AMATH 482 Assignment 3

Carter Peyton

February 24, 2021

Abstract

This assignment aims to analyze video clips of a spring-mass system in order to understand the dynamics of that system. Multiple different scenarios of the spring mass system with varying dynamics will be analyzed. Through image analysis and the singular value decomposition, the dynamics of each case can be extracted from the respective video clips.

1 Introduction and Overview

Consider four cases of a spring-mass system. A displacement solely in the Z direction, displacement in the Z direction with noise, horizontal and vertical (Z) displacement, and horizontal displacement with rotation. For each test case there are three videos corresponding to three different angles of the spring-mass system. Using image analysis, we isolate the mass for each angle. Then the SVD will be applied to a matrix that contains this data. Analyzing the SVD of this matrix, will allow us to plot the general motion of the system and understand its dynamics for each case.

2 Theoretical Background

2.1 The Singular Value Decomposition

The singular value decomposition (SVD) is, as the name suggests, a method to decompose a matrix A into unitary matrices U, V^* and diagonal matrix Σ, so that $A = U\Sigma V^*$. The diagonal elements of Σ, which are non negative and in descending order, are called the singular values (σ_j). U and V^* are rotation matrices that apply a change of basis to the original data. The columns of these matrices correspond to the respective singular values, which in turn give an idea of the information capture of these columns in their respective bases. Note that the mean of each row in A must be subtracted from the row.

2.1.1 Computing Singular Value Energies

The energy of each singular component (energy is analagous to the information of the system) can be computed with the formula

$$E = \frac{\sigma_1}{\sum_{i=1}^{i=n} \sigma_i} \tag{1}$$

2.2 Principal Component Analysis (PCA)

Given some data, principal component analysis aims to find a new set of coordinates (a change of basis) so that the variables become uncorrelated. Therefore, each variable contains unique information and eliminates redundancies. It does so by diagonalizing a covariance matrix C_x where large diagonal values correspond to large variance and large non-diagonal elements correspond to large redundancies. Our goal is to reduce redundancies and find the maximal variances, therefore we want a diagonal matrix with descending values on the diagonal and non-diagonal elements set to 0. The SVD is great at computing this, and if we let $Y = U * X$ where X is the original data and U comes from the SVD then $C_y = \Sigma^2$. Since the non-diagonal

elements in Σ are 0, we have effectively eliminated redundancy in X via computing the SVD of it, and found the values that correspond to the largest changes in the data. Remember these σ values correspond to changes in the U and V^* bases, so we call the columns of U and V^* the principal components.

3 Algorithm Implementation and Development

3.1 Initialization

The three videos for each test case are loaded into MATLAB where the 4D matrices (x, y, rgb values, time) are converted into doubles and then reduced into 3D matrices (x, y, time) by converting from RGB to greyscale. In some of the test cases, the videos from different angles are of varying time lengths. When initializing the time lengths of each case, we use the shortest video length from the three angles (i.e cutting off the last frames of the other angles).

3.2 Image Processing

In order to track the movement of the mass we notice that there is a bright spot at the top of the mass. We use the movement of this spot to track the movement of the entire mass. Due to the spot's brightness, we can compute the maximum value for each frame, and find the index, which corresponds to the spot. However, there are other bright spots in the video, so we use a Shannon filter in order to crop each frame to an area around the mass before the maximum is computed. The x value and y value of this spot (namely the x and y indices) for each frame are stored in xN and yN vectors where N = 1, 2, 3 corresponds to each of the camera angles.

3.3 Plotting Motion

For all four test cases, the x and y coordinates are plotted over time for each video angle. Changes in y coordinates correspond to vertical motion (i.e movement in the z axis) and changes in the x coordinates correspond to horizontal motion (i.e movement in the x-y plane).

3.4 Computing the SVD

After the x and y values have been extracted from each angle (remember that x values correspond to x-y plane movement and y values correspond to movement in the z direction) they are placed in a matrix and the means of the x and y data are subtracted from that matrix. Three angles means that there are 3 x vectors and 3 y vectors spanning the length of the shortest video for each test case. For example, test case 1 results in a matrix that is 6 x 226. We then compute the SVD (with the 'econ' parameter) on this matrix.

3.4.1 Plotting Components

In order to capture the "core" motion of the spring-mass system, we need to look at how much energy each singular value captures. To compute the energy of each value we use Formula 1 and then plot the energies vs the singular values. After determining the number of singular values that capture the majority of the energy in the system (we will use a threshold of 90%), we plot the respective columns of the V matrix versus time. The columns of the V matrix describe how the data evolves over time, where the first column (corresponding to the first singular value) captures the most information.

4 Computational Results

4.1 Test 1 - Ideal Case

Test 1 is the ideal scenario because there is only motion in the vertical direction with minimal noise.

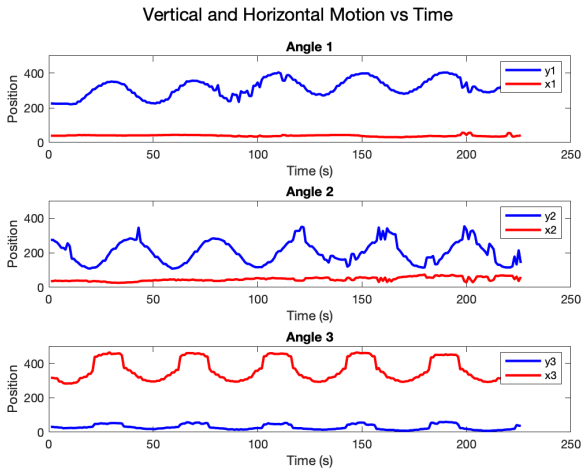

Figure 1: This displays the x and y motion of the mass for the three camera angles. Notice there is oscillation in the y direction for angles 1, 2 and in the x direction for angle 3.

Through image processing we are able to x and y trajectories of the mass in all three videos. The oscillating motion in the x direction for angle 3 is due to the video being rotated 90 degrees compared to angles 1 and 2.

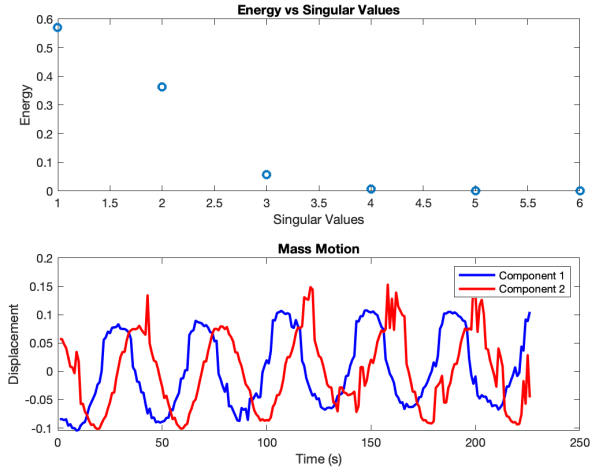

Figure 2: Shows the energies of each singular value and plots the first two columns of the V matrix (corresponding to the first two singular values) against time.

Note that the first two singular values encompass nearly all of the energy in this system (93.3% in this case). Together they represent the best capture of the motion in the spring-mass system. Plotting the two components results in sin and cos respectively. The second component (cos) is the derivative of the first component (sin). These graphs are consistent with the dynamics of a spring-mass system, which oscillate in the form of sin and cos.

4.2 Test 2 - Noisy Case (Camera Shakes)

Test 2 introduced camera shakes, which adds noise to the motion.

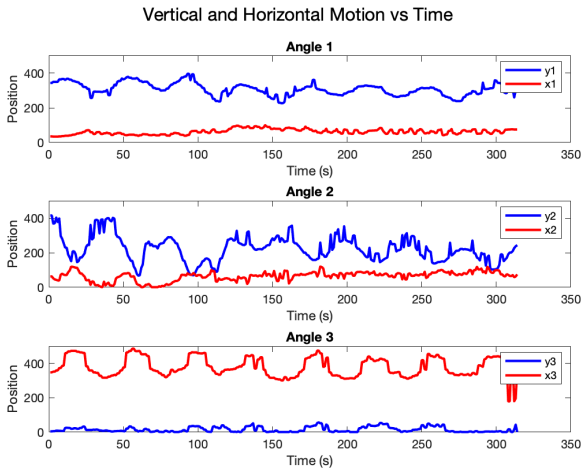

Figure 3: This displays the x and y motion of the mass for the three camera angles in case 2. Notice the additional noise in both the x and y directions, this is due to the camera shake.

The camera shakes in the second test case add noise to the x and y trajectories, which can be seen around the peaks and troughs in the oscillatory motion and as general noise in the "flat" trajectories.

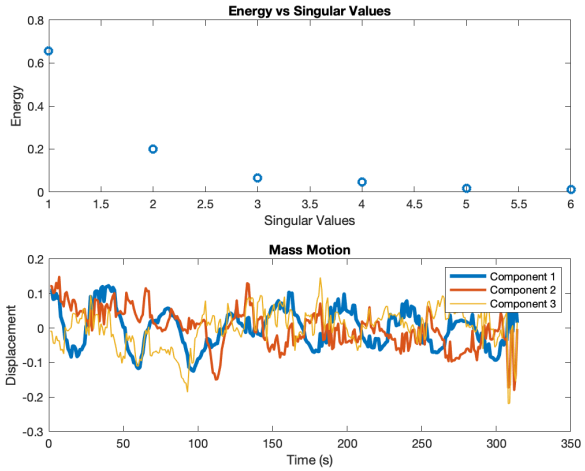

Figure 4: Singular value energy and components plot for test case 2.

In order to reach the threshold of 90% energy capture, the first three singular values must be used. These together capture 92.2% of the energy in the system. We can still see an oscillatory motion defined by the first component. However, the second and third components (representing a much smaller portion of energy than the first) have less of this oscillatory pattern. These components are likely capturing some of the noise from the camera shake, however, we include it in order to reach our 90% energy threshold.

4.3 Test 3 - Vertical and Horizontal Displacement

The third test case introduces a displacement in the x-y plane along with the initial z displacement. This causes the mass to sway in the x-y plane as it bounces up and down in the z direction.

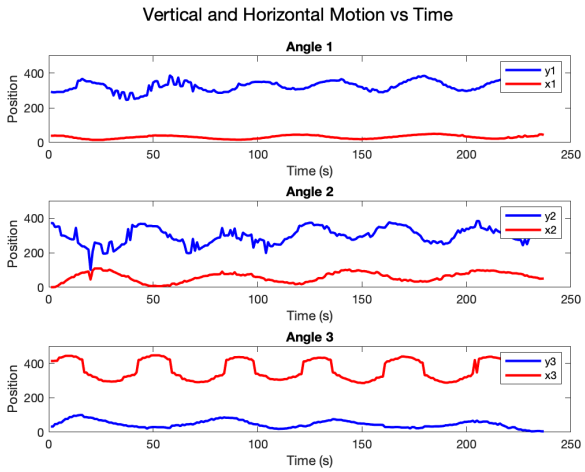

Figure 5: This displays the x and y motion of the mass for the three camera angles in case 3. Notice the oscillatory patterns in both the y trajectories (z direction) and the x direction (x-y plane)

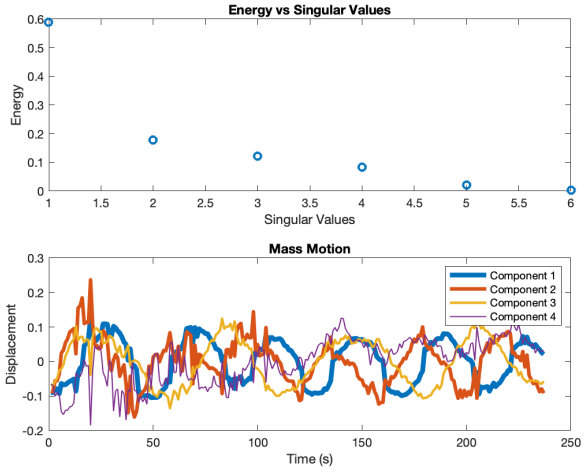

Figure 6: Singular value energy and components plot for test case 3.

The first four components correspond to 97.5% of the energy in the system. All of the components seem to follow an oscillatory motion, which is likely due to the fact that the mass is oscillating in the z direction and in the x-y plane. The z direction oscillation is much more drastic than the x-y motion (which is more of a subtle back and forth), which could explain why the first singular component captures much more energy than the others, and the smooth curve for the first component.

4.4 Test 4 - Vertical Displacement and Rotation

In test case 4, the mass is released off center to give rotation in the x-y plane in addition to oscillating in the z direction.

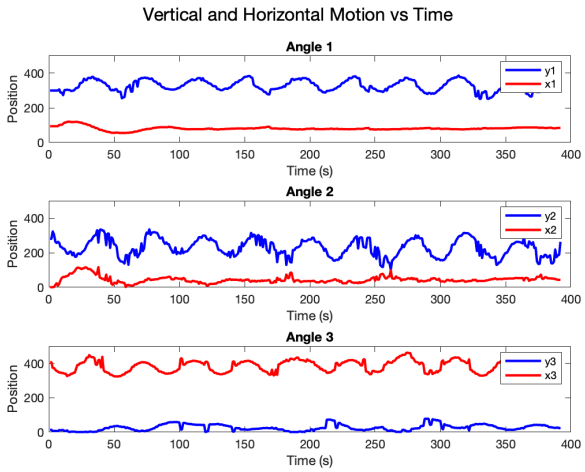

Figure 7: This displays the x and y motion of the mass for the three camera angles in case 4. Notice how there is some oscillation that is dampened over time.

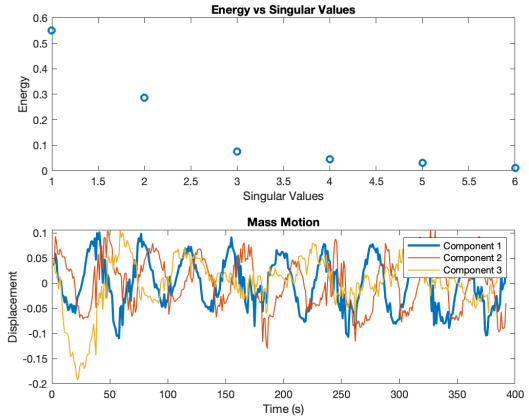

Figure 8: Singular value energy and components plot for test case 4.

Similar to the previous test case, the first singular component captures much of the oscillatory motion in the z direction. We also note that the second component captures oscillation as well. Interestingly, the third component has a large oscillation at the beginning, that is slowly dampened over time (i.e the amplitude gets slightly smaller). This component could represent the rotation in the x-y plane. Moreover, this dampening is consistent with the physics of a Foucault pendulum (a suspended mass in rotation) where the radius of the rotation decreases with time resulting in a smaller amplitude. These components encompass 91.4% of the system's energy.

5 Summary and Conclusion

This assignment uses PCA to extract meaningful information about the dynamics of a spring-mass system given videos of the system. Four different cases were tested, an ideal case of vertical motion, vertical motion with camera shake, horizontal and vertical motion, and lastly vertical motion with rotation. With an energy threshold of 90%, varying amounts of singular components were necessary to capture sufficient information about each system. The SVD is a powerful tool in data analysis, and this assignment aims to showcase its ability to pull out meaningful information from a complex assortment of data.

Appendix A MATLAB Functions

- `y = linspace(x1,x2,n)` returns a row vector of `n` evenly spaced points between `x1` and `x2`.
- `[val,ind] = max(oneD)` returns the maximum value and corresponding index of the vector oneD
- `I2 = im2double(I)` converts the image I to double precision. I can be a grayscale intensity image, a truecolor image, or a binary image. im2double rescales the output from integer data types to the range [0, 1].
- `I = rgb2gray(RGB)` converts the truecolor image RGB to the grayscale image I.
- `[i,j,k] = ind2sub(s,ind)` returns 3 subscript arrays i,j, and k containing the equivalent multidimensional array subscripts equivalent to ind for an array of size s.
- `[U,S,V] = svd(A,'econ')` produces an economy-size decomposition of m-by-n matrix A:

Appendix B MATLAB Code

```
%% Test 1 - Init
clear all; clc
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')

% load the videos and transform them into 3D greyscale matrices
numFrames = size(vidFrames1_1,4);
for j = 1:numFrames
    X1(:,:,j) = im2double(rgb2gray(vidFrames1_1(:,:,:,j)));
    X2(:,:,j) = im2double(rgb2gray(vidFrames2_1(:,:,:,j)));
    X3(:,:,j) = im2double(rgb2gray(vidFrames3_1(:,:,:,j)));
end

%% Test 1 - Image processing
for i = 1:numFrames
    frame1 = X1(:,340-59:340+60,i);
    frame2 = X2(:,300-59:300+60,i);
    frame3 = X3(:,:,i) - mean(X3(:,:,i));
    frame3_filt = frame3(270-29:270+30,:);
    [m1, index1] = max(frame1(:));
    [m2, index2] = max(frame2(:));
    [m3, index3] = max(frame3_filt(:));
    s = [480 120];
    %imshow(frame3_filt)
    [y1(i),x1(i)] = ind2sub(s,index1);
    [y2(i),x2(i)] = ind2sub(s,index2);
    [y3(i),x3(i)] = ind2sub([60 640],index3);
end

%% Test 1 - plotting motion
lims = [0 500];
time_axis = linspace(1,numFrames,numFrames);
subplot(3,1,1);
plot(time_axis,y1,'b-','Linewidth',2)
hold on
```

```
plot(time_axis,x1,'r-','Linewidth',2)
ylim(lims); legend('y1','x1'); ylabel('Position'); xlabel('Time (s)'); title('Angle 1')
subplot(3,1,2);
plot(time_axis,y2,'b-','Linewidth',2)
hold on
plot(time_axis,x2,'r-','Linewidth',2)
ylim(lims); legend('y2','x2'); ylabel('Position'); xlabel('Time (s)'); title('Angle 2')
subplot(3,1,3);
plot(time_axis,y3,'b-','Linewidth',2)
hold on
plot(time_axis,x3,'r-','Linewidth',2)
ylim(lims); legend('y3','x3'); ylabel('Position'); xlabel('Time (s)'); title('Angle 3')

sgtitle('Vertical and Horizontal Motion vs Time')
print('-dpng','test1motion.png')



%% Test 1 - SVD and plotting
mean_vect = [mean(y1); mean(x1); mean(y2); mean(x2); mean(y3); mean(x3)];
X_matrix = [y1;x1;y2;x2;y3;x3] - mean_vect;
[U,S,V] = svd(X_matrix,'econ');
size(U)
size(S)
size(V) % captures the time information
v_trans = V';

% Compute energies
sig = diag(S);
for l = 1:6
    energy(l) = sig(l)^2/sum(sig.^2);
end
sum(energy(1:2))

% Plot energy graphs and dominant singular value position
subplot(2,1,1)
plot(linspace(1,6,6),energy,'o','Linewidth',2)
title('Energy vs Singular Values'); ylabel('Energy'); xlabel('Singular Values')
subplot(2,1,2)
plot(time_axis,v_trans(1,:),'b-','Linewidth',2)
hold on
plot(time_axis,v_trans(2,:),'r-','Linewidth',2)
ylabel('Displacement'); xlabel('Time (s)'); title('Mass Motion'); legend('Component 1','Component 2')


print('-dpng','test1svd.png')

%% Test 2 - init

% the shake introduces more noise in, so may need to consider more
% components

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

```
load('cam3_2.mat')
% note we cut off the last 40 or so frames from videos 2 and 3 (all same
% length of first video)
numFrames2 = size(vidFrames1_2,4);
for j = 1:numFrames2
    A1(:,:,j) = im2double(rgb2gray(vidFrames1_2(:,:,:,j)));
    A2(:,:,j) = im2double(rgb2gray(vidFrames2_2(:,:,:,j)));
    A3(:,:,j) = im2double(rgb2gray(vidFrames3_2(:,:,:,j)));
end
%% Test 2 - Image processing
for i = 1:numFrames2
    frame1a = A1(:,340-59:340+60,i);
    frame2a = A2(:,300-59:300+60,i);
    frame3a = A3(:,:,i) - mean(A3(:,:,i));
    frame3a_filt = frame3a(270-29:270+30,:);
    [m1, index1a] = max(frame1a(:));
    [m2, index2a] = max(frame2a(:));
    [m3, index3a] = max(frame3a_filt(:));
    s = [480 120];
    %imshow(frame3_filt)
    [y1a(i),x1a(i)] = ind2sub(s,index1a);
    [y2a(i),x2a(i)] = ind2sub(s,index2a);
    [y3a(i),x3a(i)] = ind2sub([60 640],index3a);
end

%% Test 2 - Motion plot
lims = [0 500];
time_axisa = linspace(1,numFrames2,numFrames2);
subplot(3,1,1);
plot(time_axisa,y1a,'b-','Linewidth',2)
hold on
plot(time_axisa,x1a,'r-','Linewidth',2)
ylim(lims); legend('y1','x1'); ylabel('Position'); xlabel('Time (s)'); title('Angle 1')
subplot(3,1,2);
plot(time_axisa,y2a,'b-','Linewidth',2)
hold on
plot(time_axisa,x2a,'r-','Linewidth',2)
ylim(lims); legend('y2','x2'); ylabel('Position'); xlabel('Time (s)'); title('Angle 2')
subplot(3,1,3);
plot(time_axisa,y3a,'b-','Linewidth',2)
hold on
plot(time_axisa,x3a,'r-','Linewidth',2)
ylim(lims); legend('y3','x3'); ylabel('Position'); xlabel('Time (s)'); title('Angle 3')

sgtitle('Vertical and Horizontal Motion vs Time')
print('-dpng','test2motion.png')

%% Test 2 - SVD and plotting
mean_vect = [mean(y1a); mean(x1a); mean(y2a); mean(x2a); mean(y3a); mean(x3a)];
A_matrix = [y1a;x1a;y2a;x2a;y3a;x3a] - mean_vect;
[Ua,Sa,Va] = svd(A_matrix,'econ');
va_trans = Va';

% Compute energies
```

```
siga = diag(Sa);
for l = 1:6
    energya(l) = siga(l)^2/sum(siga.^2);
end
sum(energya(1:3))

% Plot energy graphs and dominant singular value position
subplot(2,1,1)
plot(linspace(1,6,6),energya,'o','Linewidth',2)
title('Energy vs Singular Values'); ylabel('Energy'); xlabel('Singular Values')
subplot(2,1,2)
plot(time_axisa,va_trans(1,:),'Linewidth',3)
hold on
plot(time_axisa,va_trans(2,:),'Linewidth',2)
plot(time_axisa,va_trans(3,:),'Linewidth',1)
ylabel('Displacement'); xlabel('Time (s)'); title('Mass Motion'); legend('Component 1','Component 2','C

print('-dpng','test2svd.png')

%% Test 3 - Init
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
% note we cut off the last 40 or so frames from videos 2 and 3 (all same
% length of first video)
numFrames3 = size(vidFrames3_3,4);
for j = 1:numFrames3
    B1(:,:,j) = im2double(rgb2gray(vidFrames1_3(:,:,:,j)));
    B2(:,:,j) = im2double(rgb2gray(vidFrames2_3(:,:,:,j)));
    B3(:,:,j) = im2double(rgb2gray(vidFrames3_3(:,:,:,j)));
end

%% Test 3 - Image Processing
for i = 1:numFrames3
    frame1b = B1(:,340-59:340+60,i);
    frame2b = B2(:,300-59:300+60,i);
    frame3b = B3(:,:,i) - mean(B3(:,:,i));
    frame3b_filt = frame3b(270-69:270+70,:);
    [m1, index1b] = max(frame1b(:));
    [m2, index2b] = max(frame2b(:));
    [m3, index3b] = max(frame3b_filt(:));
    s = [480 120];
    %imshow(frame3_filt)
    [y1b(i),x1b(i)] = ind2sub(s,index1b);
    [y2b(i),x2b(i)] = ind2sub(s,index2b);
    [y3b(i),x3b(i)] = ind2sub([140 640],index3b);
end

%% Test 3 - Motion Plot
lims = [0 500];
time_axisb = linspace(1,numFrames3,numFrames3);
subplot(3,1,1);
plot(time_axisb,y1b,'b-','Linewidth',2)
hold on
```

```
plot(time_axisb,x1b,'r-','Linewidth',2)
ylim(lims); legend('y1','x1'); ylabel('Position'); xlabel('Time (s)'); title('Angle 1')
subplot(3,1,2);
plot(time_axisb,y2b,'b-','Linewidth',2)
hold on
plot(time_axisb,x2b,'r-','Linewidth',2)
ylim(lims); legend('y2','x2'); ylabel('Position'); xlabel('Time (s)'); title('Angle 2')
subplot(3,1,3);
plot(time_axisb,y3b,'b-','Linewidth',2)
hold on
plot(time_axisb,x3b,'r-','Linewidth',2)
ylim(lims); legend('y3','x3'); ylabel('Position'); xlabel('Time (s)'); title('Angle 3')

sgtitle('Vertical and Horizontal Motion vs Time')
print('-dpng','test3motion.png')

%% Test 3 - SVD and plotting
mean_vect = [mean(y1b); mean(x1b); mean(y2b); mean(x2b); mean(y3b); mean(x3b)];
B_matrix = [y1b;x1b;y2b;x2b;y3b;x3b] - mean_vect;
[Ub,Sb,Vb] = svd(B_matrix,'econ');
vb_trans = Vb';

% Compute energies
sigb = diag(Sb);
for l = 1:6
    energyb(l) = sigb(l)^2/sum(sigb.^2);
end
sum(energyb(1:4))

% Plot energy graphs and dominant singular value position
subplot(2,1,1)
plot(linspace(1,6,6),energyb,'o','Linewidth',2)
title('Energy vs Singular Values'); ylabel('Energy'); xlabel('Singular Values')
subplot(2,1,2)
plot(time_axisb,vb_trans(1,:),'Linewidth',4)
hold on
plot(time_axisb,vb_trans(2,:),'Linewidth',3)
plot(time_axisb,vb_trans(3,:),'Linewidth',2)
plot(time_axisb,vb_trans(4,:),'Linewidth',1)
ylabel('Displacement'); xlabel('Time (s)'); title('Mass Motion'); legend('Component 1','Component 2','C

print('-dpng','test3svd.png')


%% Test 4 - Init
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
numFrames4 = size(vidFrames1_4,4);
for j = 1:numFrames4
    C1(:,:,j) = im2double(rgb2gray(vidFrames1_4(:,:,:,j)));
    C2(:,:,j) = im2double(rgb2gray(vidFrames2_4(:,:,:,j)));
    C3(:,:,j) = im2double(rgb2gray(vidFrames3_4(:,:,:,j)));
end
```

```
%% Test 4 - Image Processing
for i = 1:numFrames4
    frame1c = C1(:,340-59:340+60,i);
    frame2c = C2(:,300-59:300+60,i);
    frame3c = C3(:,:,i) - mean(C3(:,:,i));
    frame3c_filt = frame3c(270-79:270+80,:);
    [m1, index1c] = max(frame1c(:));
    [m2, index2c] = max(frame2c(:));
    [m3, index3c] = max(frame3c_filt(:));
    s = [480 120];
    %imshow(frame3_filt)
    [y1c(i),x1c(i)] = ind2sub(s,index1c);
    [y2c(i),x2c(i)] = ind2sub(s,index2c);
    [y3c(i),x3c(i)] = ind2sub([160 640],index3c);
end

%% Test 4 - Motion Plotting
lims = [0 500];
time_axisc = linspace(1,numFrames4,numFrames4);
subplot(3,1,1);
plot(time_axisc,y1c,'b-','Linewidth',2)
hold on
plot(time_axisc,x1c,'r-','Linewidth',2)
ylim(lims); legend('y1','x1'); ylabel('Position'); xlabel('Time (s)'); title('Angle 1')
subplot(3,1,2);
plot(time_axisc,y2c,'b-','Linewidth',2)
hold on
plot(time_axisc,x2c,'r-','Linewidth',2)
ylim(lims); legend('y2','x2'); ylabel('Position'); xlabel('Time (s)'); title('Angle 2')
subplot(3,1,3);
plot(time_axisc,y3c,'b-','Linewidth',2)
hold on
plot(time_axisc,x3c,'r-','Linewidth',2)
ylim(lims); legend('y3','x3'); ylabel('Position'); xlabel('Time (s)'); title('Angle 3')

sgtitle('Vertical and Horizontal Motion vs Time')
print('-dpng','test4motion.png')

%% Test 4 - SVD and plotting
mean_vect = [mean(y1c); mean(x1c); mean(y2c); mean(x2c); mean(y3c); mean(x3c)];
C_matrix = [y1c;x1c;y2c;x2c;y3c;x3c] - mean_vect;
[Uc,Sc,Vc] = svd(C_matrix,'econ');
vc_trans = Vc';

% Compute energies
sigc = diag(Sc);
for l = 1:6
    energyc(l) = sigc(l)^2/sum(sigc.^2);
end
sum(energyc(1:3))

% Plot energy graphs and dominant singular value position
subplot(2,1,1)
```

```
plot(linspace(1,6,6),energyc,'o','Linewidth',2)
title('Energy vs Singular Values'); ylabel('Energy'); xlabel('Singular Values')
subplot(2,1,2)
plot(time_axisc,vc_trans(1,:),'Linewidth',2)
hold on
plot(time_axisc,vc_trans(2,:),'Linewidth',1)
plot(time_axisc,vc_trans(3,:),'Linewidth',1)
ylabel('Displacement'); xlabel('Time (s)'); title('Mass Motion'); title('Mass Motion'); legend('Componer

print('-dpng','test4svd.png')
```

Code from homework3.m