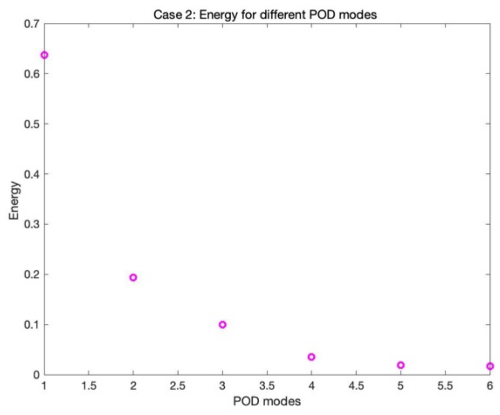


Figure 1(Left): The energy captured for the rand 1 through 6 approximations in Case 1

Figure 2(Right): The movement of z direction and the principal component in Case 1



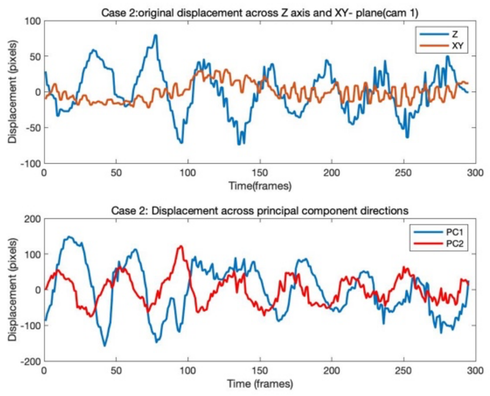
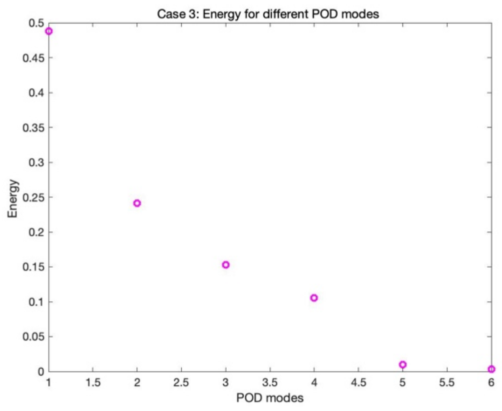


Figure 3(Left): The energy captured for the rand 1 through 6 approximations in Case 2

Figure 4(Right): The movement of z displacement and the principal component with new basis in Case 2



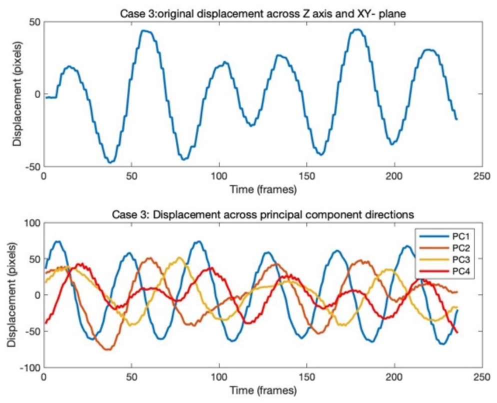
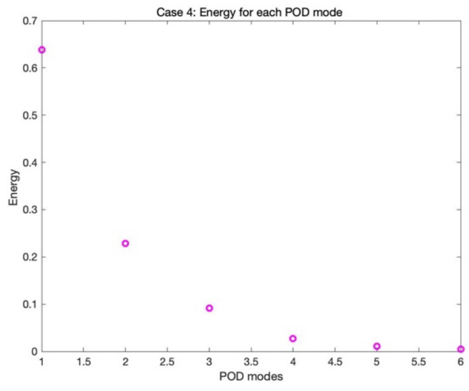


Figure 5(Left): The energy captured for the rand 1 through 6 approximations in Case 3

Figure 6(Right): The movement of z displacement and the principal component with new basis in Case 3

For test 1: ideal case, the value of the energy are 0.9340, 0.0483, 0.0088, 0.005, 0.0021, and 0.0016 for the mode 1 through 6 approximations, respectively. The rank-1 approximation contains most of the energy. The rest of them captured relatively low energy. From figure 2, we have the mass with simple hormonic motion only in Z-axis and small fluctuation on XY plane, so we set one principle component. The projection of this principle component is similar to the actual movement in Z-axis.



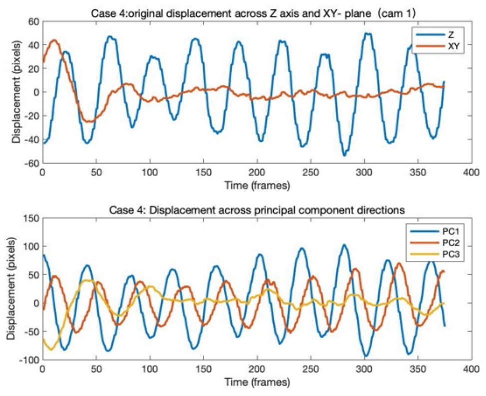


Figure 7(Left): The energy captured for the rand 1 through 6 approximations in Case 4

Figure 8(Right): The movement of z displacement and the principal component with new basis in Case 4

For test 2, we have the mass move with camera shakes. The value of the energy are 0.6368, 0.1937, 0.0998, 0.0348, 0.0185, and 0.0165 for the mode 1 through 6 approximations, respectively. Comparing to the energy capture in test 1, the energy captured in test 2 for rank 1 approximation is significantly lower. This could be the result of introduced noisy. From figure 4, two principal components are extracted and show a time different of the oscillation. Due to the noisy, the second principal component is fluctuated but we can still see the oscillated motion. There is no significant oscillation in X-Y plane showed since there was only noise presented. However, the vertical motion and the first principal components still appears periodic.

For test 3, the mass the released off centered. From figure 6, the oscillated movement is clearly presented in horizonal plane, which are not showed in case 1 and case 2. The four principal components all demonstrate a simple hormonic oscillation. The value of energy are 0.4879, 0.2413, 0.1528, 0.1051, 0.0098, and 0.0031 for the mode 1 through 6 approximations, respectively.

For test 4, the mass is released off-center and rotates. The value of energy are 0.6385, 0.2286, 0.0910, 0.0272, 0.0107, 0.0041. From figure 8, the movement in x-position and y-position are not tracked and present obviously from camera 1. Also, different filter has been applied to focus on the movement of the mass, the issue was still not fixed. However, after applying the SVD, the oscillation in Z direction and rotation in X-Y plane were presented in the three orthonormal principal components.

5 Summary and Conclusions

By performing PCA on the dataset of movies captured oscillation, we tracked the oscillation behavior and even rotation in different axis. The decomposition of the datasets allows the orthonormal principal components to clearly present the movements. We also looked at the energy captured by different number mode approximation. The further research about PCA could be focusing on how to acquiring a higherof singular value for low-rand approximation and applying the PCA of other different types of data.

Appendix A. MATLAB Functions

1. [row, col] = ind2sub(sz, ind) returns the array row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz.
2. I = rgb2gray(RGB) returns the grayscale image I of the truecolor image RGB.
3. K = find(X) returns a vector containing the linear indices of each nonzero element I array X.
4. [U, S, V] = svd(A) performs a singular value decomposition of matrix A.
5. B = repmat(A, r1, …rN) specifies a list of scalars, r1, …, rN, that describes how copies of A are arranged in each dimension.
6. D = diag(v) returns a square diagonal matrix with the element of vector v on the main diagonal.
7. X = zeros(sz1, …, szN) returns as sz1-by-…-by-szN array of zeros where sz1, …, szn indicate the size of each dimension.

Appendix B. Matlab Code

```
%% test 1
clc;clear all;close all;
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
%implay(vidFrames1_1)
numFrames11 = size(vidFrames1_1,4);
numFrames21 = size(vidFrames2_1,4);
numFrames31 = size(vidFrames3_1,4);
%%

width = 50;
filter = zeros(480, 640);

filter(300-2.6*width:1:300+2.6*width, 350-width:1:350+width)=1;


data1 = [];
for j = 1: numFrames11
    X = vidFrames1_1(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 250;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data1 = [data1; mean(X), mean(Y)];
    %imshow(uint8(Xf)); drawnow
end

filter = zeros(480, 640);

filter(100-width:1:350+width, 290-1.3*width:1:290+1.3*width)=1;


data2 = [];
for j = 1: numFrames21
    X = vidFrames2_1(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 250;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data2 = [data2; mean(X), mean(Y)];

    %imshow(uint8(Xf)); drawnow
end

filter = zeros(480, 640);

filter(250-1*width:1:250+2*width, 360-2.6*width:1:360+2.6*width)=1;


data3 = [];
for j = 1: numFrames31
    X = vidFrames3_1(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 248;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data3 = [data3; mean(X), mean(Y)];
    %imshow(uint8(Xf)); drawnow
end

[M, I] = min(data1(1:20, 2));
data1 = data1(I:end, :);
[M, I] = min(data2(1:20, 2));
data2 = data2(I:end, :);
[M, I] = min(data3(1:20, 2));
data3 = data3(I:end, :);

data2 = data2(1:length(data1), :);
data3 = data3(1:length(data1), :);
```

```
datasum = [data1';data2';data3']

[m,n] = size(datasum);
avg = mean(datasum, 2);
datasum = datasum - repmat(avg, 1, n);

[u,s,v] = svd(datasum'/sqrt(n-1))
lambda = diag(s).^2;
Y = datasum'* v; % principal components projection

sig = diag(s);

figure()
plot(1:6, lambda/sum(lambda), 'mo', 'Linewidth', 2)
title("Case 1: Energy for different POD modes ")
xlabel("modes"); ylabel("Energy ")

figure()
subplot(2, 1, 1)
plot(1:218, datasum(4,:), 1:218, datasum(6,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time(frames)")
title("Case 1:original displacement across Z axis and XY- plane")
legend('Z', 'XY')
subplot(2, 1, 2)
plot(1:218, Y(:, 1), 'r', 'Linewidth', 2)
ylabel('Displacement (pixels)'); xlabel('Time (frames)')
title ('Case 1: Displacement across principal component directions')
legend('PC1')

%% tEST 2
clc;clear all;close all;
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
%implay(vidFrames1_1)
numFrames12 = size(vidFrames1_2,4);
numFrames22 = size(vidFrames2_2,4);
numFrames32 = size(vidFrames3_2,4);

width = 50;
filter = zeros(480, 640);

filter(300-2.6*width:1:300+2.6*width, 350-width:1:350+2*width)=1;
```

```
data1 = [];
for j = 1: numFrames12
    X = vidFrames1_2(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 250;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data1 = [data1; mean(X), mean(Y)];

end


width = 50;
filter = zeros(480, 640);

filter(100-width:1:375+width, 215-width:1:290+2.7*width)=1;


data2 = [];
for j = 1: numFrames22
    X = vidFrames2_2(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 249;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data2 = [data2; mean(X), mean(Y)];

end

width = 50;
filter = zeros(480, 640);

filter(250-1*width:1:250+2.7*width, 360-2.5*width:1:360+2.7*width)=1;
```

```
data3 = [];
for j = 1: numFrames32
    X = vidFrames3_2(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 245;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data3 = [data3; mean(X), mean(Y)];

end

[M, I] = min(data1(1:20, 2));
data1 = data1(I:end, :);
[M, I] = min(data2(1:20, 2));
data2 = data2(I:end, :);
[M, I] = min(data3(1:20, 2));
data3 = data3(I:end, :);

data2 = data2(1:length(data1), :);
data3 = data3(1:length(data1), :);

datasum = [data1';data2';data3']

[m,n] = size(datasum);
avg = mean(datasum, 2);
datasum = datasum - repmat(avg, 1, n);

[u,s,v] = svd(datasum'/sqrt(n-1))
lambda = diag(s).^2;
Y = datasum'* v; % principal components projection

sig = diag(s);

figure()
plot(1:6, lambda/sum(lambda), 'mo', 'Linewidth', 2)
title("Case 2: Energy for different POD modes ")
xlabel("POD modes"); ylabel("Energy ")
```

```
figure()
subplot(2, 1, 1)
plot(1:295, datasum(2,:), 1:295, datasum(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time(frames)")
title("Case 2:original displacement across Z axis and XY- plane(cam
1)")
legend('Z', 'XY')
subplot(2, 1, 2)
plot(1:295, Y(:, 1), 1:295, Y(:,2),'r', 'Linewidth', 2)
ylabel('Displacement (pixels)'); xlabel('Time (frames)')
title ('Case 2: Displacement across principal component directions')
legend('PC1', 'PC2')

%% tEST 3
clc;clear all;close all;
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
%implay(vidFrames1_1)
numFrames13 = size(vidFrames1_3,4);
numFrames23 = size(vidFrames2_3,4);
numFrames33 = size(vidFrames3_3,4);

width = 50;
filter = zeros(480, 640);

filter(275-width:1:300+3*width, 350-1.5*width:1:350+2*width)=1;


data1 = [];
for j = 1: numFrames13
    X = vidFrames1_3(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 250;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data1 = [data1; mean(X), mean(Y)];
```

```
end


width = 50;
filter = zeros(480, 640);

filter(100-width:1:375+width, 215-width:1:290+2.7*width)=1;


data2 = [];
for j = 1: numFrames23
    X = vidFrames2_3(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 249;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data2 = [data2; mean(X), mean(Y)];

end

width = 50;
filter = zeros(480, 640);

filter(250-1.8*width:1:250+2.3*width, 360-
2.5*width:1:360+2.7*width)=1;


data3 = [];
for j = 1: numFrames33
    X = vidFrames3_3(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 245;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);
    data3 = [data3; mean(X), mean(Y)];
end

[M, I] = min(data1(1:20, 2));
data1 = data1(I:end, :);
[M, I] = min(data2(1:20, 2));
data2 = data2(I:end, :);
[M, I] = min(data3(1:20, 2));
data3 = data3(I:end, :);

data2 = data2(1:length(data3), :);
data1 = data1(1:length(data3), :);

datasum = [data1';data2';data3']

[m,n] = size(datasum);
avg = mean(datasum, 2);
datasum = datasum - repmat(avg, 1, n);

[u,s,v] = svd(datasum'/sqrt(n-1))
lambda = diag(s).^2;
Y = datasum'* v; % principal components projection

figure()
plot(1:6, lambda/sum(lambda), 'mo', 'Linewidth', 2)
title("Case 3: Energy for different POD modes ")
xlabel("POD modes"); ylabel("Energy ")

figure()
subplot(2, 1, 1)
plot(1:236, datasum(1, :), 1:236, datasum(2,:),'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)")
title("Case 3:original displacement across Z axis and XY- planeï¼ˆcam
1ï¼‰")
legend('Z', 'XY')
subplot(2, 1, 2)
plot(1:236, Y(:, 1), 1:236, Y(:,2),1:236, Y(:, 3),1:236, Y(:, 4),
'r', 'Linewidth', 2)
ylabel('Displacement (pixels)'); xlabel('Time (frames)')
title ('Case 3: Displacement across principal component directions')
legend('PC1', 'PC2','PC3', 'PC4')

%% test 4
clc;clear all;close all;
```

```
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
%implay(vidFrames1_1)
numFrames14 = size(vidFrames1_4,4);
numFrames24 = size(vidFrames2_4,4);
numFrames34 = size(vidFrames3_4,4);

width = 50;
filter = zeros(480, 640);

filter(275-width:1:400+width, 350-1.5*width:1:370+2*width)=1;


data1 = [];
for j = 1: numFrames14
    X = vidFrames1_4(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 247;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data1 = [data1; mean(X), mean(Y)];

end


width = 50;
filter = zeros(480, 640);

filter(100-width:1:350+width, 215-width:1:290+2.7*width)=1;


data2 = [];
for j = 1: numFrames24
    X = vidFrames2_4(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 249;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data2 = [data2; mean(X), mean(Y)];

end

width = 50;
filter = zeros(480, 640);

filter(150-width:1:250+1*width, 360-1.8*width:1:360+2.9*width)=1;


data3 = [];
for j = 1: numFrames34
    X = vidFrames3_4(:,:,:,j);
    Xabw = rgb2gray(X);
    X2 = double(X);

    Xabw2 = double(Xabw);
    Xf = Xabw2.*filter;
    thresh = Xf > 234;
    indeces = find(thresh);
    [Y, X] = ind2sub(size(thresh), indeces);

    data3 = [data3; mean(X), mean(Y)];
end

[M, I] = min(data1(1:20, 2));
data1 = data1(I:end, :);
[M, I] = min(data2(1:20, 2));
data2 = data2(I:end, :);
[M, I] = min(data3(1:20, 2));
data3 = data3(I:end, :);

data2 = data2(1:length(data3), :);
data1 = data1(1:length(data3), :);

datasum = [data1';data2';data3']

[m,n] = size(datasum);
avg = mean(datasum, 2);
```

```
datasum = datasum - repmat(avg, 1, n);

[u,s,v] = svd(datasum'/sqrt(n-1))
lambda = diag(s).^2;
Y = datasum'* v; % principal components projection
sig = diag(s);
figure()
plot(1:6, lambda/sum(lambda), 'mo', 'Linewidth', 2)
title("Case 4: Energy for each POD mode ")
xlabel("POD modes"); ylabel("Energy ")

figure()
subplot(2, 1, 1)
plot(1:375, datasum(2,:),1:375, datasum(1,:),'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)")
title("Case 4:original displacement across Z axis and XY- planeï¼ˆcam
1ï¼‰")
legend('Z', 'XY')
subplot(2, 1, 2)
plot(1:375, Y(:, 1), 1:375, Y(:,2),1:375, Y(:, 3), 'Linewidth', 2)
ylabel('Displacement (pixels)'); xlabel('Time (frames)')
title ('Case 4: Displacement across principal component directions')
legend('PC1', 'PC2','PC3')
```