Principal Component Analysis of a Moving Object:
Determining Primary Coordinates of Motion

Franklin Williams
AMATH 582
February 21, 2020

Abstract

Principal component analysis (PCA) is a widely-used technique in the field of data science, which takes high dimensional data and attempts to identify the linearly uncorrelated variables. By identifying these principal components, high dimensional data can be reduced to a lower dimensional space for predictive models and exploratory analysis. In the following experiments, we use PCA to identify the primary axes of motion of a weight attached to a spring.

Introduction

This report begins with an introduction to the practicality of PCA as well as an overview of data collected and experiments run. In Section II, a theoretical background of the techniques used will be discussed. Section III covers the algorithm implementation and development in MATLAB. Section IV provides the computational results of our tests. To conclude, Section V summarizes the results and ends with final remarks.

Within the experiments, three video cameras (probes) record a weight attached to a spring as it is perturbed. The probes capture two-dimensional data at arbitrary angles of the weight in motion. In the simplest case, the weight is perturbed in just the *z*-direction and creates a simple harmonic motion up and down. With prior knowledge of physics, one could solve the following governing equation for *f(t)* to fully describe the system:

$$\frac{d^2 f(t)}{dt^2} = -\omega^2 f(t) \tag{1}$$

This has a well-known solution with that can be described with a one degree of freedom system. To fully capture the information within this weight-spring system, one would only need one probe recording the system with the *x*-*y* plane orthogonal to the *z*-axis.

Much like in real-world data science problems, rather than possessing perfect knowledge of the underlying system, we have collected (possibly) redundant and noisy data. In addition to the simple harmonic case above, video data were collected from arbitrary angles as the spring displayed the following: simple harmonic motion with noisy data (camera shake), horizontal displacement (pendulum motion), and horizontal displacement with rotation (rotation combined with pendulum). By utilizing the concepts within PCA, the axes in which the weight moves are empirically determined, despite the imperfect data.

Theoretical Background

PCA is built on the underlying theory described in Singular Value Decomposition (SVD). SVD is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications (Kutz). In linear algebra, matrix multiplication can be thought of as a rotation and a stretch (compression) of a given vector. In the case of **y**=**Ax** which projects a hypersphere onto a hyper ellipse, the matrix **A** can be rewritten as:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{*} \tag{2}$$

$\boldsymbol{U} \in \mathbb{C}^{m\times m}$ is unitary
$\boldsymbol{V} \in \mathbb{C}^{n\times n}$ is unitary
$\boldsymbol{\Sigma} \in \mathbb{R}^{m\times n}$ is diagonal

The SVD of the matrix **A** shows the matrix first applies a unitary transformation preserving the unit sphere via $\mathbf{V}^{*}$. Following, **Σ** stretches the matrix to create an ellipse with principal semi-axes. Finally, the generated hyper-ellipse is rotated by **U**. Every matrix $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ has an SVD. Two particularly important properties of SVD are as follows:

A *is the sum of r rank-one matricies*

$$\boldsymbol{A} = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^{*} \tag{3}$$

For any N so that $0 \leq N \leq r$*, we can define the partial sum*

$$\boldsymbol{A}_N = \sum_{j=1}^{r} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^{*} \tag{4}$$

And if N = $\min\{m, n\}$*, define* $\sigma_{\mathrm{N}+1} = 0$*. Then*

$$\|A - A_N\|_2 = \sigma_{\mathrm{N}+1} \tag{5}$$

Likewise, if using the Frobenius norm, then

$$\|A - A_N\|_2 = \sqrt{\sigma_{\mathrm{N}+1}^2 + \sigma_{\mathrm{N}+2}^2 + \cdots + \sigma_{\mathrm{r}}^2} \tag{6}$$

Following this logic, we can generate the best approximation of a hyper-ellipsoid with a line segment by taking the longest axis (the one associated with the singular value σ_1). We can then generalize into the best approximation for 2- or *n*-dimensional ellipse by taking the longest 2 or *n* number of axes. After *r* number of steps, the energy within **A** is completely captured. The SVD allows us to project the high dimensional space onto lower dimensional spaces in a formal way similar to the least-squares fit common in statistics. The key to analyzing a given data set is reduce the dimensions of the data set's covariance matrix and find the principal components by reducing redundancy.

Let the covariance matrix of a data set be defined as:

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^{\mathbf{T}} \tag{7}$$

where **X** is a vector of the data gathered in an experiment.

Using the SVD, the covariance matrix of a system of data can be diagonalized to understand the level of importance of various dimensions. We assume that the large diagonal values that correspond to the variance of that element are of larger importance than the smaller. Additionally, large off diagonal elements are assumed to be redundant while small off diagonal elements are assumed to be statistically independent. The high dimensional space can be projected into a lower dimensional space, with unnecessary (redundant or uninformative) dimensions compressed to make for easier analysis, modeling, and interpretation. Through this process, the system can be written in terms of its principal components.

Algorithm Implementation and Development

Data used in the four experiments come from the course website. Twelve data sets in total were used, 3 probes recording 4 perturbations of the weight-spring system. Each data set was a 4-dimensional tensor object that contained frame data, 480x640 pixels and their corresponding 3 RGB values, for length T. Where T varied roughly 200 frames to 400 frames, depending on the video length.

To track the weight through time, first each frame the data sets were converted from RGB to grayscale. A light was attached to the weight, and by locating the brightest pixel on the display, the weight can be tracked. The lengths of the probe recordings varied in size. They were edited so that each vector of *x* and *y* coordinates were of the same length. Figure (1) shows the locations of the weight in the horizontal displacement experiment.

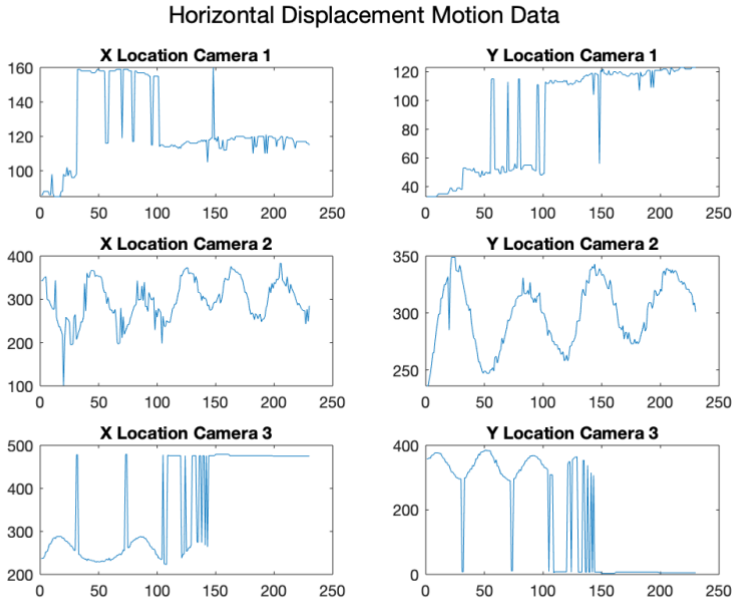

Figure (1): *x*-*y* locations of probe data in horizontal displacement experiment

For each experiment, the data collected were gathered into a single matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{y_1} \\ \mathbf{x_2} \\ \mathbf{y_2} \\ \mathbf{x_3} \\ \mathbf{y_3} \end{bmatrix} \tag{8}$$

where $\mathbf{x_n}$ and $\mathbf{y_n}$ are 1×T row vectors of the *x*-*y* location of the weight through time. For each of the four experiments PCA analysis was conducted. For each row in the matrix **X**, the mean was calculated and subtracted from each value in the row. Mean centering is needed to ensure that the first principal component describes the direction of maximum variance, rather than the mean of the data.

The SVD was conducted on each demeaned $\mathbf{C_X}$. From the SVD we find the $\mathbf{\Sigma}$ and the values along its diagonal. Each diagonal value represents a portion of the total variance within the covariance matrix. By plotting each of these values, σ_j, we can identify the principal components, axes of motion, in which the weight is moving.

In addition to the classical PCA conducted, Robust PCAs (RPCA) were performed to extract a better signal from the noisy data. With RPCA, rather than using the L_2 norm, the L_1 norm is used to avoid problems with corrupted data or data with large noise shocks. The MATLAB program **inexact_alm_rpca** was used to conduct the RPCA analysis.

Computational Results

Experiment 1: Simple Harmonic Motion

Below we show the values of the modes calculated from the PCA on a normal and log scale. We see that the first mode clearly explains the majority of variance. However, we do notice that modes 2 and 3 are non-zero. We interpret mode 1 to be the *z*-axis, as this was the axis in which the weight moved. In a perfect system, we would see mode 1 explaining 100% of the variance. However, due to imperfections in our system (slight camera shake, slight perturbations in the *x*-*y* plane) we can expect some variance to be explained by other modes. A 2-dimensional estimation of the experiment would explain roughly 87% of variance in the data.

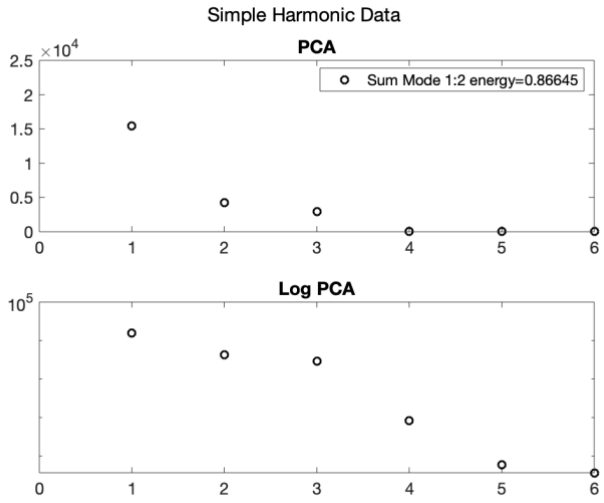

Figure (2): PCA analysis of Simple Harmonic Motion

Experiment 2: Noisy Simple Harmonic Motion

In Experiment 2, noise was added to the data from the simple harmonic motion experiment. Because the underlying signal is the same, we see a similar PCA analysis as before. However, in this case, we see a drop in variance explained due to mode 1, the *z*-axis. The noise added to the data has caused spurious correlations to occur and the SVD algorithm gives more weight to modes 2-6. To remedy the added noise, an RPCA analysis was conducted through the

inexact_alm_rpca algorithm. In the graph below, we see a dramatic drop in variance explained by the modes 2 and 3, with an increase in mode 1. By comparison, a 2-dimensional approximation of the experiment would account for only 75% of the variance in the noisy data compared to nearly 100% through the RPCA.

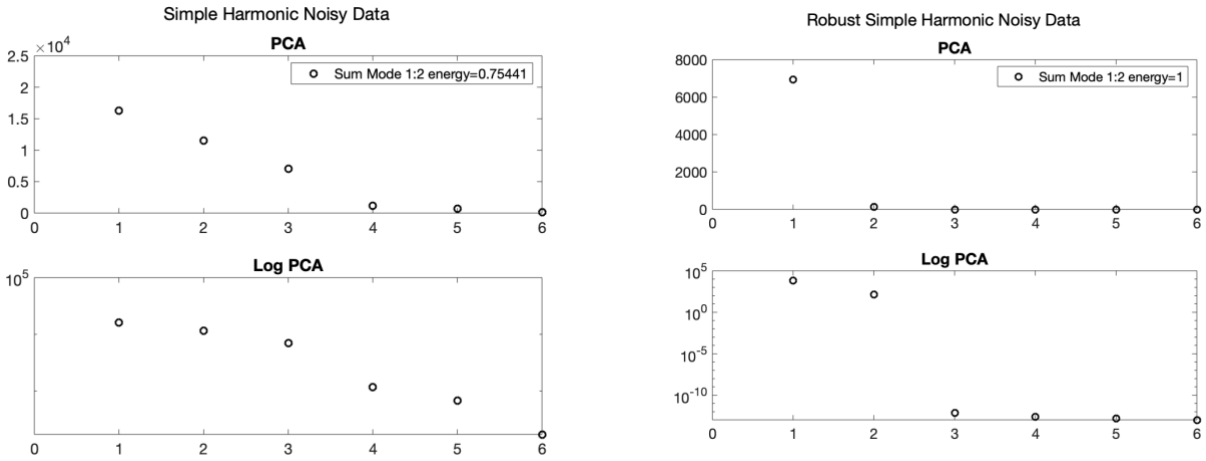

Figure (3): (left) PCA analysis of Noisy Simple Harmonic Motion
(right) RPCA analysis of Noisy Simple Harmonic Motion

Experiment 3: Horizontal Displacement

While the following figure closely resembles the original experiment of simple harmonic motion, the modes have different interpretations. Again, the PCA algorithm identified one mode that accounts for the majority of variance within the data. However, in this case, rather than mode 1 corresponding to the *z*-axis, the mode corresponds to motion along the *x*-*y* plane. A 2-dimensional approximation of the data captures roughly 96% of the variation within the data. Again, small amounts of variance can be seen in modes 2 and 3, however this is due to imperfections within the weight-spring system, rather than the PCA algorithm misattributing variance.

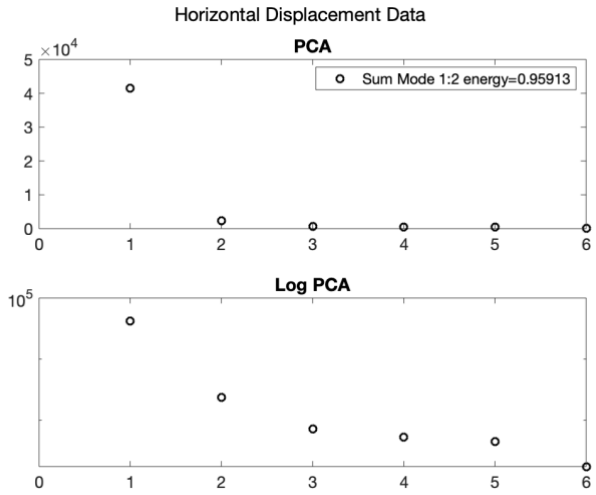

Figure (4): PCA analysis of Horizontal Displacement Motion

Experiment 4: Horizontal Displacement and Rotation

In the final experiment, a horizontal perturbation and rotation were placed on the weight-spring system. Unlike the other experiments, we expect to see multiple modes needed in order to fully explain the variance. Within this system, we have motion in the *x*-*y* plane, the *z*-axis harmonic motion, and a rotation around the *z*-axis. Because this system is more complicated, we do not expect to be able to clearly explain the variance of the weight with a 1- or 2-dimensional approximation. Below we see this is the case in the PCA plots.

Again, there is a large amount of variance explained by mode 1, however the first two modes only capture roughly 75% of the variance in the data (compared to roughly 90% in the simple harmonic motion case).

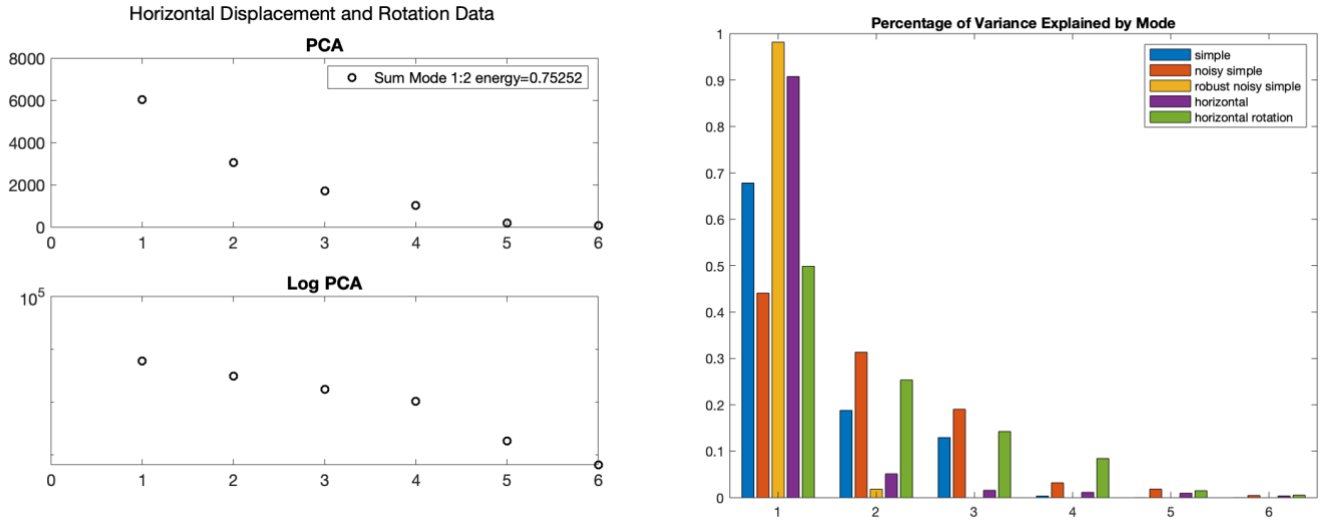

Figure (5 and 6): (left) PCA analysis of Horizontal Displacement and Rotation Motion (right) Percentage of Variance Explained by Mode by Experiment

Summary and Conclusions

Overall, the PCA and RPCA analysis of the systems worked as theorized. The probes collected redundant and noisy data, much like real-world systems across domains (biology, physics, finance etc.) The motion of the system was tracked, and a covariance matrix of each system was constructed. SVD analysis of each covariance matrix identified the variance explained by each mode of the system. Figure 6 summarizes the percentage of variance explained in the 5 PCA graphs. It appears that the Simple Harmonic Motion can be closely approximated with 2 modes. The Noisy Simple Harmonic is largely corrupted and requires 3 modes to generate a close approximation of the system. On the other hand, using a RPCA algorithm for the noisy data reduces the number of modes needed to 1. The Horizontal Displacement experiment can be closely approximated with one mode. Finally, due to the complex nature of the Horizontal Displacement with Rotation, multiple modes are required to generate a close approximation to the system.

Appendix A

```
%% functions
```

Point tracker identifies x-y coordinates the brightest pixel captured by the probes

```
function [x_vt, x_it, y_vt, y_it] = pointTracker(video)
x_vt = [];
x_it = [];
y_vt = [];
y_it = [];
    for f=1:size(video,4)
        frame=rgb2gray(video(:,:,:,f));
        [M,I]=max(frame(:));
        [x_i, y_i]=ind2sub(size(frame), I);
        [x_v, y_v]=ind2sub(size(frame), M);
        x_vt=[x_vt x_v];
        x_it=[x_it x_i];
        y_vt=[y_vt y_v];
        y_it=[y_it y_i];
    end

end
```

producePCAProj takes the data matrix **X,** demeans each row and computs the SVD of the covariance matrix

```
function [u,s,v,lambda,sigma,Y] = producePCProj(X)
[m,n]=size(X);   %  compute data size
mn=mean(X,2); %  compute mean for each row
X=X-repmat(mn,1,n);  % subtract mean
Cx=(1/(n-1))*X*X'; % Cov matrix
[u,s,v]=svd(Cx);  % perform the SVD
lambda=diag(s).^2;  % produce diagonal variances
sigma=diag(s);
Y=u'*Cx;  % produce the principal components projection
end
```

plotPCA plots the PCA and Log scale PCA and calculates the variance explained in the first 2 modes.

```
function plotPCA(sigma, fignum, u, grouptitle, ylim)
x=[1:1:size(u,1)];
figure(fignum);
sgtitle(grouptitle);

subplot(2,1,1), plot(sigma,'ko','Linewidth',[1.5]);
set(gca,'Fontsize',[13],'Xtick',[0:6]);
axis([0 6 0 ylim]);
title('PCA');
energy2=sum(sigma(1:2))/sum(sigma);
leg2 = strcat('Sum Mode 1:2 energy=', num2str(energy2));
legend(leg2, 'Location', 'NorthEast') ;

subplot(2,1,2), semilogy(sigma,'ko','Linewidth',[1.5]);
axis([0 6 0 10^(5)]);
set(gca,'Fontsize',[13],'Ytick',[10^(-15) 10^(-10) 10^(-5) 10^0 10^5],...
      'Xtick',[0:6]);
title('Log PCA');
end
```

Appendix B

```
%% load data
close all; clear all;
for N=[1:3]
    for C=[1:4]
        load(strcat('cam', num2str(N), '_', num2str(C), '.mat'))
    end
end

% simple harmonic
[xv1_1, xi1_1, yv1_1, yi1_1] = pointTracker(vidFrames1_1);
[xv2_1, xi2_1, yv2_1, yi2_1] = pointTracker(vidFrames2_1);
[xv3_1, xi3_1, yv3_1, yi3_1] = pointTracker(vidFrames3_1);

% noisy simple harmonic
[xv1_2, xi1_2, yv1_2, yi1_2] = pointTracker(vidFrames1_2);
[xv2_2, xi2_2, yv2_2, yi2_2] = pointTracker(vidFrames2_2);
[xv3_2, xi3_2, yv3_2, yi3_2] = pointTracker(vidFrames3_2);

% horizontal displacement
[xv1_3, xi1_3, yv1_3, yi1_3] = pointTracker(vidFrames1_3);
[xv2_3, xi2_3, yv2_3, yi2_3] = pointTracker(vidFrames2_3);
[xv3_3, xi3_3, yv3_3, yi3_3] = pointTracker(vidFrames3_3);

% horizontal displacement and rotation
[xv1_4, xi1_4, yv1_4, yi1_4] = pointTracker(vidFrames1_4);
[xv2_4, xi2_4, yv2_4, yi2_4] = pointTracker(vidFrames2_4);
[xv3_4, xi3_4, yv3_4, yi3_4] = pointTracker(vidFrames3_4);


%% align data

% simple harmonic
xv1_1 = xv1_1(1:226); xi1_1 = xi1_1(1:226); yv1_1 = yv1_1(1:226); yi1_1 = yi1_1(1:226);
xv2_1 = xv2_1(1:226); xi2_1 = xi2_1(1:226); yv2_1 = yv2_1(1:226); yi2_1 = yi2_1(1:226);
xv3_1 = xv3_1(1:226); xi3_1 = xi3_1(1:226); yv3_1 = yv3_1(1:226); yi3_1 = yi3_1(1:226);

% noisy simple harmonic
xv1_2 = xv1_2(1:226); xi1_2 = xi1_2(1:226); yv1_2 = yv1_2(1:226); yi1_2 = yi1_2(1:226);
xv2_2 = xv2_2(1:226); xi2_2 = xi2_2(1:226); yv2_2 = yv2_2(1:226); yi2_2 = yi2_2(1:226);
xv3_2 = xv3_2(1:226); xi3_2 = xi3_2(1:226); yv3_2 = yv3_2(1:226); yi3_2 = yi3_2(1:226);

% horizontal displacement
xv1_3 = xv1_3(1:230); xi1_3 = xi1_3(1:230); yv1_3 = yv1_3(1:230); yi1_3 = yi1_3(1:230);
xv2_3 = xv2_3(1:230); xi2_3 = xi2_3(1:230); yv2_3 = yv2_3(1:230); yi2_3 = yi2_3(1:230);
xv3_3 = xv3_3(1:230); xi3_3 = xi3_3(1:230); yv3_3 = yv3_3(1:230); yi3_3 = yi3_3(1:230);

% horizontal displacement and rotation
xv1_4 = xv1_4(1:392); xi1_4 = xi1_4(1:392); yv1_4 = yv1_4(1:392); yi1_4 = yi1_4(1:392);
xv2_4 = xv2_4(1:392); xi2_4 = xi2_4(1:392); yv2_4 = yv2_4(1:392); yi2_4 = yi2_4(1:392);
xv3_4 = xv3_4(1:392); xi3_4 = xi3_4(1:392); yv3_4 = yv3_4(1:392); yi3_4 = yi3_4(1:392);

% X matrix tests 1:4
X1=[xi1_1; yi1_1; xi2_1; yi2_1; xi3_1; yi3_1];
X2=[xi1_2; yi1_2; xi2_2; yi2_2; xi3_2; yi3_2];
X3=[xi1_3; yi1_3; xi2_3; yi2_3; xi3_3; yi3_3];
X4=[xi1_4; yi1_4; xi2_4; yi2_4; xi3_4; yi3_4];


%%
% simple harmonic plots
figure(1); sgtitle('Simple Harmonic Motion Data');
subplot(3,2,1), plot(xi1_1); title('X Location Camera 1');
subplot(3,2,2), plot(yi1_1); title('Y Location Camera 1');

subplot(3,2,3), plot(xi2_1); title('X Location Camera 2');
subplot(3,2,4), plot(yi2_1); title('Y Location Camera 2');

subplot(3,2,5), plot(xi3_1); title('X Location Camera 3');
subplot(3,2,6), plot(yi3_1); title('Y Location Camera 3');

% noisy simple harmonic plots
figure(2); sgtitle('Noisy Simple Harmonic Motion Data');
```

```
subplot(3,2,1), plot(xi1_2); title('X Location Camera 1');
subplot(3,2,2), plot(yi1_2); title('Y Location Camera 1');

subplot(3,2,3), plot(xi2_2); title('X Location Camera 2');
subplot(3,2,4), plot(yi2_2); title('Y Location Camera 2');

subplot(3,2,5), plot(xi3_2); title('X Location Camera 3');
subplot(3,2,6), plot(yi3_2); title('Y Location Camera 3');

% horizontal displacement plots
figure(3); sgtitle('Horizontal Displacement Motion Data');
subplot(3,2,1), plot(xi1_3); title('X Location Camera 1');
subplot(3,2,2), plot(yi1_3); title('Y Location Camera 1');

subplot(3,2,3), plot(xi2_3); title('X Location Camera 2');
subplot(3,2,4), plot(yi2_3); title('Y Location Camera 2');

subplot(3,2,5), plot(xi3_3); title('X Location Camera 3');
subplot(3,2,6), plot(yi3_3); title('Y Location Camera 3');

% horizontal displacement and rotation plots
figure(4); sgtitle('Horizontal Displacement and Rotation Motion Data');
subplot(3,2,1), plot(xi1_4); title('X Location Camera 1');
subplot(3,2,2), plot(yi1_4); title('Y Location Camera 1');

subplot(3,2,3), plot(xi2_4); title('X Location Camera 2');
subplot(3,2,4), plot(yi2_4); title('Y Location Camera 2');

subplot(3,2,5), plot(xi3_4); title('X Location Camera 3');
subplot(3,2,6), plot(yi3_4); title('Y Location Camera 3');

%% PCA

[uS,sS,vS,lambdaS,sigmaS,YS] = producePCProj(X1); %simple harmonic data
[uN,sN,vN,lambdaN,sigmaN,YN] = producePCProj(X2); % noisy simple harmonics PCA
[uH,sH,vH,lambdaH,sigmaH,YH] = producePCProj(X3); % horizontal displacement PCA
[uHR,sHR,vHR,lambdaHR,sigmaHR,YHR] = producePCProj(X4); % horizontal displacement and rotation PCA

lam=0.06; [R1rN,R2rN]=inexact_alm_rpca(X2.',lam); % noisy simple harmonics RPCA
lam=0.1; [R1rH,R2rH]=inexact_alm_rpca(X3.',lam); % horizontal RPCA
lam=0.1; [R1rHR,R2rHR]=inexact_alm_rpca(X4.',lam); % horizontal rotation RPCA

[urN,srN,vrN]=svd(R1rN); sigmarN = diag(srN); % noisy simple harmonics RPCA
[urH,srH,vrH]=svd(R1rH); sigmarH = diag(srH); % horizontal RPCA
[urHR,srHR,vrHR]=svd(R1rHR); sigmarHR = diag(srHR); % horizontal rotation RPCA

% plot PCA
plotPCA(sigmaS, 5, uS, "Simple Harmonic Data", 25000);
plotPCA(sigmaN, 6, uN, "Simple Harmonic Noisy Data", 25000);
plotPCA(sigmaH, 7, uH, "Horizontal Displacement Data", 50000);
plotPCA(sigmaHR, 8, uHR, "Horizontal Displacement and Rotation Data", 8000);

plotPCA(sigmarN, 9, urN, "Robust Simple Harmonic Noisy Data", 8000);
plotPCA(sigmarH, 10, urH, "Robust Horizontal Displacement Data", 13000);
plotPCA(sigmarHR, 11, urHR, "Robust Horizontal Displacement and Rotation Data", 13000);

%% calculate energy

energyS = sigmaS/sum(sigmaS); % simple harmonic POD
energyN = sigmaN/sum(sigmaN); % noisey simple harmonic POD
energyH = sigmaH/sum(sigmaH); % horizontal displacement POD
energyHR = sigmaHR/sum(sigmaHR); % horizontal displacement and rotation POD

energyrN = sigmarN/sum(sigmarN); % robust simple harmonic noisy POD
energyrH = sigmarH/sum(sigmarH); % robust horizontal displacement POD
energyrHR = sigmarHR/sum(sigmarHR); % robust horizontal displacement and rotation POD


figure(12);
bar([energyS, energyN, energyrN, energyH, energyHR]); %plot energy
title('Percentage of Variance Explained by Mode');
legend('simple', 'noisy simple','robust noisy simple' ,'horizontal', 'horizontal rotation');

figure(13);
bar([energyrN, energyrH, energyrHR]);
title('Percentage of Variance Explained by Mode');
```

```
legend('robust noisy simple', 'robust horizontal', 'robust horizontal rotation');

%% Simple Harmonic Comparison
figure(14)
subplot(3,1,1), plot(sigmaS,'ko','Linewidth',[1.5]);
set(gca,'Fontsize',[13],'Xtick',[0:6]);
axis([0 6 0 20000]);
subplot(3,1,2), plot(sigmaN,'ko','Linewidth',[1.5]);
set(gca,'Fontsize',[13],'Xtick',[0:6]);
axis([0 6 0 12000]);
subplot(3,1,3), plot(sigmarN,'ko','Linewidth',[1.5]);
set(gca,'Fontsize',[13],'Xtick',[0:6]);
axis([0 6 0 12000]);
```