Assignment 3

PCA and a Spring-Mass System

Author: Sofia Chen

Abstract: This assignment is to explore the Principal Component Analysis (PCA) method, by applying them to analyze 4 different cases of a spring-mass system.

Section I. Introduction and Overview

Consider an experiment with a spring-mass system. That is, a mass is hanging from a spring and the mass moves in some directions. Then, we have the following 4 cases:

1. **Test 1: Ideal Case.** Consider a small displacement of the mass in the z direction and the ensuing oscillations. In this case, the entire motion is in the z direction with simple harmonic motion being observed.
2. **Test 2: Noisy Case**. Repeat the ideal case experiment, but this time introduce camera shakes into the video recording.
3. **Test 3: Horizontal Displacement.** In this case the mass is released off-centre so as to produce motion in the x − y plane as well as the z direction.
4. **Test 4: Horizontal Displacement and Rotation.** In this case the mass is released off-centre and rotates so as to produce rotation in the x − y plane and motion in the z direction.

In order to understand the motion of the mass-spring system, we can set up three cameras around the system and collect video. We will first extract the mass positions from the video frames. Then we can use the Principal Component Analysis algorithms to extract the dynamics of each scenario. We will compare and contrast the different cases.

Section II. Theoretical Background

let A be an $m \times n$ matrix, representing a transformation from R^n to R^m. The image of the n-dimensional unit sphere under A is a hyperellipse in m-dimensional space. In the case that $m \geq n$ and A is full rank, the hyperellipse will be n-dimensional.

There are now n-principal semiaxes with lengths $\sigma_1, \sigma_2, \ldots, \sigma_n$ that point in the directions $u_1,\ u_2, \ldots, u_n$. Again there are vectors $v_1, v_2, \ldots, v_n$ such that $Av_1, = \sigma_1 u_1, Av_2 = \sigma_2\, u_2, \ldots, Av_n = \sigma_n\, u_n$. We can again write this compactly as $AV = U\hat{}\ \Sigma\hat{}$, where the matrices have the same meaning as they did in the 2 × 2 case. Again using the fact that the columns of V form an orthonormal basis, we can isolate for A to get $A\ U\hat{}\ \Sigma\hat{} V^x$. The above decomposition of the matrix A is called the reduced SVD. Notice that U^ is an $m \times n$ matrix with orthonormal columns.

What is typically done is to construct a matrix U from $U\hat{}$ by adding an addition $m - n$ columns that are orthonormal to the already existing set $U\hat{}$. In this way, the matrix U becomes a square $m \times m$ matrix, and in order to make the decomposition work, and additional $m - n$ rows of zeros are also added to the matrix $\Sigma\hat{}$, resulting in the matrix Σ. This leads to the *singular value decomposition* of the matrix A: $\boldsymbol{A} = \boldsymbol{U\Sigma V^*}$, where $U \in R^{m\times m}$ and $V \in R^{n\times n}$ are unitary matrices, and $\Sigma \in R^{m\times m}$ is diagonal.

The values σ_n on the diagonal of Σ are called the *singular values* of the matrix A. The vectors u_n which make up the columns of U are called the *left singular vectors* of A. The vectors v_n which make up the columns of V are called the *right singular vectors* of A.

Then we have the following theorem: If A is a matrix of rank r, then A is the sum of r rank 1 matrices: $A = \sum_{j=1}^{r} \sigma_j u_j v_j^*$.

So SVD can be used to produce low-rank approximations of a data set. This method is called *principal component analysis (PCA)*. By applying PCA to the systems that we want to analyze, we can extract the dynamics of each scenario and compare the differences of the cases.

Section III. Algorithm Implementation and Development

In order to track the displacement of the mass in the videos, I first find out the number of snapshots (frames) that each video has. And then, for each frame, I convert the image into gray scale, and then turn it into a matrix with a filter, which makes sure we only look for the mass in the maximum displacements in the x and y direction of its movement. I do this by inspecting the pixel values in the Movie Player in the Matlab.

Because there is a white light on the mass, I look for the mass by restricting the values above certain pixel values. Notice that the most common pixel format is the byte image, where this number is stored as an 8-bit integer giving a range of possible values from 0 to 255. Typically, zero is taken to be black, and 255 is taken to be white. I mostly use 245 as the lower bound to find the white point. After that, we can store x and y coordinates of the displacement of the mass (white point) as a n-by-2 matrix.

After we collect that for each of the camera, we can put them into a 6-by-n matrix. We also need to generically trim each movie clip down to the same size as well as align them in time. I do this by finding the minimum y-value of each camera data and use it as the start point.

Then I can apply the PCA to the matrix. After doing the singular value decomposition, I plot the singular values and the energy of each modes of the system. We can also find out the rank of the matrix and use it to analyze its movement and the dynamics of the system.

I also plot the displacement diagram of the masses by extracting the first two rows of the matrix. It is just the information from camera 1. Notice that when we just track the mass, the second column corresponds to the z-axis movements, while the first column corresponds to the x-y plane movements.

Section IV. Computational Results

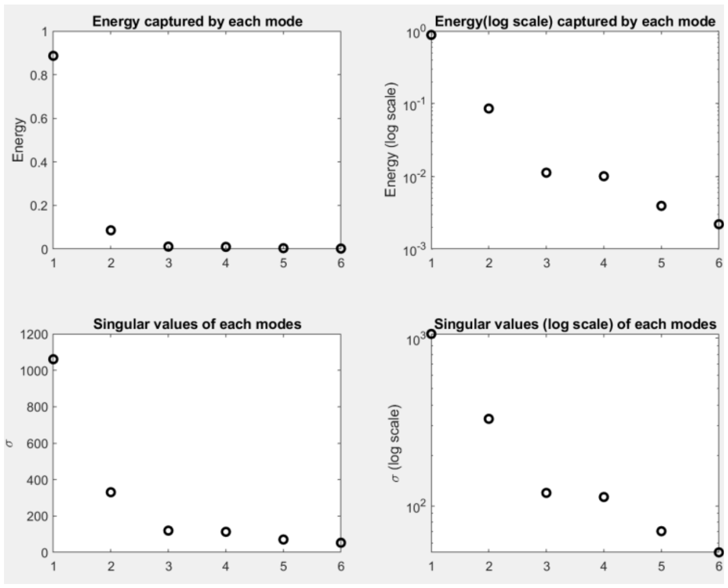

Figure 1: In Test 1, singular values and energy captured by each mode.

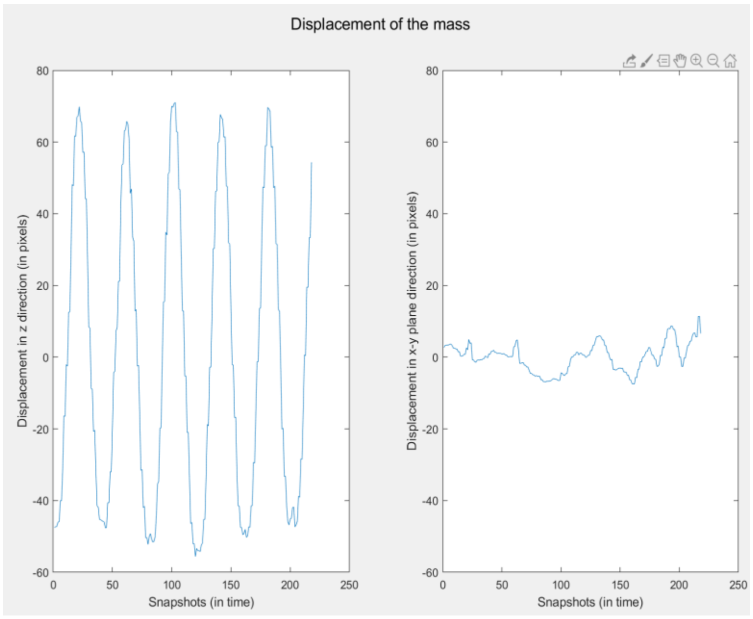

Figure 2: In Test 1, the displacements of the mass in z-direction and the x-y plane direction.

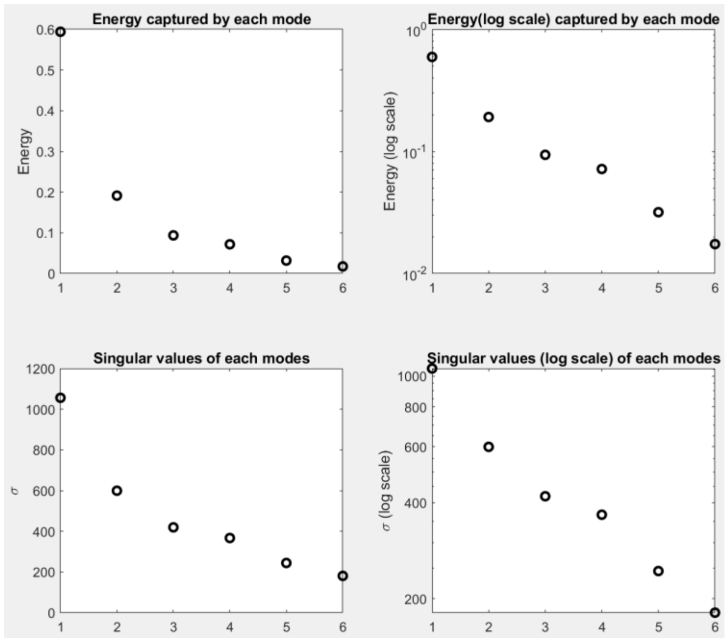

Figure 3: In Test 2, singular values and energy captured by each mode.

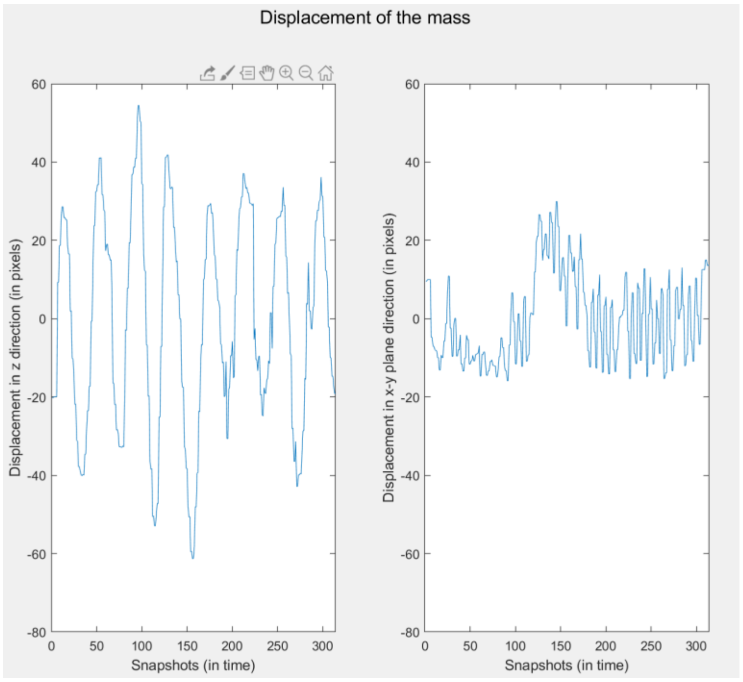

Figure 4: In Test 2, the displacements of the mass in z-direction and the x-y plane direction.

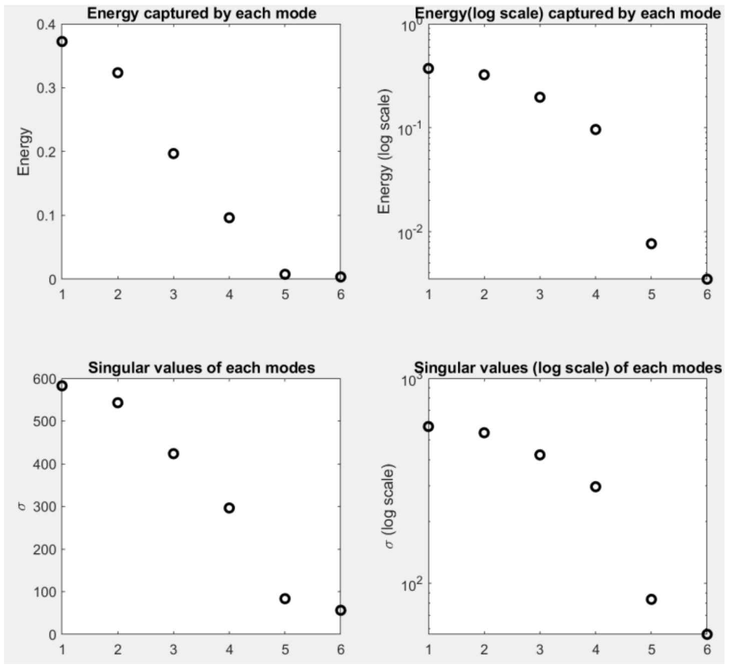

Figure 5: In Test 3, singular values and energy captured by each mode.

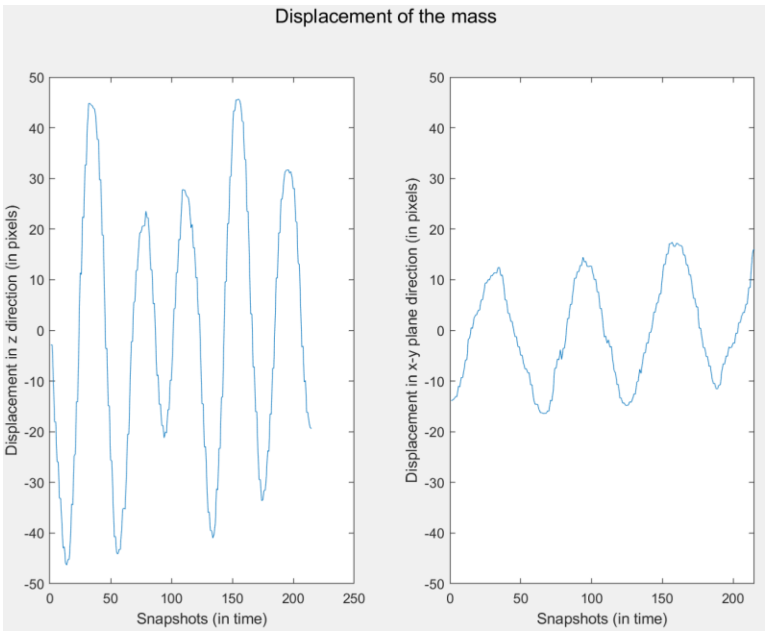

Figure 6: In Test 3, the displacements of the mass in z-direction and the x-y plane direction

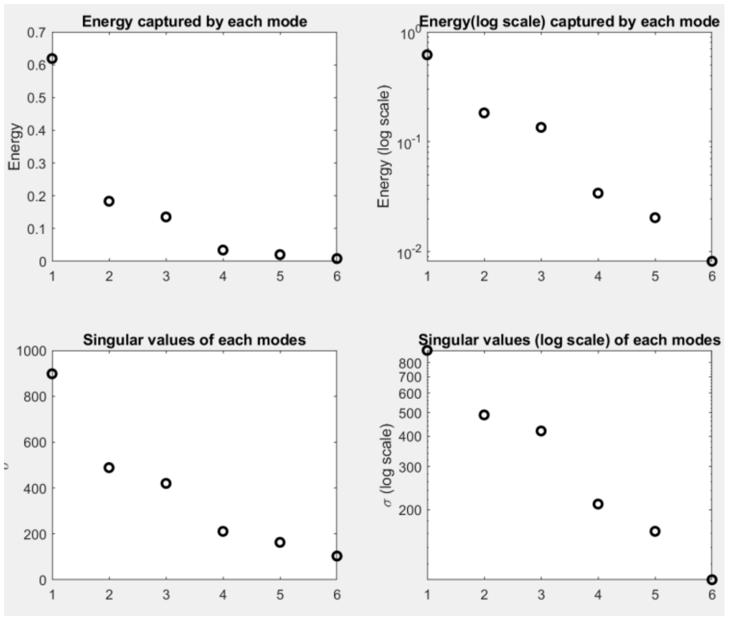

Figure 7: In Test 4, singular values and energy captured by each mode.

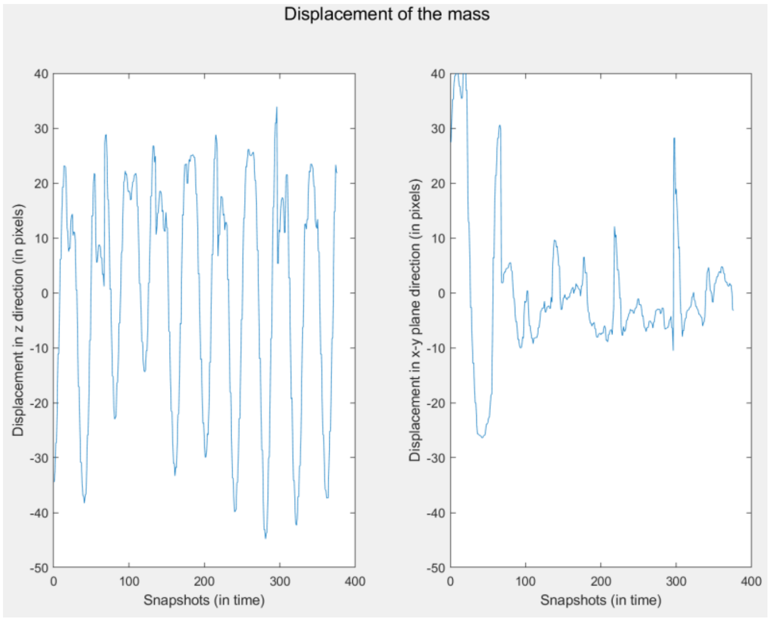

Figure 8: In Test 4, the displacements of the mass in z-direction and the x-y plane direction

Section V. Summary and Conclusions

We can compare the 4 cases from the above results. From figure 1, we can see how much larger the first singular value is to the rest of the singular values. The energy graph also tells that the energy is super greatly captured by the first mode. It represents the fact that the motion is nearly 1D. From figure 2, we can also notice that the mass only moves up and down smoothly in the z-direction, while it almost does not move in the x-y plane direction (all the points fall around 0).

For case 2, the difference between the first and second modes is not that big. That is because when the system is added some noises, the motion of mass is not that near to be 1D. From figure 4, we can see that although the mass is still mainly move in the z-direction, the displacement is not that smooth as the last case. It is also obvious that there is also a small movement in the xy-direction, which is caused by the shaking of the camera.

From figure 5, we can notice that there is not only 1 singular value that is greatly larger than others. Actually, the first two singular values are closed. Thus, the movement of the mass in this case is definitely not 1D. From figure 6, we can see that the mass also does a regular and nearly smooth movement in the xy-direction, although it does not to the extent as the displacement of z-direction. It does correspond to the horizontal movement as the introduction above.

For the last two figures, the result is a little bit weird. I think one of the reasons might be when the mass is rotating, the video cannot catch the white pixel clearly. We can see that most of the energy falls on the first modes. And for figure 8, the graph showing the movement in the xy-plane might tell us that the rotation is becoming smaller and smaller.

Appendix A. MATLAB functions used and brief implementation explanation

- ***rgb2gray*** - Convert RGB image or colormap to grayscale: This MATLAB function converts the truecolor image RGB to the grayscale image I.
- ***find*** - Find requirements in requirements set that have matching attribute values: This MATLAB function finds and returns an slreq.Requirement object myReq in the requirements set rs specified by the properties matching PropertyName and PropertyValue.
- ***svd*** - Singular value decomposition: This MATLAB function returns the singular values of matrix A in descending order.
- ***semilogy*** - Plot specified circuit object parameters using log scale for y-axis: This MATLAB function plots the specified parameter in the default format using a logarithmic scale for the y-axis.
- ***implay*** - Play movies, videos, or image sequences: This MATLAB function opens the Video Viewer app.

Appendix B. MATLAB codes

Assign3.m:

```
% Clean workspace
% clear all; close all; clc

%% Test 1: Ideal Case
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
% implay(vidFrames1_1)
disp1_1 = synchronizing(tracking(vidFrames1_1, 200, 380, 325, 345, 250));
disp2_1 = synchronizing(tracking(vidFrames2_1, 100, 300, 250, 320, 250));
disp3_1 = synchronizing(tracking(vidFrames3_1, 200, 345, 235, 500, 247));
l = length(disp1_1);
T1 = [disp1_1'; disp2_1(1:l,:)'; disp3_1(1:l,:)'];
[m,n] = size(T1);
for i = 1:6
    a = mean(T1(i,:));
    for j = 1:n
        T1(i,j) = T1(i,j) - a;
    end
end
[U,S,V] = svd(T1, 'econ');
sig = diag(S);
figure(1)
subplot(2,2,1)
```

```
plot(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy')
title('Energy captured by each mode')
subplot(2,2,2)
semilogy(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy (log scale)')
title('Energy(log scale) captured by each mode')
subplot(2,2,3)
plot(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma')
title('Singular values of each modes')
subplot(2,2,4)
semilogy(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma (log scale)')
title('Singular values (log scale) of each modes')
figure(2)
subplot(1,2,1)
plot(1:l, T1(2,:))
xL = get(gca, 'XLim');
yL = get(gca, 'YLim');
ylabel('Displacement in z direction (in pixels)')
xlabel('Snapshots (in time)')
subplot(1,2,2)
plot(1:l, T1(1,:))
ylabel('Displacement in x-y plane direction (in pixels)')
xlabel('Snapshots (in time)')
set(gca, 'XLim', xL);
set(gca, 'YLim', yL);
suptitle('Displacement of the mass')
%% Test 2: Noisy Case
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
disp1_2 = synchronizing(tracking(vidFrames1_2, 200, 400, 330, 400, 240));
disp2_2 = synchronizing(tracking(vidFrames2_2, 140, 400, 200, 400, 247));
disp3_2 = synchronizing(tracking(vidFrames3_2, 210, 280, 250, 450, 245));
l = length(disp1_2);
T2 = [disp1_2'; disp2_2(1:l,:)'; disp3_2(1:l,:)'];
[m,n] = size(T2);
for i = 1:6
    a = mean(T2(i,:));
    for j = 1:n
        T2(i,j) = T2(i,j) - a;
    end
end
[U,S,V] = svd(T2, 'econ');
sig = diag(S);
figure(3)
subplot(2,2,1)
plot(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy')
title('Energy captured by each mode')
subplot(2,2,2)
semilogy(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy (log scale)')
title('Energy(log scale) captured by each mode')
subplot(2,2,3)
```

```matlab
plot(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma')
title('Singular values of each modes')
subplot(2,2,4)
semilogy(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma (log scale)')
title('Singular values (log scale) of each modes')
figure(4)
subplot(1,2,1)
plot(1:l, T2(2,:))
xL = get(gca, 'XLim');
yL = get(gca, 'YLim');
ylabel('Displacement in z direction (in pixels)')
xlabel('Snapshots (in time)')
subplot(1,2,2)
plot(1:l, T2(1,:))
ylabel('Displacement in x-y plane direction (in pixels)')
xlabel('Snapshots (in time)')
set(gca, 'XLim', xL);
set(gca, 'YLim', yL);
suptitle('Displacement of the mass')
%% Test 3: Horizontal Displacement
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
disp1_3 = synchronizing(tracking(vidFrames1_3, 225, 450, 270, 450, 250));
disp2_3 = synchronizing(tracking(vidFrames2_3, 100, 450, 160, 425, 250));
disp3_3 = synchronizing(tracking(vidFrames3_3, 160, 365, 200, 500, 245));
l = length(disp1_3);
T3 = [disp1_3'; disp2_3(1:l,:)'; disp3_3(1:l,:)'];
[m,n] = size(T3);
for i = 1:6
    a = mean(T3(i,:));
    for j = 1:n
        T3(i,j) = T3(i,j) - a;
    end
end
[U,S,V] = svd(T3, 'econ');
sig = diag(S);
figure(5)
subplot(2,2,1)
plot(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy')
title('Energy captured by each mode')
subplot(2,2,2)
semilogy(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy (log scale)')
title('Energy(log scale) captured by each mode')
subplot(2,2,3)
plot(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma')
title('Singular values of each modes')
subplot(2,2,4)
semilogy(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma (log scale)')
title('Singular values (log scale) of each modes')
figure(6)
```

```
subplot(1,2,1)
plot(1:l, T3(2,:))
xL = get(gca, 'XLim');
yL = get(gca, 'YLim');
ylabel('Displacement in z direction (in pixels)')
xlabel('Snapshots (in time)')
subplot(1,2,2)
plot(1:l, T3(1,:))
ylabel('Displacement in x-y plane direction (in pixels)')
xlabel('Snapshots (in time)')
set(gca, 'XLim', xL);
set(gca, 'YLim', yL);
suptitle('Displacement of the mass')
%% Test 4: Horizontal Displacement and Rotation
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
disp1_4 = synchronizing(tracking(vidFrames1_4, 230, 360, 300, 480, 235));
disp2_4 = synchronizing(tracking(vidFrames2_4, 95, 295, 230, 420, 245));
disp3_4 = synchronizing(tracking(vidFrames3_4, 150, 275, 300, 470, 235));
l = length(disp3_4);
T4 = [disp1_4(1:l,:)'; disp2_4(1:l,:)'; disp3_4'];
[m,n] = size(T4);
for i = 1:6
    a = mean(T4(i,:));
    for j = 1:n
        T4(i,j) = T4(i,j) - a;
    end
end
[U,S,V] = svd(T4, 'econ');
sig = diag(S);
figure(7)
subplot(2,2,1)
plot(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy')
title('Energy captured by each mode')
subplot(2,2,2)
semilogy(sig.^2/sum(sig.^2), 'ko', 'Linewidth', 2)
ylabel('Energy (log scale)')
title('Energy(log scale) captured by each mode')
subplot(2,2,3)
plot(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma')
title('Singular values of each modes')
subplot(2,2,4)
semilogy(sig, 'ko', 'Linewidth', 2)
ylabel('\sigma (log scale)')
title('Singular values (log scale) of each modes')
figure(8)
subplot(1,2,1)
plot(1:l, T4(2,:))
xL = get(gca, 'XLim');
yL = get(gca, 'YLim');
ylabel('Displacement in z direction (in pixels)')
xlabel('Snapshots (in time)')
subplot(1,2,2)
plot(1:l, T4(1,:))
```

```
ylabel('Displacement in x-y plane direction (in pixels)')
xlabel('Snapshots (in time)')
set(gca, 'XLim', xL);
set(gca, 'YLim', yL);
suptitle('Displacement of the mass')
```

tracking.m:

```
function disp = tracking(vidFr, ymin, ymax, xmin, xmax, wp)
    snapshots = size(vidFr, 4);
    disp = [];
    filter = zeros(480, 640);
    filter(ymin:1:ymax, xmin:1:xmax) = 1;
    for i = 1:snapshots
        A = double(rgb2gray(vidFr(:,:,:,i))).*filter;
        [Y, X] = find(A>wp);
        disp = [disp; mean(X), mean(Y)];
    end
end
```

synchronizing.m

```
function syn_vidFr = synchronizing(vidFr)
    [B, A] = min(vidFr(1:25,2));
    syn_vidFr = vidFr(A:end,:);
end
```