AMATH482: Principle Components Analysis

Shuying Zhang
email: shuying3@uw.com

Abstract

This project explores the Principle Components Analysis (PCA) through mass and spring experiments under four different cases. After extracting out the true trajectory, we use Singular Value Decomposition (SVD) to calculate the principle components projection.

1 Introduction and Overview

Singular Value Decomposition (SVD) is an powerful techinique in linear algebra that has a broad application including Principle Components Analysis (PCA). This report explores the application of PCA through mass and spring system.

There are four sets of experiment data. Each contains three data files and represents a test case: ideal, noisy, horizontal displacement, and horizontal displacement with rotation.

There are five sections in this report. The introduction and overview section gives a brief description of the experiments and the report. The theoretical part provides background knowledge of SVD and PCA. In algorithm implementation and development, we will process experiment data and perform PCA. The computational results show the trajectories we extract through experiment data and the final principle components projection.

2 Theoretical Background

We will introduce the basic idea behind SVD and PCA here. Some descriptions and figures are from the class notes [1].

2.1 Singular Value Decomposition (SVD)

SVD is a factorization of matrix into three components and use them in many applications. The full SVD takes the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{1}$$

where

$$\begin{aligned} &\mathbf{U} \in C^{m\times m} \text{ is unitary} \\ &\mathbf{V} \in C^{n\times n} \text{ is unitary} \\ &\mathbf{\Sigma} \in R^{m\times n} \text{ is diagonal} \end{aligned} \tag{2}$$

The diagonal entries of Σ has the property that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ where $p = \min(m, n)$. We can also do diagonalization via SVD.

In general, one can apply SVD on every matrix. To compute SVD, in MATLAB, use command `[U,S,V] = svd(A)`.

2.2 Principle Components Analysis (PCA)

PCA is an application of SVD. PCA allows us to remove the redundancy and lower the dimensions of the complex and random data by identifying the signals with maximal variance. See below.

Before applying PCA, we need to stack rows of data into a single matrix. For example, in this project, we get data from camera a, camera b, camera c. Suppose the data has x and y values. Then to stack them, we need

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{y}_a \\ \mathbf{x}_b \\ \mathbf{y}_b \\ \mathbf{x}_c \\ \mathbf{y}_c \end{bmatrix} \tag{3}$$

Now let us take a look at how SVD helps us construct covariance matrix $\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T$. Since SVD can diagnolize any matrix, we can define a transformed variable:

$$\mathbf{Y} = \mathbf{U}^*\mathbf{X} \tag{4}$$

.

Then we can derive the covariance in Y:

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T \tag{5}$$

$$= \frac{1}{n-1}\left(\mathbf{U}^*\mathbf{X}\right)\left(\mathbf{U}^*\mathbf{X}\right)^T \tag{6}$$

$$= \frac{1}{n-1}\mathbf{U}^*\left(\mathbf{X}\mathbf{X}^T\right)\mathbf{U} \tag{7}$$

$$= \frac{1}{n-1}\mathbf{U}^*\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}\mathbf{U}^* \tag{8}$$

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{\Sigma}^2 \tag{9}$$

Combine with `svd command`, it's easy to calculate the variance and identify any principle components. Notice, before calling `[u,s,v]=svd(X/sqrt(n-1))` we need to manually substract the row mean from the data with the following lines: `[m,n]=size(X); mn=mean(X,2); X=X-repmat(mn,1,n);`

Plot the diagonal of Σ^2, we can view the main directions of the data and determine the principle components.

2.3 General Procedure of PCA

1. Organize the data into a single matrix, where rows are number of measurement types (cam 1, cam 2...) and columns are the number of measurements with each equipment.

2. Substract the row mean from each row.

3. Compute SVD and use the covariance matrix generated to determine the principle components.

3 Algorithm Implementation and Development

3.1 Starting the problem

Preparation Each set of data contains three data files. They are actually video frames recorded by three cameras in three different locations. The content of the videos are the same: mass and spring system doing simple harmonic motion. On can take a look at the video clips through MATLAB commands. In the first set of test cases, the data is free from all the noise. We call it the ideal case. In the second set of test cases, the data is very noisy because of shaking. In the third set of test cases, the data has horizontal displacement and the last set of test cases not only has horizontal displacement but also rotation. In the videos, one can also see a flash light on top of the mass. We will track the light to obtain the true trajectory.

The way of processing the data, extracting the trajectory and performing PCA algorithms are highly similar in four tests, and the code only contains minor changes. Hence, we will show the common steps here, instead of elaborating on every test cases.

3.2 Extracting the trajectory

Find the location of the flashlight In every frame, we could notice the existence of the flashlight, which represents the displacement of the mass. Each frame are actually an image consisting of many pixels. Therefore, if we turn the RGB data into gray scale, all the value will be between 0 to 255, where 0 and 255 stand for black and white respectively. The flashlight must be the brightest pixel among all the pixels. As long as we find the location

of the brightest pixel in each frame, combining them all together, we get the oscillating trajectories.

To find the location of the brightest pixel, simply returns the subscripts of the pixel with maximum value.

One thing to notice here is that the data files not necessarily share the same number of video frames. To simplify the process, we take the minimum video frames among the data files in four tests.

We will end up getting six rows of data in each test. They are x and y coordinates of flashlights for those three cameras.

Smooth the trajectory Since we already know that the trajectory is simple harmonic, we can shift the phase of the coordinates we get and cut them to make uniform number of frames. We end up showing 201 frames.

The trajectories we extract may contain noise. In order to make it look nicer, we apply the Gaussian filter on it.

Stack data and perform SVD In order to perform SVD, we need to stack all the rows of data together like introduced in the theoretical background. We perform the SVD and get the PCA projections.

Plot them to see what PCA tells us.

4 Computational Results

4.1 Test 1

Test 1 is the ideal spring and mass system doing simple harmonic motion. The extracted path is in Figure 1. Simple Harmonic oscillations are clearly observed. The displacement happens in one direction only. Since camera3 is horizontal, the x and y values are flipped.

Then we apply PCA, we get the singular values, square them. There are 6 singular values meaning six directions. We can see the first direction dominates the squared diagonal values with approximately 91% in Figure 2. We can conclude that the x direction in camera 1 is a principle components. This is expected since case 1 only involves displacement in one direction. Draw the principle components projection after smoothing with Gaussian filter, we get Figure 3. Other directions are there for comparison purpose only. They are almost flat.

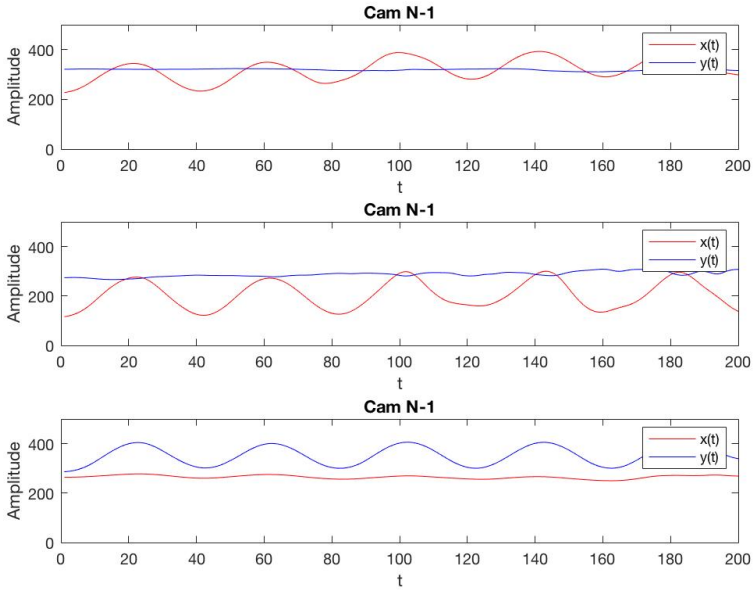

Figure 1: Displacement in x and y direction in three cameras

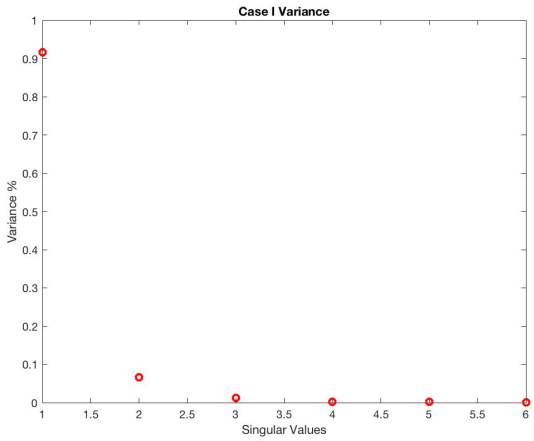

Figure 2: Squared Singular Values in case 1

4.2 Test 2

Test 2 is the noisy spring and mass system doing simple harmonic motion. The extracted path is in Figure 4. To be honest, the data is highly polluted in this case. It's difficult to tell any motion from the extracted path. Since camera3 is horizontal, the x and y values are flipped.

Then we apply PCA, we get the singular values, square them. There are 6 singular values meaning six directions. We can see the first direction dominates the squared diagonal values with approximately 55% and the second direction with 20 % in Figure 7. Other directions

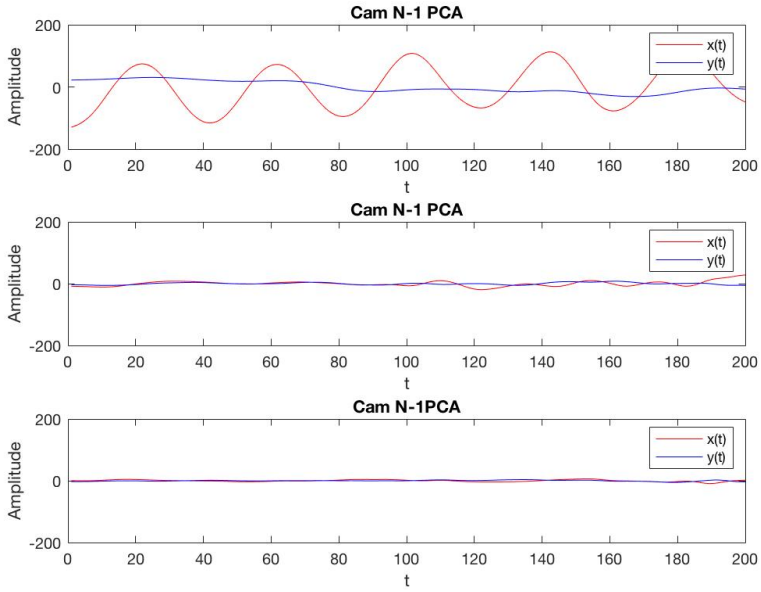

Figure 3: Principle Components Projection

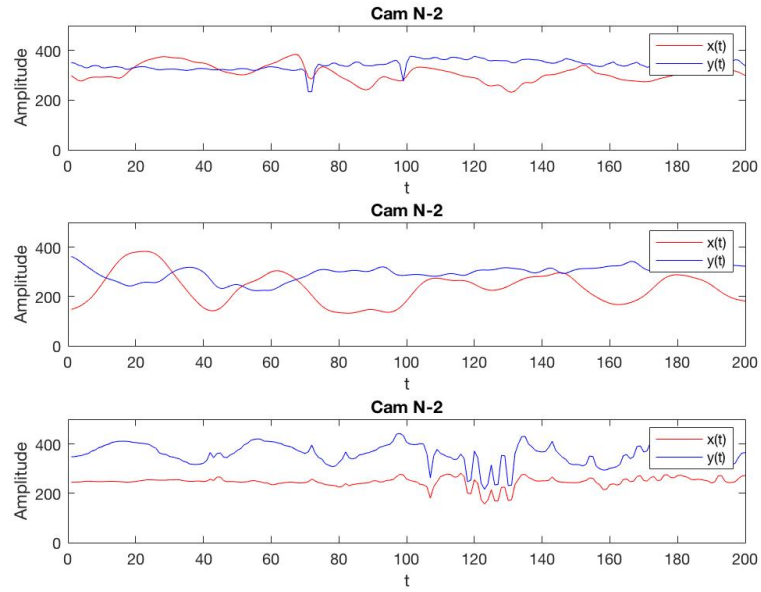

Figure 4: Displacement in x and y direction in three cameras

are not 0 but relatively small. We can conclude that the x and y direction in camera 1 is a principle components. This is expected since case 2 involves displacement in one direction. Shaking may change the y displacement. Draw the principle components projection after smoothing with Gaussian filter, we get Figure 8. Other directions are there for comparison purpose only. We can observe a simple harmonic pattern in the top plot along x directions,

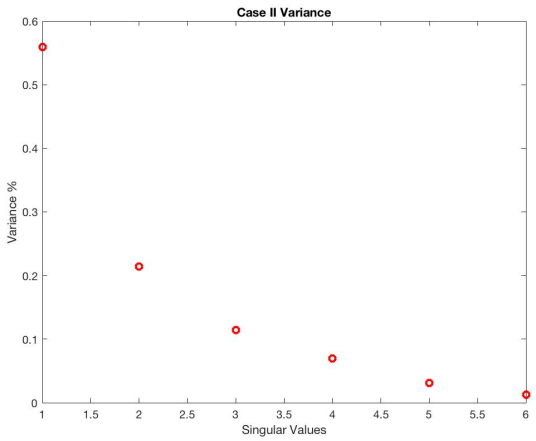

Figure 5: Squared Singular Values in case 2

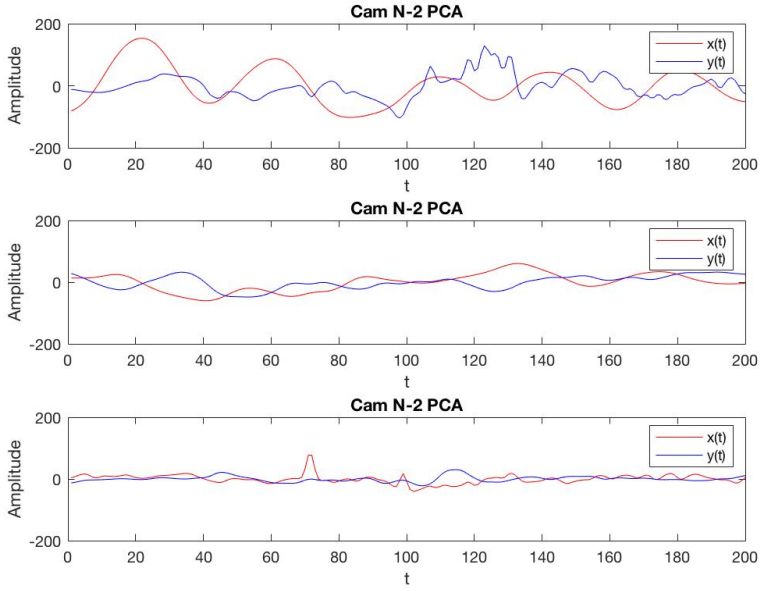

Figure 6: Principle Components Projection

y direction shows some major turbulence. other directions have some changes but can be ignored.

4.3 Test 3

Test 3 is the horizontal spring and mass system doing simple harmonic motion and pendulum motion. The extracted path is in Figure 9. The data from the first camera is not stable in the first few seconds. Later, the data shows a simple harmonic pattern. Since camera3 is horizontal, the x and y values are flipped. Then we apply PCA, we get the singular

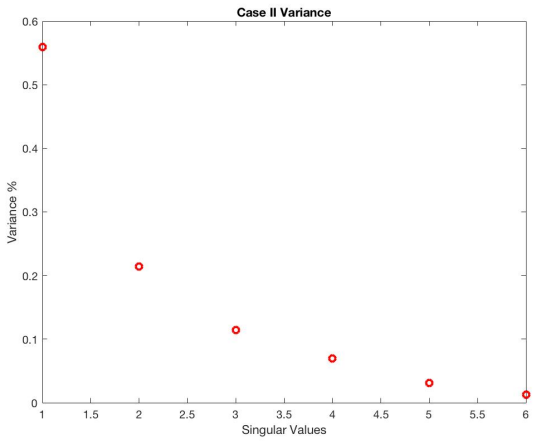

Figure 7: Squared Singular Values in case 2

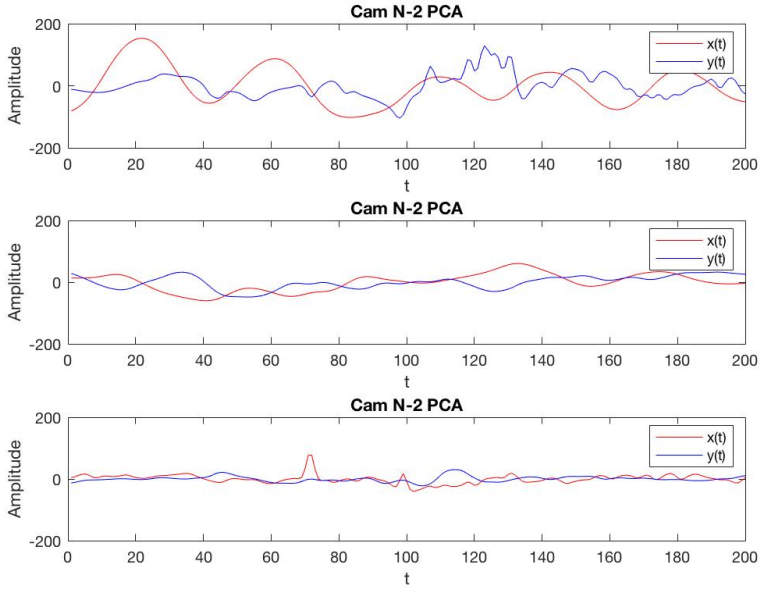

Figure 8: Principle Components Projection

values, square them. There are 6 singular values meaning six directions. We can see the first direction dominates the squared diagonal values with approximately 50%, the second direction with 30 %, the third direction with 10%. in Figure 10. Other directions are not 0 but relatively small. We can conclude that the x and y direction in camera 1 is a principle components. This is expected since case 3 involves displacement in z direction, pendulum motion may change the x and y displacement. Draw the principle components projection after smoothing with Gaussian filter, we get Figure 11. Other directions are there for comparison purpose only.

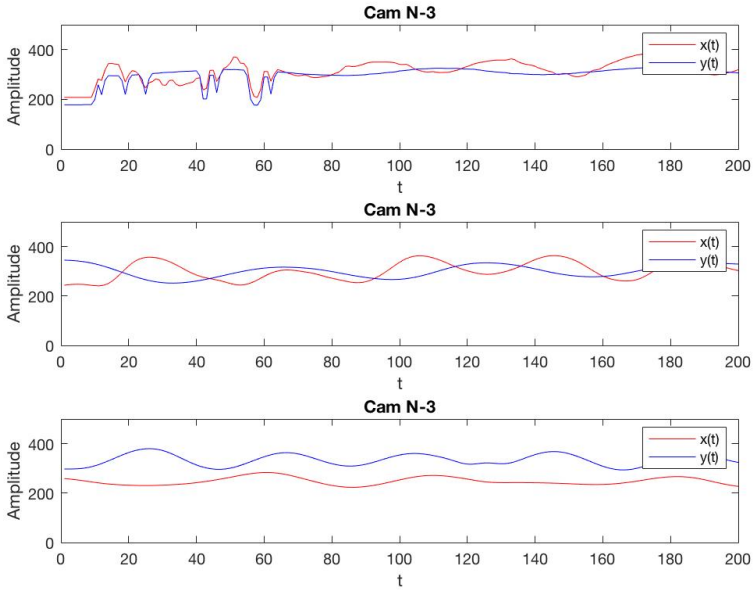

Figure 9: Displacement in x and y direction in three cameras

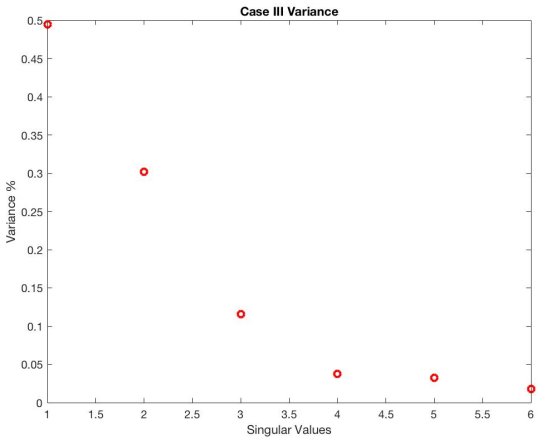

Figure 10: Squared Singular Values in case 3

4.4 Test 4

Test 4 is the horizontal spring and mass system doing simple harmonic motion and pendulum motion while rotating. The extracted path is in Figure 12. The data from the first camera is not stable in the first few seconds because the mass is released off-center. Later, the data shows a simple harmonic pattern. Since camera3 is horizontal, the x and y values are flipped. In this case, we are dealing with the motion in multiple directions, so we are expecting more principle components. Then we apply PCA, we get the singular values, square them. There are 6 singular values meaning six directions. We can see the first direction dominates the

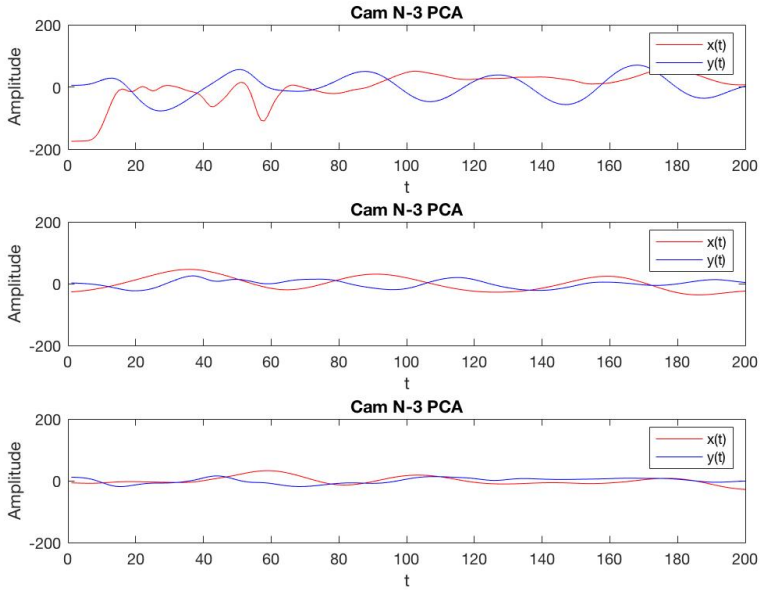

Figure 11: Principle Components Projection

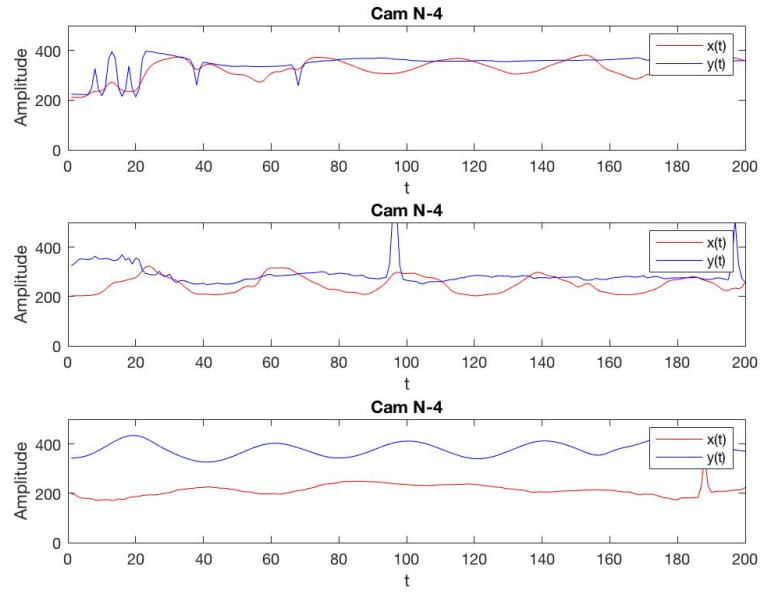

Figure 12: Displacement in x and y direction in three cameras

squared diagonal values with approximately 45%, the second direction with 25 %, the third direction with 15% and two with 5%. in Figure 13. Other directions are not 0 but relatively small. We can identify three principle components. This is expected since case 4 involves displacement in z direction. Pendulum motion and rotation cause changes in other directions as well. Draw the principle components projection after smoothing with Gaussian filter, we

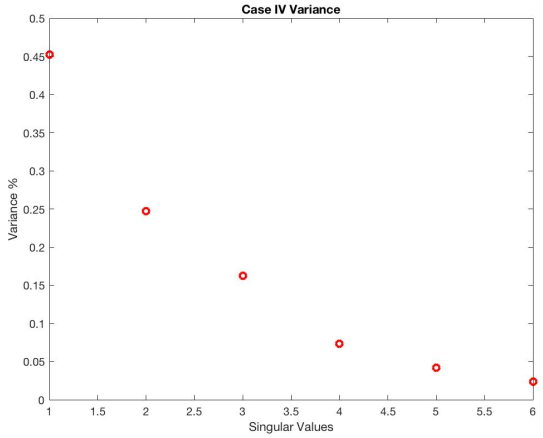

Figure 13: Squared Singular Values in case 4

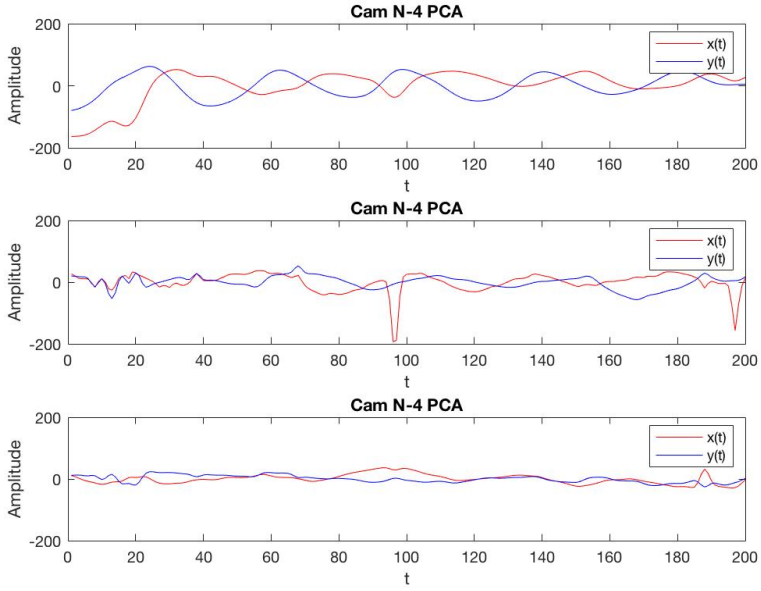

Figure 14: Principle Components Projection

get Figure 14. Other directions are there for comparison purpose only.

5 Summary and Conclusions

In this project, we explore a powerful application of SVD: Principle Components Analysis. PCA enables us to identify the major components in data and removes redundancy. In our experiments, PCA successfully reduce the data dimensions under different circumstances. Even though the data is noisy or involves multiple motions together, PCA still returns a

satisfactory result.

References

[1] Part3 : Computation Methods for Data Analysis. Data-Driven Modeling & Scientific Computation: Methods for Complex Systems Big Data, by J. Nathan Kutz, OUP Oxford, 2013, available at `http://faculty.washington.edu/kutz/582.pdf`.

A MATLAB commands

Documentation for those commands can be found in MATLAB using *help command_name*.

A.1 Data Modification

size(X) Returns the size of X.

rgb2gray(X) Turn rgb value into gray scale.

ind2sub(size(V), INDEX) Determines the subscripts in a size V data of elements with linear indices.

repmat Replicate arrays.

smoothdata(X,"filtername) Apply certain filter and smooth data

svd(X) find the SVD of matrix X. Returns u, s, v. One can add 'econ' to perform a reduced SVD rather a full SVD.

A.2 Plotting

plot(X,Y) Plots 2-dim data.

subplot(i,j,k)) Draw multiple plots in one figure. i and j specifies the lay out, k is the index of the plots.

legend, xlab, ylab, title Labels figure information.

B MATLAB code

```
%HW3
%% Ideal Case
clear all; close all; clc;
%load data
load cam1_1.mat
[x1, y1, ~, numFrames1] = size(vidFrames1_1);
```

```
load cam2_1.mat
[x2, y2, ~, numFrames2] = size(vidFrames2_1);
load cam3_1.mat
[x3, y3, ~, numFrames3] = size(vidFrames3_1);

% %View frames
% for k = 1 : numFrames3
%     mov(k).cdata = vidFrames3_1(:,:,:,k);
%     mov(k).colormap = [];
% end
% for j=1:numFrames3
%     X=frame2im(mov(j));
%     imshow(X); drawnow
% end
% for eyeball ranges purpose only
%image(vidFrames3_1(:,:,:,200))
%uniform number of frames
numFrames = min([numFrames1,numFrames2,numFrames3]);
frames = zeros(x1,y1);
% record the location of the pixels with maximum value
X1 = []; Y1 = [];
X2 = []; Y2 = [];
X3 = []; Y3 = [];
%convert frames from rgb to gray
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames1_1(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:200,:) = 0;
 frames(450:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X1 = [X1 x];
  Y1 = [Y1 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames2_1(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:100,:) = 0;
 frames(450:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X2 = [X2 x];
  Y2 = [Y2 y];
end
for i = 1:numFrames
```

```
 frames(:,:) = rgb2gray(vidFrames3_1(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 %this cam3 is rotated
 frames(:,1:200) = 0;
 frames(:,450:end) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X3 = [X3 x];
  Y3 = [Y3 y];
end

% shift phase to show simple harmonic
[~,ind1] = min(X1(1:26));X1 = X1(ind1:ind1+200);Y1 = Y1(ind1:ind1+200);
[~,ind2] = min(X2(1:26));X2 = X2(ind2:ind2+200);Y2 = Y2(ind2:ind2+200);
[~,ind3] = min(Y3(1:26));Y3 = Y3(ind3:ind3+200);X3 = X3(ind3:ind3+200);
figure(1)
%  plot(X1,'b')
% axis([0 200 0 500])

% filter trajectory
%Gaussian filter to smooth data
X1 = smoothdata(X1,'gaussian');Y1 = smoothdata(Y1,'gaussian');
X2 = smoothdata(X2,'gaussian');Y2 = smoothdata(Y2,'gaussian');
X3 = smoothdata(X3,'gaussian');Y3 = smoothdata(Y3,'gaussian');
figure(1)
subplot(3,1,1)
plot(X1,'r'); hold on
plot(Y1,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-1');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,2)
plot(X2,'r'); hold on
plot(Y2,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-1');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,3)
plot(X3,'r'); hold on
plot(Y3,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-1');
legend('x(t)','y(t)')
axis([0 200 0 500])
```

```
%Stack to perform SVD and PCA
%diagonalizing
figure(2)
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n]=size(X); % compute data size
mn=mean(X,2); % compute mean for each row
X=X-repmat(mn,1,n); % subtract mean
[u,s,v] = svd(X/sqrt(n-1)); %perform SVD
lambda = diag(s).^2; %produce diagonal covariance
Y = u' * X;
plot(lambda/sum(lambda),'ro','Linewidth',[2])
title('Case I Variance'); xlabel('Singular Values');ylabel('Variance %')

% plot PCA
figure(3)
subplot(3,1,1)
plot(smoothdata(Y(1,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(2,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-1 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,2)
plot(smoothdata(Y(3,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(4,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-1 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,3)
plot(smoothdata(Y(5,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(6,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-1PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])

%% Noisy case
clear all; close all; clc;
%load data
load cam1_2.mat
[x1, y1, ~, numFrames1] = size(vidFrames1_2);
load cam2_2.mat
[x2, y2, ~, numFrames2] = size(vidFrames2_2);
load cam3_2.mat
[x3, y3, ~, numFrames3] = size(vidFrames3_2);
```

```
% %View frames
% for k = 1 : numFrames3
%     mov(k).cdata = vidFrames3_2(:,:,:,k);
%     mov(k).colormap = [];
% end
% for j=1:numFrames3
%     X=frame2im(mov(j));
%     imshow(X); drawnow
% end

% for eyeball ranges purpose only
%image(vidFrames3_1(:,:,:,200))
%uniform number of frames
numFrames = min([numFrames1,numFrames2,numFrames3]);
frames = zeros(x1,y1);
% record the location of the pixels with maximum value
X1 = []; Y1 = [];
X2 = []; Y2 = [];
X3 = []; Y3 = [];
%convert frames from rgb to gray
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames1_2(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:200,:) = 0;
 frames(450:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X1 = [X1 x];
  Y1 = [Y1 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames2_2(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:100,:) = 0;
 frames(450:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X2 = [X2 x];
  Y2 = [Y2 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames3_2(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 %this cam3 is rotated
```

```
 frames(:,1:200) = 0;
 frames(:,450:end) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X3 = [X3 x];
  Y3 = [Y3 y];
end

% shift phase to show simple harmonic
[~,ind1] = min(X1(1:26));X1 = X1(ind1:ind1+200);Y1 = Y1(ind1:ind1+200);
[~,ind2] = min(X2(1:26));X2 = X2(ind2:ind2+200);Y2 = Y2(ind2:ind2+200);
[~,ind3] = min(Y3(1:26));Y3 = Y3(ind3:ind3+200);X3 = X3(ind3:ind3+200);
figure(1)
%  plot(X1,'b')
% axis([0 200 0 500])
%Gaussian filter to smooth data
X1 = smoothdata(X1,'gaussian');Y1 = smoothdata(Y1,'gaussian');
X2 = smoothdata(X2,'gaussian');Y2 = smoothdata(Y2,'gaussian');
X3 = smoothdata(X3,'gaussian');Y3 = smoothdata(Y3,'gaussian');
figure(1)
subplot(3,1,1)
plot(X1,'r'); hold on
plot(Y1,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-2');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,2)
plot(X2,'r'); hold on
plot(Y2,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-2');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,3)
plot(X3,'r'); hold on
plot(Y3,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-2');
legend('x(t)','y(t)')
axis([0 200 0 500])
%Stack to perform SVD and PCA
%diagonalizing
figure(2)
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n]=size(X); % compute data size
mn=mean(X,2); % compute mean for each row
```

```
X=X-repmat(mn,1,n); % subtract mean
[u,s,v] = svd(X/sqrt(n-1)); %perform SVD
Y = u' * X;
lambda = diag(s).^2; %produce diagonal covariance
plot(lambda/sum(lambda),'ro','Linewidth',[2])
title('Case II Variance'); xlabel('Singular Values');ylabel('Variance %')
% plot PCA
figure(3)
subplot(3,1,1)
plot(smoothdata(Y(1,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(2,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-2 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,2)
plot(smoothdata(Y(3,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(4,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-2 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,3)
plot(smoothdata(Y(5,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(6,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-2 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])

%% Horizontal Displacement

clear all; close all; clc;
%load data
load cam1_3.mat
[x1, y1, ~, numFrames1] = size(vidFrames1_3);
load cam2_3.mat
[x2, y2, ~, numFrames2] = size(vidFrames2_3);
load cam3_3.mat
[x3, y3, ~, numFrames3] = size(vidFrames3_3);

% %View frames
% for k = 1 : numFrames3
%     mov(k).cdata = vidFrames3_3(:,:,:,k);
%     mov(k).colormap = [];
% end
% for j=1:numFrames3
```

```
%     X=frame2im(mov(j));
%     imshow(X); drawnow
% end
% for eyeball ranges purpose only
%image(vidFrames3_1(:,:,:,200))
%uniform number of frames
numFrames = min([numFrames1,numFrames2,numFrames3]);
frames = zeros(x1,y1);
% record the location of the pixels with maximum value
X1 = []; Y1 = [];
X2 = []; Y2 = [];
X3 = []; Y3 = [];
%convert frames from rgb to gray
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames1_3(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:200,:) = 0;
 frames(500:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X1 = [X1 x];
  Y1 = [Y1 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames2_3(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:200,:) = 0;
 frames(400:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X2 = [X2 x];
  Y2 = [Y2 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames3_3(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 %this cam3 is rotated
 frames(:,1:200) = 0;
 frames(:,450:end) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X3 = [X3 x];
  Y3 = [Y3 y];
end
```

```
% shift phase to show simple harmonic
[~,ind1] = min(X1(1:26));X1 = X1(ind1:ind1+200);Y1 = Y1(ind1:ind1+200);
[~,ind2] = min(X2(1:26));X2 = X2(ind2:ind2+200);Y2 = Y2(ind2:ind2+200);
[~,ind3] = min(Y3(1:26));Y3 = Y3(ind3:ind3+200);X3 = X3(ind3:ind3+200);
figure(1)
%  plot(X1,'b')
% axis([0 200 0 500])

X1 = smoothdata(X1,'gaussian');Y1 = smoothdata(Y1,'gaussian');
X2 = smoothdata(X2,'gaussian');Y2 = smoothdata(Y2,'gaussian');
X3 = smoothdata(X3,'gaussian');Y3 = smoothdata(Y3,'gaussian');
figure(1)
subplot(3,1,1)
plot(X1,'r'); hold on
plot(Y1,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-3');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,2)
plot(X2,'r'); hold on
plot(Y2,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-3');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,3)
plot(X3,'r'); hold on
plot(Y3,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-3');
legend('x(t)','y(t)')
axis([0 200 0 500])
%Stack to perform SVD and PCA
%diagonalizing
figure(2)
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n]=size(X); % compute data size
mn=mean(X,2); % compute mean for each row
X=X-repmat(mn,1,n); % subtract mean
[u,s,v] = svd(X/sqrt(n-1)); %perform SVD
lambda = diag(s).^2; %produce diagonal covariance
Y = u' * X;
plot(lambda/sum(lambda),'ro','Linewidth',[2])
title('Case III Variance'); xlabel('Singular Values');ylabel('Variance %')
```

```
% plot PCA
figure(3)
subplot(3,1,1)
plot(smoothdata(Y(1,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(2,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-3 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,2)
plot(smoothdata(Y(3,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(4,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-3 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,3)
plot(smoothdata(Y(5,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(6,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-3 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])

%% Horizontal displacement and rotation

clear all; close all; clc;
%load data
load cam1_4.mat
[x1, y1, ~, numFrames1] = size(vidFrames1_4);
load cam2_4.mat
[x2, y2, ~, numFrames2] = size(vidFrames2_4);
load cam3_4.mat
[x3, y3, ~, numFrames3] = size(vidFrames3_4);
% %View frames
% for k = 1 : numFrames3
%     mov(k).cdata = vidFrames3_4(:,:,:,k);
%     mov(k).colormap = [];
% end
% for j=1:numFrames3
%     X=frame2im(mov(j));
%     imshow(X); drawnow
% end
% for eyeball ranges purpose only
%image(vidFrames3_1(:,:,:,200))
%uniform number of frames
numFrames = min([numFrames1,numFrames2,numFrames3]);
```

```
frames = zeros(x1,y1);
% record the location of the pixels with maximum value
X1 = []; Y1 = [];
X2 = []; Y2 = [];
X3 = []; Y3 = [];
%convert frames from rgb to gray
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames1_4(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:200,:) = 0;
 frames(500:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X1 = [X1 x];
  Y1 = [Y1 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames2_4(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 frames(1:200,:) = 0;
 frames(400:end,:) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X2 = [X2 x];
  Y2 = [Y2 y];
end
for i = 1:numFrames
 frames(:,:) = rgb2gray(vidFrames3_4(:,:,:,i));
 %searching for max value along the trajectory
 %eyeball the approximate area
 %this cam3 is rotated
 frames(:,1:200) = 0;
 frames(:,450:end) = 0;
 [~,ind] = max(frames(:));
 [x,y] = ind2sub(size(frames),ind);
  X3 = [X3 x];
  Y3 = [Y3 y];
end

% shift phase to show simple harmonic overall 392 show 200 frames max
[~,ind1] = min(X1(1:40));X1 = X1(ind1:ind1+200);Y1 = Y1(ind1:ind1+200);
[~,ind2] = min(X2(1:40));X2 = X2(ind2:ind2+200);Y2 = Y2(ind2:ind2+200);
[~,ind3] = min(Y3(1:40));Y3 = Y3(ind3:ind3+200);X3 = X3(ind3:ind3+200);
figure(1)
%  plot(X1,'b')
```

```
% axis([0 200 0 500])

%smooth data using Gaussian Filter
X1 = smoothdata(X1,'gaussian');Y1 = smoothdata(Y1,'gaussian');
X2 = smoothdata(X2,'gaussian');Y2 = smoothdata(Y2,'gaussian');
X3 = smoothdata(X3,'gaussian');Y3 = smoothdata(Y3,'gaussian');

figure(1)
subplot(3,1,1)
plot(X1,'r'); hold on
plot(Y1,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-4');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,2)
plot(X2,'r'); hold on
plot(Y2,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-4');
legend('x(t)','y(t)')
axis([0 200 0 500])
subplot(3,1,3)
plot(X3,'r'); hold on
plot(Y3,'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-4');
legend('x(t)','y(t)')
axis([0 200 0 500])

% Stack to perform SVD and PCA
%diagonalizing
figure(2)
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n]=size(X); % compute data size
mn=mean(X,2); % compute mean for each row
X=X-repmat(mn,1,n); % subtract mean
[u,s,v] = svd(X/sqrt(n-1)); %perform SVD
lambda = diag(s).^2; %produce diagonal covariance
Y = u' * X;
plot(lambda/sum(lambda),'ro','Linewidth',[2])
title('Case IV Variance'); xlabel('Singular Values');ylabel('Variance %')

% plot PCA
figure(3)
subplot(3,1,1)
plot(smoothdata(Y(1,:),'gaussian'),'r'); hold on
```

```
plot(smoothdata(Y(2,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-4 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,2)
plot(smoothdata(Y(3,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(4,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-4 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
subplot(3,1,3)
plot(smoothdata(Y(5,:),'gaussian'),'r'); hold on
plot(smoothdata(Y(6,:),'gaussian'),'b')
xlabel('t'); ylabel('Amplitude');
title('Cam N-4 PCA');
legend('x(t)','y(t)')
axis([0 200 -200 200])
```