AMATH 482 Homework 3

Jeremy Lu

February 24, 2021

Abstract

We want to investigate applications of Principal Component Analysis (PCA), its usefulness, and how noise affects PCA algorithms. To do this, we are given four sets of videos of a spring-mass system, shot from three different angles each. The four cases are labelled as follows: ideal, noisy, horizontal displacement, and horizontal displacement with rotation. After extracting the X and Y displacement of the mass from each of the videos, they were placed in a vector of 6 principal components. Performing PCA on this data shows us that despite the noise, there is one main energy component, indicating that the primary harmonic motion we wish to investigate is only in one dimension.

1 Introduction and Overview

Principal Component Analysis is a powerful tool that can be used to reduce the dimensionality of large data. Another benefit is that noisy and/or redundant data are removed in exchange for a simpler representation, that also takes less computing power to analysis. While this may come at the cost of some accuracy, it makes understanding such large data much simpler. In the case of this project, the aforementioned issue is less of a concern, as our six original variables, the X and Y positions of the spring-mass system from different camera angles, will still be interpretable even after PCA.
The four sets of videos all capture simple harmonic motion, but they have different properties, which will allow us to compare and contrast the results from each case. This show us the effectiveness of PCA on our data, especially when factoring in noise. The four cases, in more detail, are:

- **Ideal case:** Quality recording of spring-mass oscillation, in the z-direction.
- **Noisy case:** Shaky recording of spring-mass oscillation described in the Ideal Case.
- **Horizontal Displacement:** Spring-mass oscillation with mass being released off-centre, causing both x and y motion, as well as in the z-direction.
- **Horizontal Displacement with Rotation:** Rotation is added to the mass, following the same setup in Horizontal Displacement.

To track the mass in the videos, we focus on the light attached, which will have the highest RGB value. Scanning for this light in a defined pixel area of the mass's movement allows us to capture the position of the oscillating mass in each video. The data then takes an $6 \times f$ shape, where f is the number of frames in the videos.
Performing PCA on our data reveals an increase in principal components as we move to the noisy and two horizontal displacement cases. This is consistent with the videos, and shows that our PCA effectively captures the spring-mass motion in our videos.

2 Theoretical Background

2.1 Singular Value Decomposition

Given an $m \times x$ matrix A, the Singular Value Decomposition, or SVD, rewrites the matrix A in the form of $\mathbf{U}\,\Sigma\,\mathbf{V}^*$, where $\mathbf{U}$ and $\mathbf{V}$ are $m \times m$ and $n \times n$ unitary matrices and Σ is a matrix containing the eigenvalues

in the order $\sigma_1 < \sigma_2 < \cdots < \sigma_n$. Geometrically, this is a rotation, rescaling, and second rotation of the original matrix A. The SVD is a valuable tool in reducing data, and is particularly useful in data analysis projects such as this one.

2.2 Principal Component Analysis

Principal Component Analysis, or PCA, is an extension of the SVD, which allows us to analyze the importance of the dimensions within our data. Given a matrix $\mathbf{X}$ that is constructed by multiple vectors of length n, we can compute the covariance matrix $\mathbf{C}_X$ as $\frac{1}{n-1}\mathbf{X}\mathbf{X}^T$. Diagonalizing $\mathbf{C}_X$ puts the matrix into the form $\mathbf{V}\Lambda\mathbf{V}^{-1}$, and the basis of eigenvectors contained in $\mathbf{V}$ are known as the principal components, and are uncorrelated.
Through SVD, we can obtain $\mathbf{U}$ and $\mathbf{V}$ from $\mathbf{X}$, and create a new variable $\mathbf{Y}$=$\mathbf{U}^*\mathbf{X}$, where $\mathbf{Y}$ becomes a projection of $\mathbf{X}$ onto the principal components. Computing the variances of $\mathbf{Y}$, we have:

$$\begin{aligned}\mathbf{C}_Y &= \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T \\ &= \frac{1}{n-1}(\mathbf{U}^*\mathbf{X})(\mathbf{U}^*\mathbf{X})^T \\ &= \frac{1}{n-1}\mathbf{U}^*(\mathbf{X}\mathbf{X}^T)\mathbf{U} \\ &= \frac{1}{n-1}\mathbf{U}^*\mathbf{U}\Sigma^2\mathbf{U}\mathbf{U}^* \\ \mathbf{C}_Y &= \frac{1}{n-1}\Sigma^2\end{aligned}$$

and this gives us the diagonalized covariance matrix of $\mathbf{Y}$. Since the off-diagonal elements of Σ are zero, we again see that the variables in $\mathbf{Y}$ are uncorrelated.

3 Algorithm Implementation and Development

3.1 Data Collection

To obtain the location of the mass in each frame of our videos, we can analyze the position of the light attached to the mass. This white light has the highest RGB value, so searching in a certain area of the video will allow us to accurately pinpoint the mass location.
By analyzing the videos, the pixel area and range of movement of the mass was determined, allowing the rest of the video to be masked, via setting its color to 0. This ensures that the highest brightness in the remaining area of the video will be the light on the mass. At points, the light is rotating and becomes obscured in certain segments and angles; but PCA accounts for variance and redundancy, so losing out on some data from this angle will not cause any significant problems.
One issue that we run into is the difference in the length of videos. To address this, we can take shortened lengths of the longer clips where they are in sync with the motion captured in the minimum video.

3.2 Principal Component Analysis

After obtaining and storing the coordinates for the positional data as $\mathbf{X}$, we can apply PCA to examine the spring-mass system motion. Prior to performing the SVD, we set the mean in each row of $\mathbf{X}$ to be 0. After applying this technique, we can find the principal component projection with $\mathbf{Y}$=$\mathbf{U}^*\mathbf{X}$, along with computing the energies of each dimension with the formula specified below.

Algorithm 1: Principal Component Analysis

Remove mean from $\mathbf{X}$
Perform SVD on $\mathbf{X}$
Obtain $\mathbf{Y}$ using $\mathbf{U}^*\mathbf{X}$
Obtain eigenvalues λ by squaring diagonals of Σ
Find energies using $\frac{\lambda_i}{\sum \lambda_n}$

4 Computational Results

4.1 Ideal Case

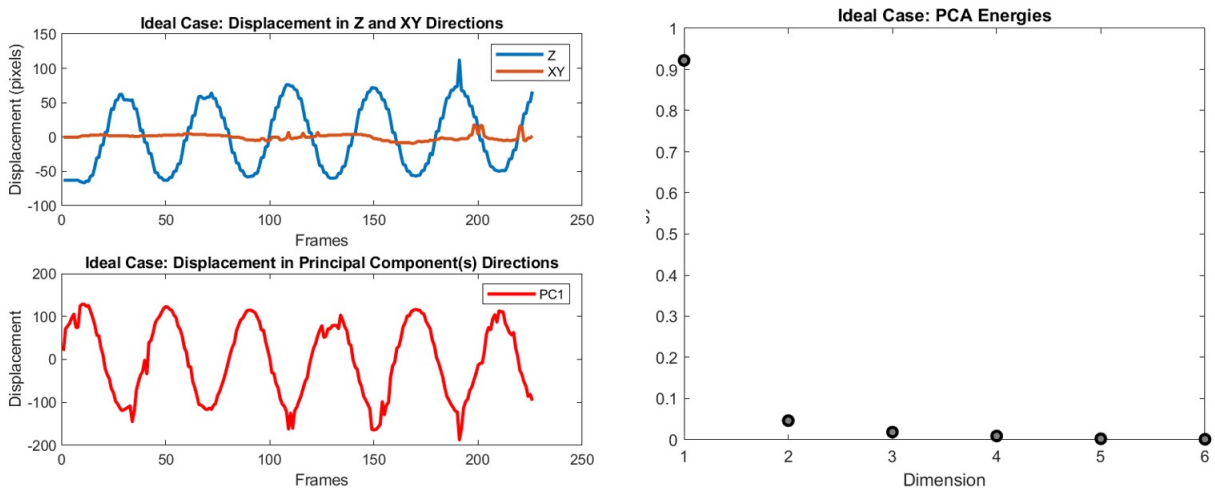

Figure 1: Original Displacement and Principal Component Displacement (left) and PCA Energies (right) in the Ideal Case

In the ideal case, we can see a clear oscillation in the z direction, which we expect from the spring-mass system. In the xy plane, we have a relative straight line, as there are no outside forces or camera shaking present, that would create displacement in this manner.
Looking at the PCA energies, it is clear that there is one primary component, which captures 92.19% of the energy. Projecting on to the basis of this primary component, we have similar motion as what we captured. This indicates that in the ideal case, a singular component can capture the harmonic motion, which is what we expect.

4.2 Noisy Case

In the noisy case, we still see clear oscillation in the z direction, albeit inconsistent in amplitude. Unlike the ideal case, we also have movement in the xy plane, which is caused from shaking of the camera.
Looking at the PCA energies, we can deduce that there might be three principal components, with energies of 55.29% 18.70%, and 12.70%. Plotting the displacement projections, we can see that there is oscillatory behavior, but they are not in sync. Looking at only PC1, which has the largest energy, we see that the primary movement is still in the z direction, and is similar to the original displacement. The main takeaway here is that noisy data can make it harder for us to capture accurate observations with PCA.

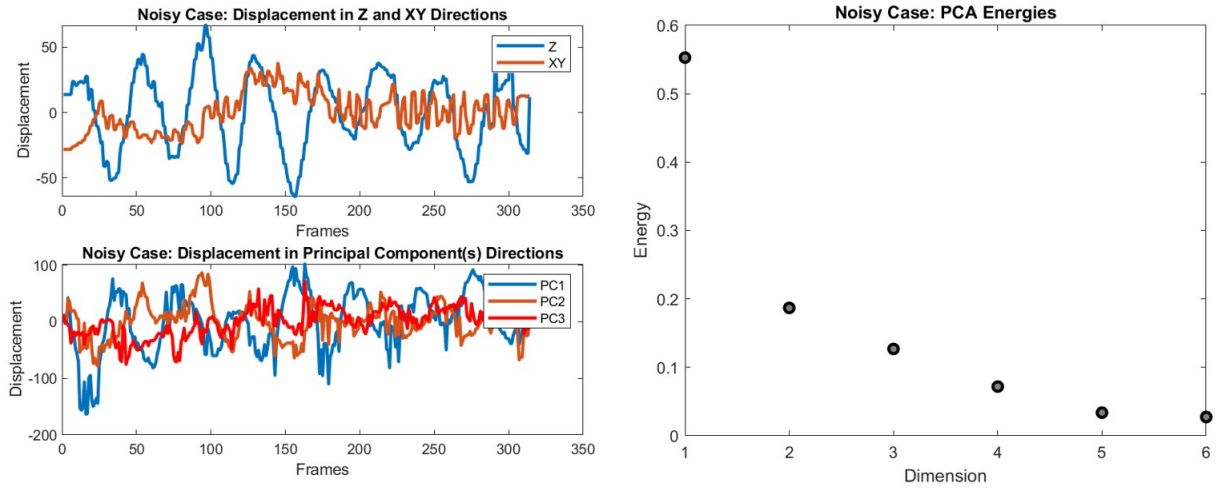

Figure 2: Original Displacement and Principal Component Displacement (left) and PCA Energies (right) in the Noisy Case

4.3 Horizontal Displacement

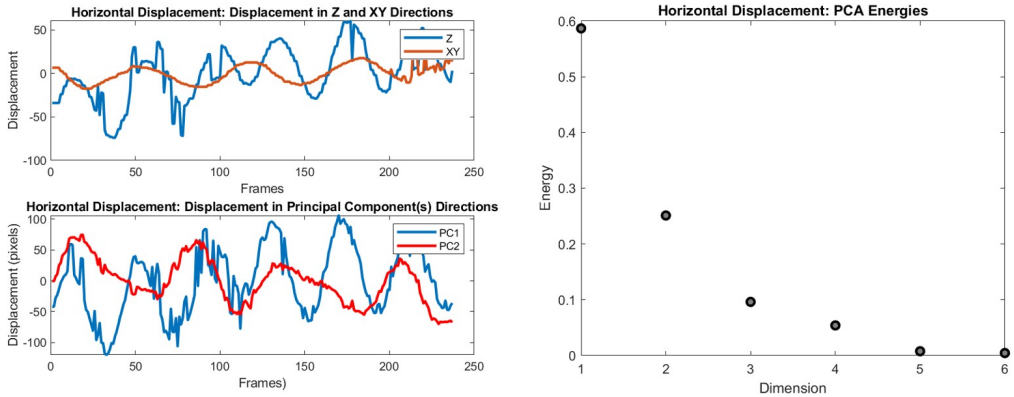

Figure 3: Original Displacement and Principal Component Displacement (left) and PCA Energies (right) with Horizontal Displacement

With horizontal displacement being added to the spring mass systems, there are two types of motion being captured. In the z direction, we have the expected harmonic motion, although the amplitudes are inconsistent like int he noisy case. There is also a pendulum-like motion in the xy plane, as the mass rocks back and forth due to the horizontal displacement.
Looking at the PCA energies, there are two primary principal components, with energies of 58.69% and 25.12%. Plotting the projections onto the principal components, we can see the resemblance of both the the spring-mass and pendulum motion in the z and xy planes respectively, just as in the original displacement graph. Although horizontal displacement was added, PCA still successfully captured the two primary movements recorded. Additionally, we can still see that the main movement is the harmonic motion of the spring-mass system.

4.4 Horizontal Displacement with Rotation

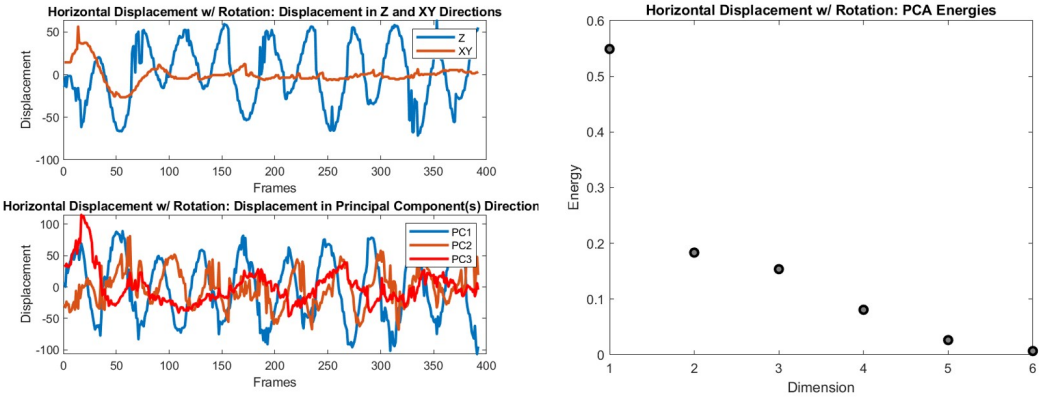

Figure 4: Original Displacement and Principal Component Displacement (left) and PCA Energies (right) with Horizontal Displacement and Rotation

Similar to figure 3, we have harmonic motion in the z direction, along with movement in the xy plane. The key difference is that due to rotation, sometimes the light attached to the mass is obscured or not entirely visible. Thus the data may not always capture the true position of the mass, which explains why the movement in the xy plane is not as consistent and smooth as in figure 3.
Looking at the PCA energies, there are three primary principal components, with energies of 54.90%, 18.34%, and 15.37%. Plotting the projections onto the principal components, we can see a clear harmonic motion in the mass. The other two motions are the rotation and the horizontal displacement of the mass, so PCA is working as expected. Overall, the primary movement is still captured by the oscilliation in the z direction.

5 Summary and Conclusions

Through this project, we are able to see the applications of PCA as well as the effect of noise on its performance. Without any principles of physics, we captured the harmonic motion of a spring-mass system, and identified both primary and additional motions based on the number of principal components.
In the ideal case, PCA worked perfectly in capturing the displacement of the mass. However, when introducing noise in the form of camera shaking, PCA captured additional motions as it identified more principal components than expected. With horizontal displacement being added, PCA revealed both harmonic motion in the z direction as well as the pendulum-like motion in the xy plane. Finally, in the case of horizontal displacement with rotation, PCA has three principal components, which represent the mass's movement as well as its rotation, which is whatwe expect. For all the cases, the primary displacement still corresponded to the spring-mass system's movement, which shows that PCA has the right overall motion, but loses accuracy with noise and other forces being involved.
Overall, PCA is a powerful tool that can reduce data, as well as clean up redundancy and slight inaccuracies in data. However, its effectiveness can be reduced due to noise in the data.

Appendix A MATLAB Functions

- `diag(X)`: Returns the diagonal entries of a given a matrix `X`.
- `max(X)`: Returns the maximum value in a given vector `X`.
- `mean(X, n)` Returns a vector of mean along the n dimension of a given matrix `X`.
- `repmat(X, m, n)`: Returns a matrix with copies of the given matrix `X` in an `m` $\times$ `n` size.

- `size(X)`: Returns the dimensions of a given matrix `X`.
- `sum(X)`: Returns the sum of all the elements in a given vector `X`.
- `svd(X)`: Performs SVD on a given matrix `X` and returns U, Σ, and V.
- `zeros(m,n)`: Returns an `m` $\times$ `n` matrix of zeroes.

Appendix B MATLAB Code

```
clear; close all; clc;

%% Noisy Case
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')

%% Collect position data
ideal_1 = size(vidFrames1_1, 4);
ideal_x1 = zeros(1,ideal_1);
ideal_y1 = zeros(1,ideal_1);

for j = 1:ideal_1
    V = vidFrames1_1(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,ideal_x1(j)] = max(mean(max(V,[],1),3));
    [~,ideal_y1(j)] = max(mean(max(V,[],2),3));
end

ideal_2 = size(vidFrames2_1, 4);
ideal_x2 = zeros(1,ideal_2);
ideal_y2 = zeros(1,ideal_2);

for j = 1:ideal_2
    V = vidFrames2_1(:,:,:,j);
    V(1:80,:,:) = 0;
    V(400:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,380:640,:) = 0;

    [~,ideal_x2(j)] = max(sum(max(V,[],1),3));
    [~,ideal_y2(j)] = max(sum(max(V,[],2),3));
end

ideal_3 = size(vidFrames3_1, 4);
ideal_x3 = zeros(1,ideal_3);
ideal_y3 = zeros(1,ideal_3);

for j = 1:ideal_3
    V = vidFrames3_1(:,:,:,j);
    V(1:200,:,:) = 0;
```

```
    V(400:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,540:640,:) = 0;

    [~,ideal_x3(j)] = max(sum(max(V,[],1),3));
    [~,ideal_y3(j)] = max(sum(max(V,[],2),3));
end

%% Adjust video lengths
ideal_x2 = ideal_x2(10:235);
ideal_y2 = ideal_y2(10:235);
ideal_x3 = ideal_x3(1:226);
ideal_y3 = ideal_y3(1:226);

%% Perform PCA
X = [ideal_x1;ideal_y1;ideal_x2;ideal_y2;ideal_x3;ideal_y3];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X, 'econ');
sig = diag(S);
lambda = diag(S).^2;
Y = U'*X;

figure()
plot(1:6, lambda/sum(lambda), 'ko', 'Linewidth', 2, 'MarkerFaceColor'
    ,[0.5,0.5,0.5]);
title("Ideal Case: PCA Energies")
xlabel("Dimension"); ylabel("Energy");
%saveas(gcf,'Ideal_pca.png')

figure()
subplot(2,1,1)
plot(1:226, X(2,:),1:226, X(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Frames");
title("Ideal Case: Displacement in Z and XY Directions");
legend("Z", "XY")
subplot(2,1,2)
plot(1:226, Y(1,:),'r','Linewidth', 2)
ylabel("Displacement"); xlabel("Frames");
title("Ideal Case: Displacement in Principal Component(s) Directions");
legend("PC1")
%saveas(gcf,'Ideal_displacement.png')

%% Noisy Case

load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')

%% Collect position data
noisy_1 = size(vidFrames1_2, 4);
noisy_x1 = zeros(1,noisy_1);
```

```
noisy_y1 = zeros(1,noisy_1);

for j = 1:noisy_1
    V = vidFrames1_2(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,noisy_x1(j)] = max(mean(max(V,[],1),3));
    [~,noisy_y1(j)] = max(mean(max(V,[],2),3));
end

noisy_2 = size(vidFrames2_2, 4);
noisy_x2 = zeros(1,noisy_2);
noisy_y2 = zeros(1,noisy_2);

for j = 1:noisy_2
    V = vidFrames2_2(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,noisy_x2(j)] = max(mean(max(V,[],1),3));
    [~,noisy_y2(j)] = max(mean(max(V,[],2),3));
end

noisy_3 = size(vidFrames3_2, 4);
noisy_x3 = zeros(1,noisy_3);
noisy_y3 = zeros(1,noisy_3);

for j = 1:noisy_3
    V = vidFrames3_2(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,noisy_x3(j)] = max(mean(max(V,[],1),3));
    [~,noisy_y3(j)] = max(mean(max(V,[],2),3));
end

%% Adjust video lengths
noisy_x2 = noisy_x2(20:333);
noisy_y2 = noisy_y2(20:333);
noisy_x3 = noisy_x3(1:314);
noisy_y3 = noisy_y3(1:314);

%% Perform PCA
X = [noisy_x1;noisy_y1;noisy_x2;noisy_y2;noisy_x3;noisy_y3];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X, 'econ');
sig = diag(S);
```

```
lambda = diag(S).^2;
Y = U'*X;

figure()
plot(1:6, lambda/sum(lambda), 'ko', 'Linewidth', 2, 'MarkerFaceColor'
    ,[0.5,0.5,0.5]);
title("Noisy Case: PCA Energies")
xlabel("Dimension"); ylabel("Energy");
%saveas(gcf,'Noisy_pca.png')

figure()
subplot(2,1,1)
plot(1:314, X(2,:),1:314, X(1,:), 'Linewidth', 2)
ylabel("Displacement"); xlabel("Frames");
title("Noisy Case: Displacement in Z and XY Directions");
legend("Z", "XY")
subplot(2,1,2)
plot(1:314, Y(1,:),1:314, Y(2,:),1:314, Y(3,:),'r','Linewidth', 2)
ylabel("Displacement"); xlabel("Frames");
title("Noisy Case: Displacement in Principal Component(s) Directions");
legend("PC1", "PC2", "PC3")
%saveas(gcf,'Noisy_displacement.png')

%% Horizontal Displacement
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')

%% Collect position data
hdisp_1 = size(vidFrames1_3, 4);
hdisp_x1 = zeros(1,hdisp_1);
hdisp_y1 = zeros(1,hdisp_1);

for j = 1:hdisp_1
    V = vidFrames1_3(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,hdisp_x1(j)] = max(mean(max(V,[],1),3));
    [~,hdisp_y1(j)] = max(mean(max(V,[],2),3));
end

hdisp_2 = size(vidFrames2_3, 4);
hdisp_x2 = zeros(1,hdisp_2);
hdisp_y2 = zeros(1,hdisp_2);

for j = 1:hdisp_2
    V = vidFrames2_3(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,hdisp_x2(j)] = max(mean(max(V,[],1),3));
```

```
    [~,hdisp_y2(j)] = max(mean(max(V,[],2),3));
end

hdisp_3 = size(vidFrames3_3, 4);
hdisp_x3 = zeros(1,hdisp_3);
hdisp_y3 = zeros(1,hdisp_3);

for j = 1:hdisp_3
    V = vidFrames3_3(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,hdisp_x3(j)] = max(mean(max(V,[],1),3));
    [~,hdisp_y3(j)] = max(mean(max(V,[],2),3));
end

%% Adjust video lengths
hdisp_x1 = hdisp_x1(3:239);
hdisp_y1 = hdisp_y1(3:239);
hdisp_x2 = hdisp_x2(35:271);
hdisp_y2 = hdisp_y2(35:271);

%% Perform PCA
X = [hdisp_x1;hdisp_y1;hdisp_x2;hdisp_y2;hdisp_x3;hdisp_y3];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X, 'econ');
sig = diag(S);
lambda = diag(S).^2;
Y = U'*X;

figure()
plot(1:6, lambda/sum(lambda), 'ko', 'Linewidth', 2, 'MarkerFaceColor'
    ,[0.5,0.5,0.5]);
title("Horizontal Displacement: PCA Energies")
xlabel("Dimension"); ylabel("Energy");
%saveas(gcf,'hd_pca.png')

figure()
subplot(2,1,1)
plot(1:237, X(2,:),1:237, X(1,:), 'Linewidth', 2)
ylabel("Displacement"); xlabel("Frames");
title("Horizontal Displacement: Displacement in Z and XY Directions");
legend("Z", "XY")
subplot(2,1,2)
plot(1:237, Y(1,:),1:237, Y(2,:),1:237, Y(3,:),'r','Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Frames)");
title("Horizontal Displacement: Displacement in Principal Component(s)
    Directions");
legend("PC1", "PC2", "PC3")
%saveas(gcf,'hd_displacement.png')
```

```
%% Horizontal Displacement with Rotation
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')

%% Collect position data
rotate_1 = size(vidFrames1_4, 4);
rotate_x1 = zeros(1,rotate_1);
rotate_y1 = zeros(1,rotate_1);

for j = 1:rotate_1
    V = vidFrames1_4(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,rotate_x1(j)] = max(mean(max(V,[],1),3));
    [~,rotate_y1(j)] = max(mean(max(V,[],2),3));
end

rotate_2 = size(vidFrames2_4, 4);
rotate_x2 = zeros(1,rotate_2);
rotate_y2 = zeros(1,rotate_2);

for j = 1:rotate_2
    V = vidFrames2_4(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,rotate_x2(j)] = max(mean(max(V,[],1),3));
    [~,rotate_y2(j)] = max(mean(max(V,[],2),3));
end

rotate_3 = size(vidFrames3_4, 4);
rotate_x3 = zeros(1,rotate_3);
rotate_y3 = zeros(1,rotate_3);

for j = 1:rotate_3
    V = vidFrames3_4(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,rotate_x3(j)] = max(mean(max(V,[],1),3));
    [~,rotate_y3(j)] = max(mean(max(V,[],2),3));
end

%% Adjust video lengths
rotate_x2 = rotate_x2(14:405);
rotate_y2 = rotate_y2(14:405);
rotate_x3 = rotate_x3(1:392);
rotate_y3 = rotate_y3(1:392);
```

```
%% PCA
X = [rotate_x1;rotate_y1;rotate_x2;rotate_y2;rotate_x3;rotate_y3];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X, 'econ');
sig = diag(S);
lambda = diag(S).^2;
Y = U'*X;

figure()
plot(1:6, lambda/sum(lambda), 'ko', 'Linewidth', 2, 'MarkerFaceColor'
    ,[0.5,0.5,0.5]);
title("Horizontal Displacement w/ Rotation: PCA Energies")
xlabel("Dimension"); ylabel("Energy");
%saveas(gcf,'hdrotate_pca.png')

figure()
subplot(2,1,1)
plot(1:392, X(2,:),1:392, X(1,:), 'Linewidth', 2)
ylabel("Displacement"); xlabel("Frames");
title("Horizontal Displacement w/ Rotation: Displacement in Z and XY
    Directions");
legend("Z", "XY")
subplot(2,1,2)
plot(1:392, Y(1,:),1:392, Y(2,:), 1:392, Y(3,:),'r','Linewidth', 2)
ylabel("Displacement"); xlabel("Frames");
title("Horizontal Displacement w/ Rotation: Displacement in Principal
    Component(s) Directions");
legend("PC1", "PC2", "PC3", "PC4")
%saveas(gcf,'hdrotate_displacement.png')
```