AMATH 482 Homework 3

Jonah Wu

February 24, 2021

Abstract

We will analyze various videos of a spring-mass system with different cases to acquire position data over time from different angles of cameras. This experiment allows us to explore various aspects and practical usefulness of the Principal Component Analysis (PCA), and the effects of noise on the PCA algorithms.

1 Introduction and Overview

We are going to analyze 4 different cases of a spring-mass system with 3 different cameras for each case. The 4 cases include:

- Ideal Case: the entire motion is in the z direction with simple harmonic motion being observed
- Noisy Case: camera shakes are introduced in video recordings for the ideal case
- Horizontal Displacement: mass is released off-centre so as to produce motion in the x-y plane as well as the z direction
- Horizontal Displacement and Rotation: mass is released off-centre and rotates so as to produce rotation in the x-y plane and motion in the z direction

2 Theoretical Background

2.1 Singular Value Decomposition (SVD)

The SVD is a form of factoring a real or complex matrix that generalizes the eigen decomposition of a square normal matrix to any $m \times n$ matrix. Singular value decomposition of the matrix $\mathbf{A}$:

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

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal.
The values σ_n on the diagonal of Σ are called the singular values of the matrix $\mathbf{A}$. The number of non-zero singular values is equal to the rank of $\mathbf{A}$. The vectors u_n which make up the columns of $\mathbf{U}$ are called the left singular vectors of $\mathbf{A}$. The vectors v_n which make up the columns of $\mathbf{V}$ are called the right singular vectors of $\mathbf{A}$.

2.2 Principal Component Analysis (PCA)

Principal component analysis is a process of using the principal components it computes and uses them to perform a change of basis on the data. We determine if we use only the first few principal components and ignoring the rest by looking at the energy of each dimension:

$$\text{energy}_i = \frac{{\sigma_i}^2}{\sum {\sigma_i}^2} \tag{2}$$

From this, the energy associated with each singular value can be determined. For the purposes of PCA, this determines how much information is stored in each of the principal components. Finally, the principal component projection can be obtained from $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$.

3 Algorithm Implementation and Development

After we have the coordinates for the positional data as $\mathbf{X}$, we can apply Principal Component Analysis to examine the motion of the spring-mass system. We can find the principal component projection with $\mathbf{Y} = \mathbf{U^*X}$, and we also compute the energies of each dimension with the algorithm below.

Algorithm 1: Dynamic Mode Decomposition (DMD)

SVD on `X` to get $\mathbf{U\Sigma V^*}$ using `svd()`
Calculate $\mathbf{Y = U^*X}$
Calculate eigenvalues by squaring the diagonals of Σ
Calculate energies using $\frac{\lambda_i}{\sum \lambda_n}$

4 Computational Results

4.1 Ideal Case

We found out that for ideal case, we only have 1 principal component since it is rank 1 with 1 dimension of energy greater than 0.9.

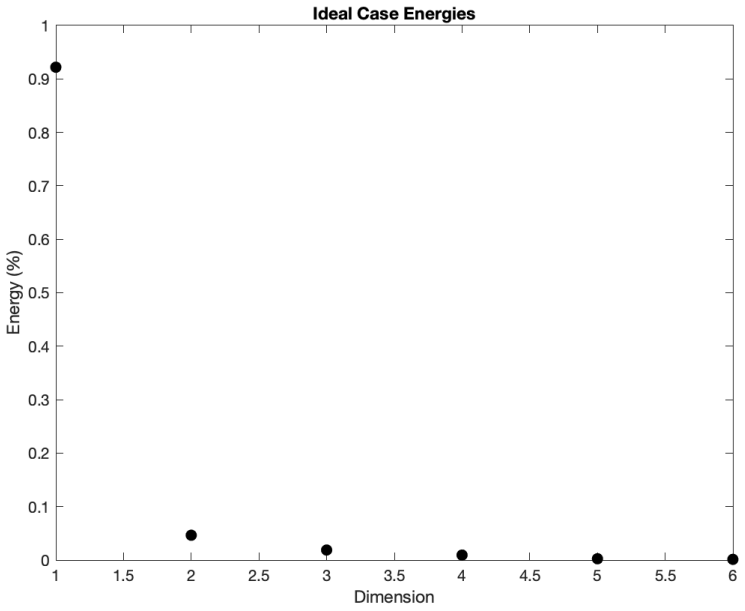

Figure 1: Ideal Case Energies

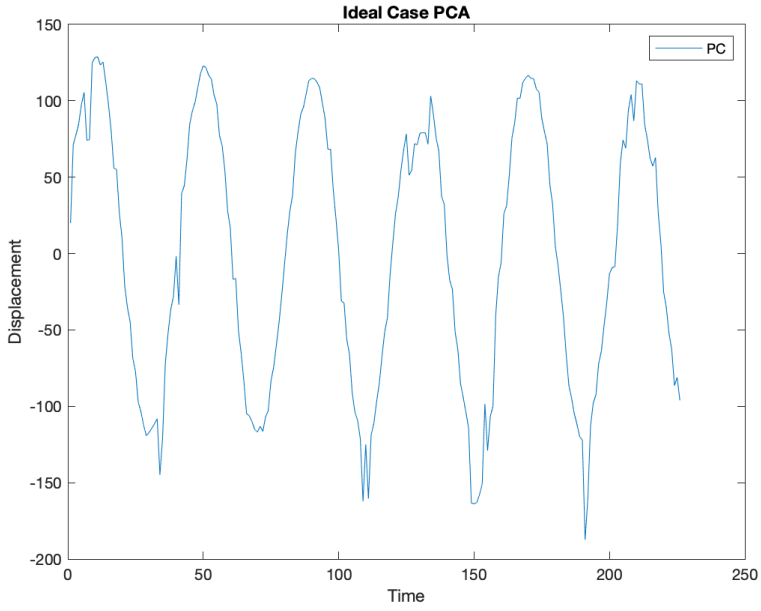

Figure 2: Ideal Case Principal Components

4.2 Noisy Case

We found out that for noisy case, we have 3 principal components since the energy for the first 3 dimension are all greater than 0.1.

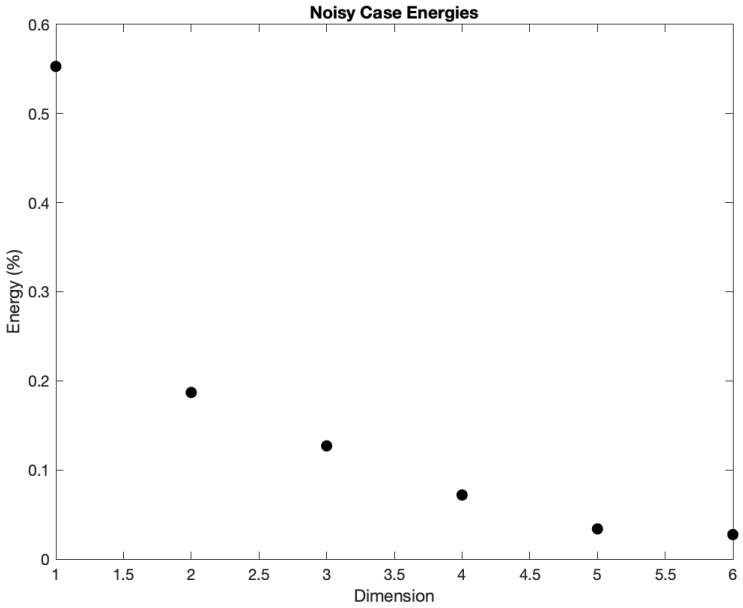

Figure 3: Noisy Case Energies

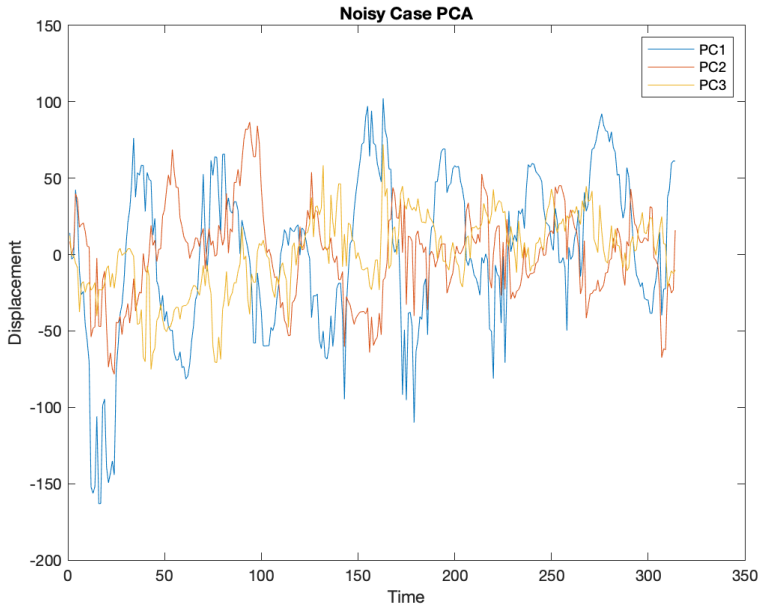

Figure 4: Noisy Case Principal Components

4.3 Horizontal Displacement

We found out that for the Horizontal Displacement case, we have 3 principal components since the energy for the first 3 dimension are all greater than 0.1.

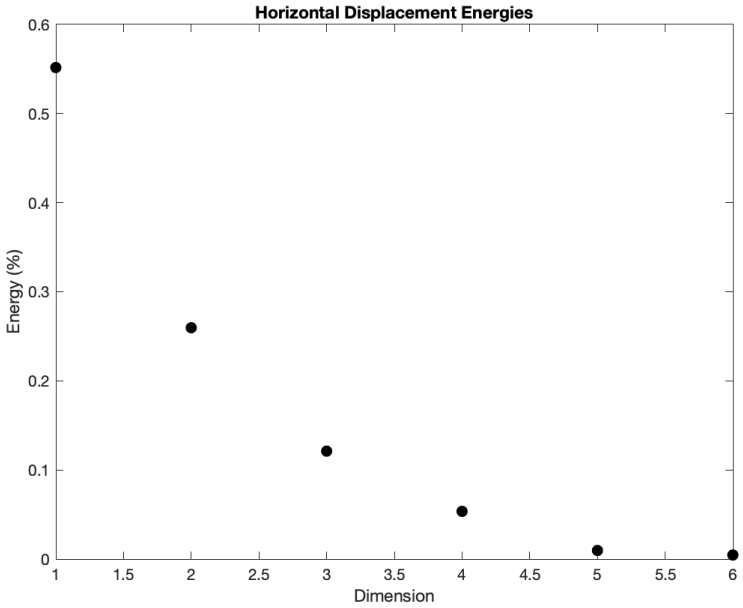

Figure 5: Horizontal Displacement Energies

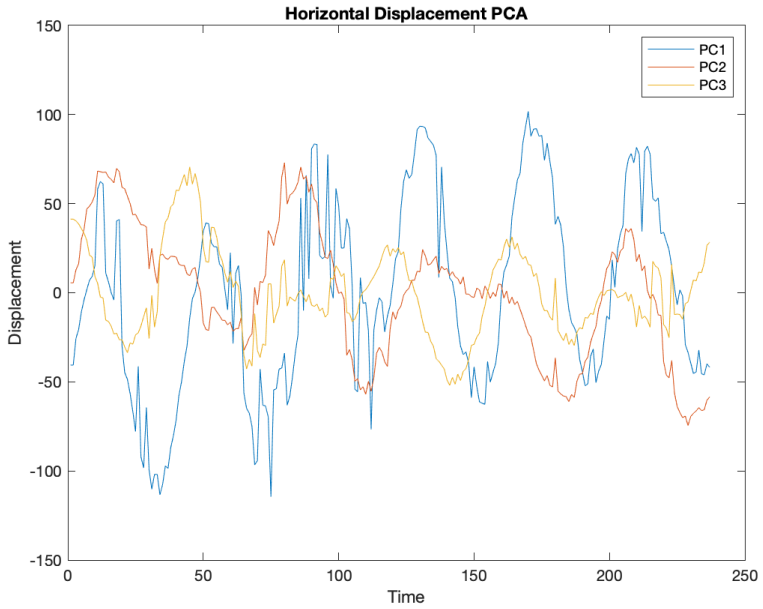

Figure 6: Horizontal Displacement Principal Components

4.4 Horizontal Displacement and Rotation

We found out that for the Horizontal Displacement and Rotation case, we have 3 principal components since the energy for the first 3 dimension are all greater than 0.1.

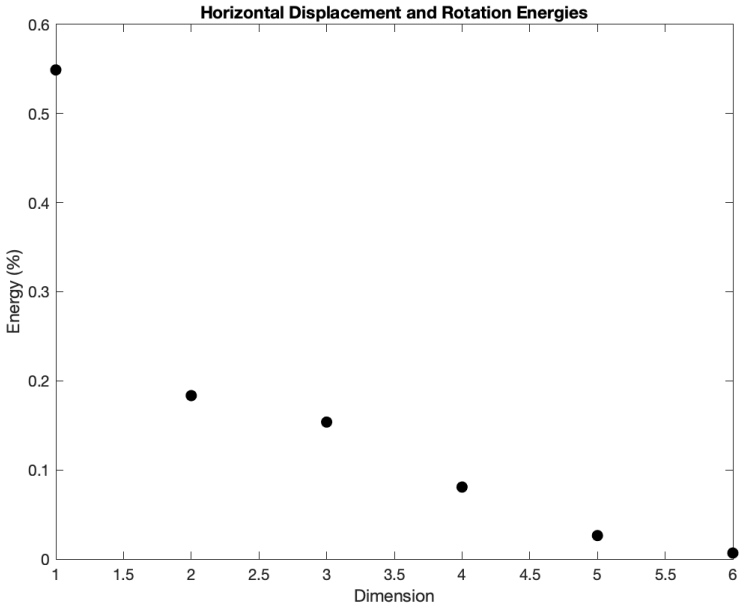

Figure 7: Horizontal Displacement and Rotation Energies

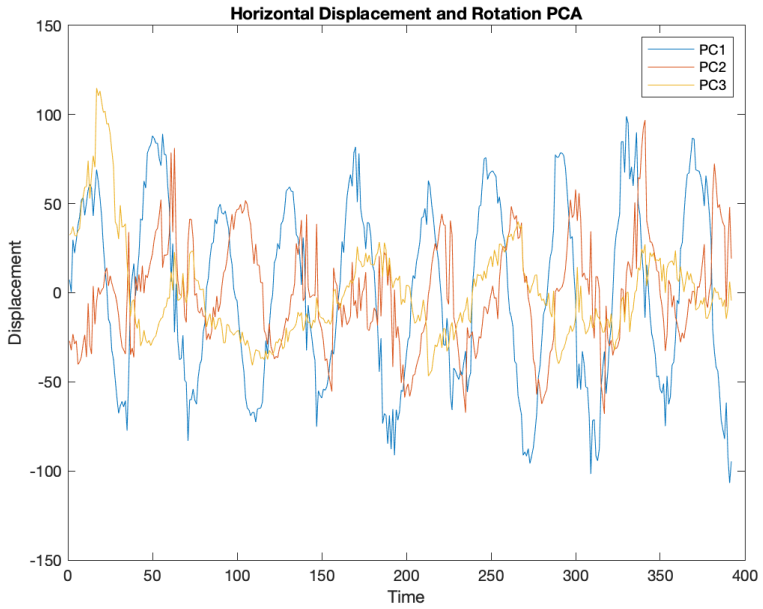

Figure 8: Horizontal Displacement and Rotation Principal Components

5 Summary and Conclusions

PCA is a powerful tool that provides insights that lead to dimension reduction of data. This was useful in our situation of recording the motion of the object in a spring-mass system. Through this assignment, we were able to see that we can interpret the movement in the ideal case in 1 dimension, and for the other 3 cases, we need data from 3 cameras to fully help us interpret the trajectory of the spring-mass system. Overall, with PCA, we do not really need a comprehensive knowledge in physics in order to analyze motions in a spring-mass system.

Appendix A MATLAB Functions

- `size(A)` returns a row vector whose elements are the lengths of the corresponding dimensions of A.
- `[U, S, V] = svd(X)`: performs a singular value decomposition of matrix A, such that A = U*S*V'.
- `B = repmat(A,n)` returns an array containing n copies of A in the row and column dimensions. The size of B is size(A) $\times$ n when A is a matrix.

Appendix B MATLAB Code

```
clear; close all; clc;

%% Ideal Case

load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')

% 1_1
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

% 2_1
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

% 3_1
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

% pca
X = [ideal_x1;ideal_y1;
     ideal_x2(10:235);ideal_y2(10:235);
     ideal_x3(1:226);ideal_y3(1:226)];
```

```
[¬,n] = size(X);
X = X - repmat(mean(X,2),1,n);
[U,S,V] = svd(X, 'econ');
Y = U'*X;
sigs = diag(S).^2;
energy = sigs/sum(sigs);

figure()
plot(1:6, energy, 'k.', 'MarkerSize',20);
title("Ideal Case Energies");
xlabel("Dimension");
ylabel("Energy (%)");
saveas(gcf,'ideal_energy.png');

figure()
plot(1:226, Y(1,:));
title("Ideal Case PCA");
xlabel("Time");
ylabel("Displacement");
legend("PC");
saveas(gcf,'ideal.png');

%% Noisy Case

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

% 1_2
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

% 2_2
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

% 3_2
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

% PCA
X = [noisy_x1;noisy_y1;
     noisy_x2(20:333);noisy_y2(20:333);
     noisy_x3(1:314);noisy_y3(1:314)];
[¬,n] = size(X);
X = X - repmat(mean(X,2),1,n);
[U,S,V] = svd(X, 'econ');
Y = U'*X;
sigs = diag(S).^2;
energy = sigs/sum(sigs);

figure()
plot(1:6, energy, 'k.', 'MarkerSize',20);
title("Noisy Case Energies");
xlabel("Dimension");
ylabel("Energy (%)");
saveas(gcf,'noisy_energy.png');

figure()
plot(1:314, Y(1,:),1:314, Y(2,:), 1:314, Y(3,:));
title("Noisy Case PCA");
xlabel("Time");
ylabel("Displacement");
legend("PC1", "PC2", "PC3");
saveas(gcf,'noisy.png');


%% Horizontal Displacement

load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')

% 1_3
horizontal_1 = size(vidFrames1_3, 4);
horizontal_x1 = zeros(1,horizontal_1);
horizontal_y1 = zeros(1,horizontal_1);
for j = 1:horizontal_1
    V = vidFrames1_3(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;
    [¬,horizontal_x1(j)] = max(mean(max(V,[],1),3));
    [¬,horizontal_y1(j)] = max(mean(max(V,[],2),3));
end

% 2_3
horizontal_2 = size(vidFrames2_3, 4);
horizontal_x2 = zeros(1,horizontal_2);
horizontal_y2 = zeros(1,horizontal_2);
for j = 1:horizontal_2
    V = vidFrames2_3(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;
    [¬,horizontal_x2(j)] = max(mean(max(V,[],1),3));
    [¬,horizontal_y2(j)] = max(mean(max(V,[],2),3));
end

% 3_3
horizontal_3 = size(vidFrames3_3, 4);
horizontal_x3 = zeros(1,horizontal_3);
horizontal_y3 = zeros(1,horizontal_3);
for j = 1:horizontal_3
    V = vidFrames3_3(:,:,:,j);
    V(1:180,:,:) = 0;
```

```
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;
    [¬,horizontal_x3(j)] = max(mean(max(V,[],1),3));
    [¬,horizontal_y3(j)] = max(mean(max(V,[],2),3));
end

% pca
X = [horizontal_x1(1:237);horizontal_y1(1:237);
     horizontal_x2(35:271);horizontal_y2(35:271);
     horizontal_x3;horizontal_y3];
[¬,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);
[U,S,V] = svd(X, 'econ');
Y = U'*X;
sigs = diag(S).^2;
energy = sigs/sum(sigs);

figure()
plot(1:6, energy, 'k.', 'MarkerSize',20);
title("Horizontal Displacement Energies")
xlabel("Dimension");
ylabel("Energy (%)");
saveas(gcf,'horizontal_energy.png')

figure()
plot(1:237, Y(1,:),1:237, Y(2,:), 1:237, Y(3,:))
title("Horizontal Displacement PCA");
xlabel("Time");
ylabel("Displacement");
legend("PC1", "PC2", "PC3")
saveas(gcf,'horizontal.png')


%% Horizontal Displacement and Rotation

load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')

% 1_4
rotatation_1 = size(vidFrames1_4, 4);
rotatation_x1 = zeros(1,rotatation_1);
rotatation_y1 = zeros(1,rotatation_1);
for j = 1:rotatation_1
    V = vidFrames1_4(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;
    [¬,rotatation_x1(j)] = max(mean(max(V,[],1),3));
    [¬,rotatation_y1(j)] = max(mean(max(V,[],2),3));
end

% 2_4
rotatation_2 = size(vidFrames2_4, 4);
rotatation_x2 = zeros(1,rotatation_2);
rotatation_y2 = zeros(1,rotatation_2);
for j = 1:rotatation_2
    V = vidFrames2_4(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;
    [¬,rotatation_x2(j)] = max(mean(max(V,[],1),3));
    [¬,rotatation_y2(j)] = max(mean(max(V,[],2),3));
end

% 3_4
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

% pca
X = [rotatation_x1;rotatation_y1;
     rotatation_x2(14:405);rotatation_y2(14:405);
      rotate_x3(1:392);rotate_y3(1:392)];
[¬,n] = size(X);
X = X - repmat(mean(X,2),1,n);
[U,S,V] = svd(X, 'econ');
Y = U'*X;
sigs = diag(S).^2;
energy = sigs/sum(sigs);

figure()
plot(1:6, energy, 'k.', 'MarkerSize',20);
title("Horizontal Displacement and Rotation Energies")
xlabel("Dimension");
ylabel("Energy (%)");
saveas(gcf,'rotation_energy.png')

figure()
plot(1:392, Y(1,:),1:392, Y(2,:),1:392, Y(3,:))
title("Horizontal Displacement and Rotation PCA");
xlabel("Time");
ylabel("Displacement");
legend("PC1", "PC2", "PC3")
saveas(gcf,'rotation.png')
```