AMATH 482 Homework 3: **Principal Component Analysis and Movement Detection**

Amelia Nathan

February 24, 2021

Abstract

Shown through implementing Singular Value Decomposition and Principal Component Analysis on a series of videos capturing test cases of a mass-spring system, this paper investigates the utility of PCA as systems contain varied amounts of noise and complex movement. After cleaning and processing data as shown in the algorithm and implementation section, we ultimately find that while PCA can be very valuable in understanding system behavior, interpreting movement specifics can be especially difficult when noise is present.

1 Introduction and Overview

1.1 Introduction to Principal Component Analysis

In many applications such as physics experimentation and data collection leads to complex representations that we often do not have equations for. This can pose a barrier to analyzing the system and associated movement, for example. However, Principal component analysis (PCA) is a technique that utilizes Singular Value Decomposition (SVD) to reduce complex systems and data to detect the significant information, in this case mass displacement. PCA is therefore useful in expanding ideas founded in experiments and data collection to predictive modeling and ultimately helps us understand and visualize data and complex situations.

1.2 Problem Overview

In this problem we are presented with matlab video files depicting four different cases of a spring-mass system. Each of the four test cases are captured with three cameras and we are tasked with using the captured film and principal component analysis to track the mass' movement throughout the video. The four test cases include a variety of noise and different movements within the system and we must apply PCA on all cases to discover the impact of noise on the utility of the algorithm. The four cases are:

1. Ideal Case: Simple vertical oscillations
2. Noisy Case: Vertical oscillations with unsteady camera work
3. Horizontal Displacement: Horizontal movement in addition to vertical movement
4. Horizontal Displacement and Rotation: Rotation in addition to horizontal and vertical movement

2 Theoretical Background

2.1 Key Matrix Foundational Ideas

A key theory directly related to matrices that gives the SVD significance in practice is the following (from lecture notes):

If A is a matrix of rank r, then A is the sum of r rank 1 matrices:

$$A = \Sigma_{j=1}^{r} \sigma_j u_j v_j^* \tag{1}$$

where $u_j v_j^*$ is the outer product and the resulting matrix is rank 1. Ultimately, this foundational idea leads to the possibility of dimensionality reduction which, for high rank matrices, greatly reduces computational time. This notion of approximation greatly informs principal component analysis as an analytical technique. As will be discussed in the next theoretical background section, Singular Value Decomposition is a technique that is based on this idea of dimensionality reduction and approximation. SVD is a tool for capturing the most relevant information about data while reducing the dimensionality of the storage matrix as much as possible, decreasing complexity and computational effort.

2.2 Singular Value Decomposition

Singular value decomposition is a factorization technique that stretches and transforms vectors by using two unitary matrices U and V and a diagonal matrix Σ. The simple definition of the SVD is given by the following equation where A is the matrix being deconstructed:

$$A = U \Sigma V^*, \tag{2}$$
$$U \in \mathbb{R}^{mxm}, \quad V \in \mathbb{R}^{nxn}, \quad \Sigma \in \mathbb{R}^{mxn}$$

The matrix Σ is a diagonal matrix with singular values along the diagonal. U and V are unitary and geometrically, do the following as described in lecture notes:

- Multiplying by V^* facilitates rotation
- Multiplying by Σ facilitates stretching
- Multiplying by U facilitates proper orientation

As previously mentioned, Σ contains singular values. U and V are unitary matrices; the columns of U contain left singular vectors and the columns of V are right singular vectors, all with respect to the factored matrix A. Some key properties augment the utility of the SVD (also found in lecture 11):

1. SVD can be performed on all matrices A $\in \mathbb{C}^{mxn}$
2. Singular values are non-negative and real and unique to matrix A
3. Singular values descend in magnitude along the diagonal
4. The rank of A corresponds to its number of nonzero singular values
5. The columns of U form the basis for the range of A and the columns of V plus 1 form the basis for the null space of A.

In finding the SVD, it is important to note that eigenvalues and eigenvectors form a key component. Calculating the eigenvalues of $A^T A$ and AA^T is a key step as the singular values are the squares of these eigenvalues (both of these matrix products have the same eigenvalues).

2.3 Principal Component Analysis

Principal component analysis is an extension of singular value decomposition; in short, the principal components are values corresponding to best fit vectors that reduce the complexity and, when accurate, closely match the data to demonstrate trends. The best fit is found and defined as the minimum average squared distance from points to line. The key equation in principal component analysis that also demonstrates the connection to SVD as it incorporates the U matrix is:

$$Y = U^T X \tag{3}$$

This equation is put in context in the next theoretical section as we go over the statistical ideas that form the foundation for PCA.

2.4 Statistical Background

Variance and covariance are two significant statistical concepts that inform principal component analysis. Variance is the spread of the data and the square is given by

$$\sigma^2 = \frac{1}{n}\Sigma_{k=1}^{n}(a_k - \mu)^2 \tag{4}$$

If we assume we have subtracted the mean (μ above) and subtract one from our denominator to account for under-prediction, we can write variance as an inner product that will be used in the algorithm section and serves as an unbiased estimator:

$$\sigma^2 = \frac{1}{n-1}aa^T \tag{5}$$

Covariance measures variables' variance with respect to each other and is represented by the following equation. Here a and b are the variables interacting with each other.

$$\sigma_{ab}^2 = \frac{1}{n-1}ab^T \tag{6}$$

An important concept to note is that a covariance of 0 indicates the two variables are uncorrelated, or statistically independent. High covariance indicates redundancy within data; as will be seen in the subsequent sections, the SVD and PCA methods used utilize covariance as achieving a covariance of 0 indicates redundancy has been removed as the principal components seek to capture as much unique information as possible while using the least amount of storage. C_X and C_Y as calculated with relation to the SVD and PCA are below. The final Sigma matrix, as described in 2.2 has 0 entries on all off-diagonals, indicating that the variables in Y are uncorrelated.

$$C_X = \frac{1}{n-1}XX^T = AA^T \tag{7}$$

$$C_X = U\Sigma^2U^T \tag{8}$$

$$\text{For a change of basis, multiply: } Y = U^TX \tag{9}$$

$$C_Y = \frac{1}{n-1}YY^T = \frac{1}{n-1}U^TXX^TU = U^TAA^TU = U^TU\Sigma^2U^TU = \Sigma^2 \tag{10}$$

3 Algorithm Implementation and Development

As detailed in the two algorithms below, the main process for implementing principal component analysis to track movement in the string-mass system involved loading the data, processing and cleaning it to create consistent length vectors of means based on grayscale video frames and then applying the svd and plotting principal components. I have broken this down into the initial processing and cleaning algorithm (1) and then combining data and analyzing using means, svd, and principal components (2).

Algorithm 1: Mass Extraction

```
Load camera data
Initialize cumulative matrix to track means
for every video frame do
  Use rgb2gray() to convert to grayscale and cast as double
  Use a filter to isolate the whitest parts of the frame
  Use find to get location of filtered components
  Use ind2sub to convert location to index subscripts
  Calculate means for x and y coordinates
  Append means to cumulative matrix
end for
Get length of matrix row using size
```

Algorithm 2: Principal Component Analysis

```
Use min to find shortest video file in terms of number of frames
for each camera means matrix do
  Truncate to length of shortest
  Take transpose
end for
Combine camera means to a matrix for the overall test
Use size to get size of new matrix
Calculate mean for each row of new matrix
Subtract mean from each matrix entry
Use svd to get $\Sigma$, U, V matrices
Project principal components
plot singular value energies and relevant principal components
```

4 Computational Results

Figure 1 and 2 below show principal components and singular values for each of the four test cases. In figure 1, for all four test cases I plotted the first component in blue and second in cyan to aid comparison.

As seen in the test 1 plot in figure 1, Principal Component Analysis was relatively successful in tracking the mass' movement. The blue plot indicates periodic oscillations and each period roughly decreases in amplitude, which is logical given the decrease in displacement per oscillation as the spring-mass system continues in time which is clear in the video. The Test 1 plot of singular values in figure two highlights that for an ideal case like this, principal component analysis and singular value decomposition are well-suited to the task. The first principal component contains about 85% of the information of the system and the second and other successive components all contain less than 20%. I did decide to plot the second component, however, because the oscillatory movement detected is also clear. This indicates that the rather clear movement in the video was consistently detected by our analysis, which is not the case in some of the noisier cases.

The dark blue line in test 2 also demonstrates that the vertical periodic movement was successfully detected. The cyan and magenta plots of the second and third component also resemble periodic functions and are in about the same range as the first principle component, with the amplitudes occurring between about $\pm$ 150. The noisy data was more difficult for our algorithms to detect, and our plot in figure 2 for test 2 shows that our first 4, and especially 3, singular values contain much more energy and therefore information than the second and third values in the first test case. Looking at the second plot in figure 1, however, does indicate that despite the difficulty our algorithm had in tracking just the bucket's movement, the principal components do rather successfully track the overall motion detected in the video. While it is difficult to say which principal component plot correlates to certain aspects of the video, when watching the original video from camera one the noise–camera shakes–are relatively repetitive from side to side. This therefore does seem to be captured somewhat successfully in second and third components.

The third test case, seen in the lower left plot in figure 2, was similar to test 2 in that the first four singular values contained most of the information gathered through svd and pca, compared to 1 in which the first singular value contained almost all of the energy and information. I therefore opted to plot the first four principle components. Shown in figure 1, all four of these components display somewhat periodic movement, but it is difficult to distinguish the mass' movement. The dark blue line, as expected, has the most consistent behavior and would seemingly resemble the bucket's movement most closely, which was predominantly vertical with some horizontal swaying in this case. The videos display repetitive swaying movement from side to side as the bucket moves vertically, creating almost circular motion, which is largely represented by the four plotted components. However, had I not seen the initial videos to begin with, this test case exemplifies limitations in Principal Component Analysis as the plot is very difficult to interpret.

The fourth case, similar to the first, has most information about the system contained within the first singular value. Also similar to the first case, the oscillations are rather consistent and clear in the plot in figure 1. As with other cases, the limitation of identifying exactly what the principal components tell us about the mass' movement is difficult. However, comparing the three camera videos to the plotted principal components, I can somewhat infer the relationship. In the video, the rotation and vertical motion is clearly predominant. The rotation is also very consistent in terms of speed, although it does change direction and vary slightly occasionally. The first principal component possibly indicates this as the periodic behavior is consistent except for some large spikes in movement. The cyan line could also indicate rotational movement and vertical movement as, like in the first case, vertical displacement was consistent and clear to detect in the videos.

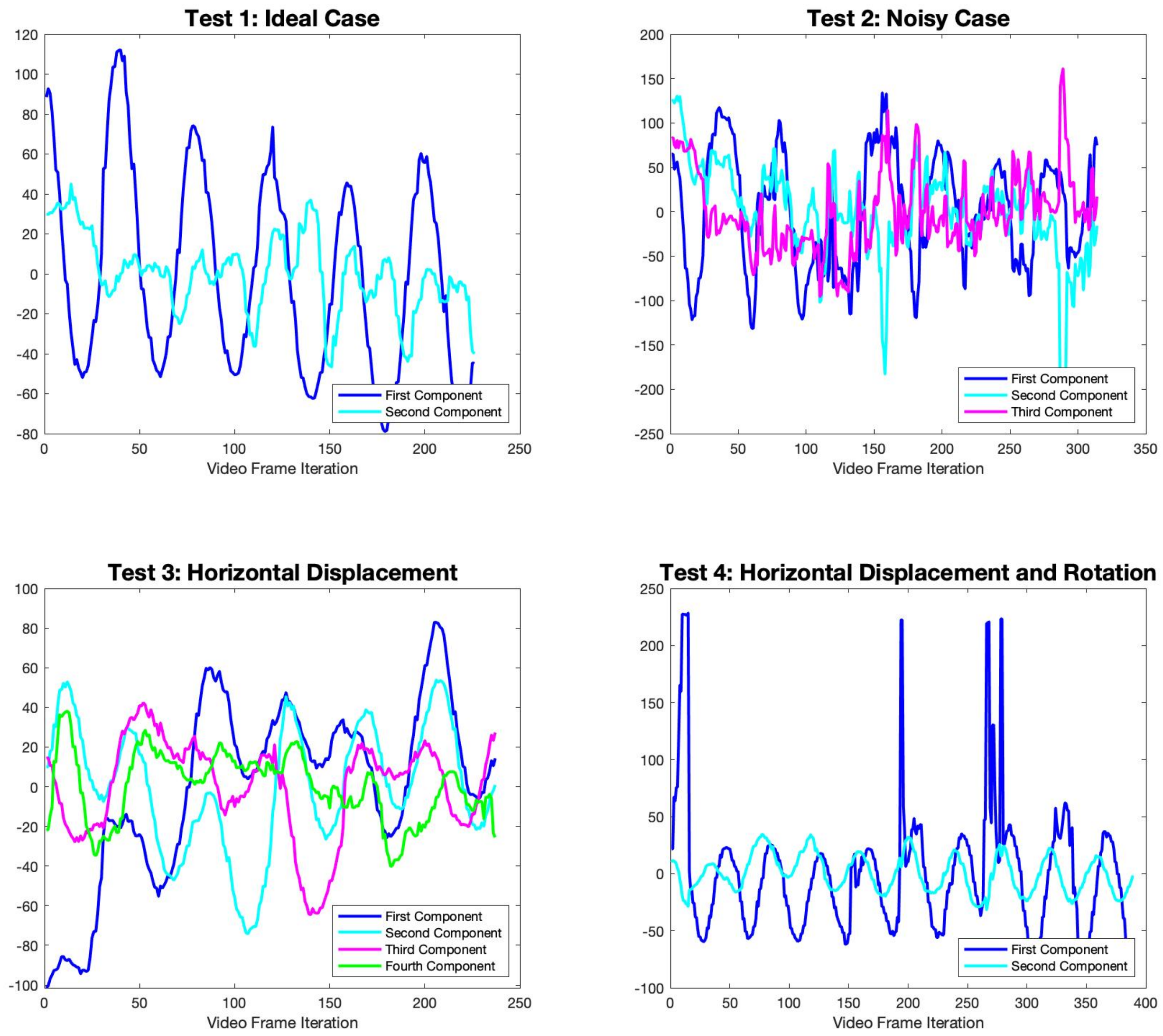

Figure 1: Principal Components for various test cases

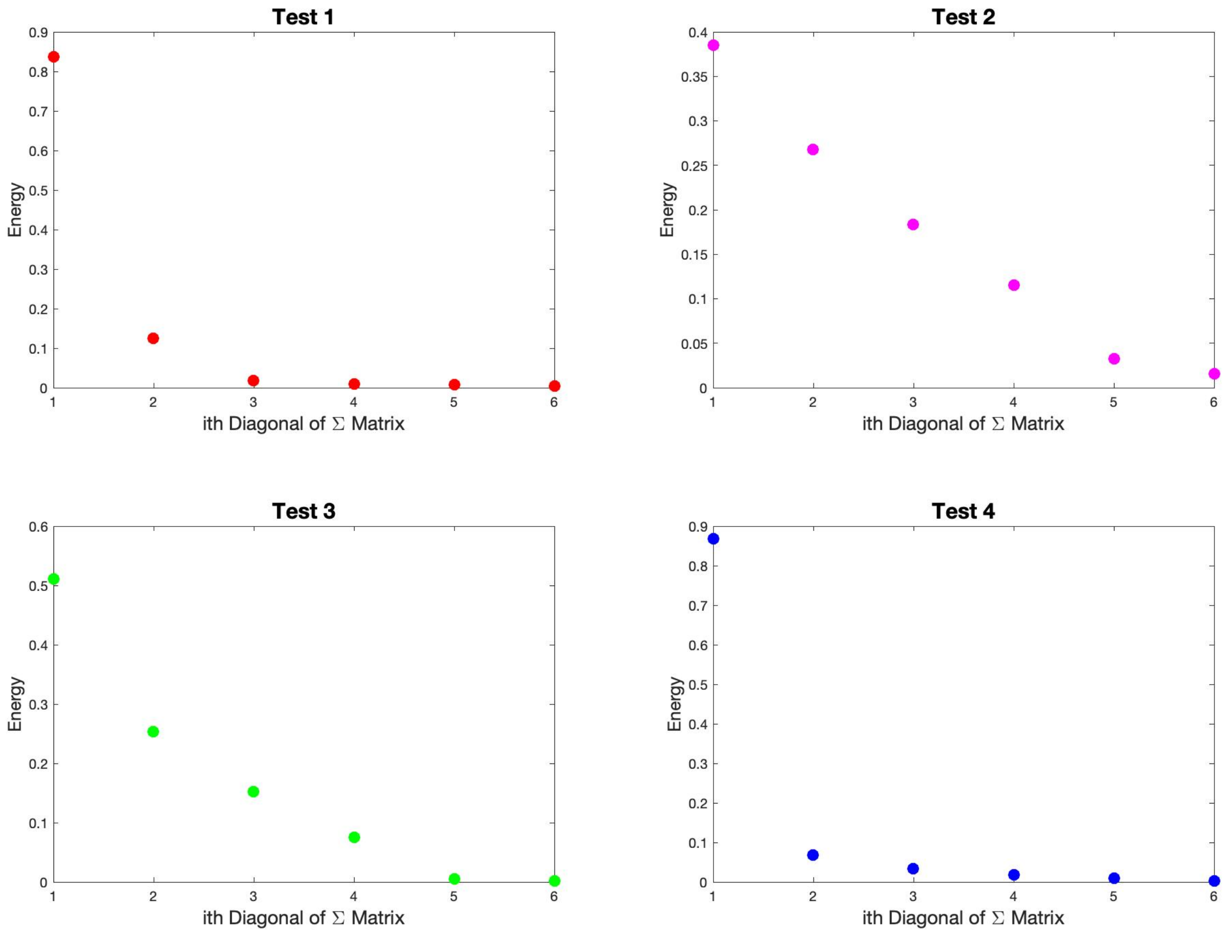

Figure 2: Singular Value Energies

5 Summary and Conclusions

As has hopefully been made clear in the computational results discussion, the Singular Value Decomposition and Principal Component Analysis used holds a lot of value in detecting and understanding systems, but is also limited. While the plots rather successfully indicated the relative magnitude and presence of motion and noise in the four cases, understanding what the principal components actually displayed in terms of movement was difficult; a lot of the discussion above is speculative and made possible by the fact that I had seen the videos and could compare them directly to the plots. On its own, the principal components detected motion well in the first and fourth test cases, but as more complex movement (and especially combinations of complex movement) and noise were added interpreting the movement based solely on SVD and PCA became very difficult.

Appendix A MATLAB Functions

- `min(X)` Finds the minimum value within X
- `mean(X)` Finds the mean of values in X
- `[x, y] = size(X)` Returns the dimensions of a matrix X in terms of number of rows and columns
- `[U, S, V] = svd(A)` Computes the singular value decomposition of matrix A, returning a diagonal matrix S and two unitary matrices U and V such that X = U*S*V'
- `repmat(A,M,N)` replicates the value A in an mxn matrix
- `diag(V)` Creates a diagonal matrix with vector V along the diagonal
- `double` Casts values as type double
- `rmmissing` Removes NaN values from matrix
- `ind2sub` returns subscript indices for a location in a matrix

Appendix B MATLAB Code

```matlab
% AMATH 482
% Assignment 3

%% Test 1: Ideal Case
close all

load('cam1_1.mat');
[l1_1, means1_1] = loadMovie(vidFrames1_1);

load('cam2_1.mat');
[l2_1, means2_1] = loadMovie(vidFrames2_1);

load('cam3_1.mat');
[l3_1, means3_1] = loadMovie(vidFrames3_1);

length = min([l1_1, l2_1, l3_1]); % find the video length with the smallest number of frames
means1_1 = means1_1(1:length, :);
means2_1 = means2_1(1:length, :);
means3_1 = means3_1(1:length, :);
test1 = [means1_1'; means2_1'; means3_1'];
[m1, n1] = size(test1);
mn = mean(test1, 2);
test1 = test1-repmat(mn,1,n1);
[U1, S1, V1] = svd(test1'/sqrt(n1-1)); % perform singular value decomposition
lambda1 = diag(S1).^2; % diagonal variances
Y1 = test1' * V1; % principal components projection, plot first two

% Plot Principal Components
subplot(2,2,1)
plot(1:226, Y1(:,1), 'b', 'Linewidth', 2)
hold on
plot(1:226, Y1(:,2), 'c', 'Linewidth', 2) % plot second principal component
title('Test 1: Ideal Case', 'Fontsize', 16)
xlabel('Video Frame Iteration')
legend('First Component', 'Second Component', 'Location', 'southeast')


%% Test 2: Noisy Case

load('cam1_2.mat');
[l1_2, means1_2] = loadMovie(vidFrames1_2);

load('cam2_2.mat');
[l2_2, means2_2] = loadMovie(vidFrames2_2);

load('cam3_2.mat');
[l3_2, means3_2] = loadMovie(vidFrames3_2);

length2 = min([l1_2, l2_2, l3_2]); % find the video length with the smallest number of frames
means1_2 = means1_2(1:length2, :);
means2_2 = means2_2(1:length2, :);
means3_2 = means3_2(1:length2, :);
test2 = [means1_2'; means2_2'; means3_2'];
```

```
[m2, n2] = size(test2);
mn2 = mean(test2, 2);
test2 = test2-repmat(mn2,1,n2);
[U2, S2, V2] = svd(test2'/sqrt(n2-1)); % perform singular value decomposition
lambda2 = diag(S2).^2; % diagonal variances
Y2 = test2' * V2; % principal components projection, plot first two

% Plot Principal Components
subplot(2,2,2)
plot(1:length2, Y2(:,1), 'b', 'Linewidth', 2)
hold on
plot(1:length2, Y2(:,2), 'c', 'Linewidth', 2)
plot(1:length2, Y2(:,3), 'm', 'Linewidth', 2)
title('Test 2: Noisy Case', 'Fontsize', 16)
xlabel('Video Frame Iteration')
legend('First Component', 'Second Component', 'Third Component', 'Location', 'southeast')


%% Test 3: Horizontal Displacement

load('cam1_3.mat');
[l1_3, means1_3] = loadMovie(vidFrames1_3);

load('cam2_3.mat');
[l2_3, means2_3] = loadMovie(vidFrames2_3);

load('cam3_3.mat');
[l3_3, means3_3] = loadMovie(vidFrames3_3);

length3 = min([l1_3, l2_3, l3_3]); % find the video length with the smallest number of frames
means1_3 = means1_3(1:length3, :)';
means2_3 = means2_3(1:length3, :)';
means3_3 = means3_3(1:length3, :)';
test3 = [means1_3; means2_3; means3_3];

[m3, n3] = size(test3);
mn3 = mean(test3, 2);
test3 = test3-repmat(mn3,1,n3);
[U3, S3, V3] = svd(test3'/sqrt(n3-1)); % perform singular value decomposition
lambda3 = diag(S3).^2; % diagonal variances
Y3 = test3' * V3; % principal components projection, plot first two

% Plot Principal Components
subplot(2,2,3)
plot(1:length3, Y3(:,1), 'b', 'Linewidth', 2)
hold on
plot(1:length3, Y3(:,2), 'c', 'Linewidth', 2) % plot second principal component
plot(1:length3, Y3(:,3), 'm', 'Linewidth', 2) % third
plot(1:length3, Y3(:,4), 'g', 'Linewidth', 2)
title('Test 3: Horizontal Displacement', 'Fontsize', 16)
xlabel('Video Frame Iteration')
legend('First Component', 'Second Component', 'Third Component',
'Fourth Component', 'Location', 'southeast')
```

```
%% Test 4: Horizontal Displacement and Rotation

load('cam1_4.mat');
[l1_4, means1_4] = loadMovie(vidFrames1_4);

load('cam2_4.mat');
[l2_4, means2_4] = loadMovie(vidFrames2_4);

load('cam3_4.mat');
[l3_4, means3_4] = loadMovie(vidFrames3_4);

length4 = min([l1_4, l2_4, l3_4]); % find the video length with the smallest number of frames
means1_4 = means1_4(1:length4, :)';
means2_4 = means2_4(1:length4, :)';
means3_4 = means3_4(1:length4, :)';
test4 = [means1_4; means2_4; means3_4];

[m4, n4] = size(test4);
mn4 = mean(test4, 2);
test4 = test4-repmat(mn4,1,n4);
[U4, S4, V4] = svd(test4'/sqrt(n4-1)); % perform singular value decomposition
lambda4 = diag(S4).^2; % diagonal variances
Y4 = test4' * V4; % principal components projection, plot first two

% Plot Principal Components
subplot(2,2,4)
plot(1:length4, Y4(:,1), 'b', 'Linewidth', 2)
hold on
plot(1:length4, Y4(:,2), 'c', 'Linewidth', 2) % plot second principal component
title('Test 4: Horizontal Displacement and Rotation', 'Fontsize', 16)
xlabel('Video Frame Iteration')
legend('First Component', 'Second Component', 'Location', 'southeast')

%% Singular Value Energy Plotting
figure(2)
subplot(2,2,1)
plot(1:6, lambda1/sum(lambda1), 'r.', 'MarkerSize', 30, 'Linewidth', 2)
title('Test 1', 'Fontsize', 16)
ylabel('Energy','Fontsize', 13)
xlabel('ith Diagonal of \Sigma Matrix','Fontsize',14)
xticks([1 2 3 4 5 6])

subplot(2,2,2)
plot(1:6, lambda2/sum(lambda2), 'm.', 'MarkerSize', 30, 'Linewidth', 2)
title('Test 2', 'Fontsize', 16)
ylabel('Energy','Fontsize', 13)
xlabel('ith Diagonal of \Sigma Matrix','Fontsize',14)
xticks([1 2 3 4 5 6])

subplot(2,2,3)
plot(1:6, lambda3/sum(lambda3), 'g.', 'MarkerSize', 30, 'Linewidth', 2)
title('Test 3', 'Fontsize', 16)
ylabel('Energy','Fontsize', 13)
xlabel('ith Diagonal of \Sigma Matrix','Fontsize',14)
xticks([1 2 3 4 5 6])
```

```
subplot(2,2,4)
plot(1:6, lambda4/sum(lambda4), 'b.', 'MarkerSize', 30, 'Linewidth', 2)
title('Test 4', 'Fontsize', 16)
ylabel('Energy','Fontsize', 13)
xlabel('ith Diagonal of \Sigma Matrix','Fontsize',14)
xticks([1 2 3 4 5 6])

function [length, all_means] = loadMovie(tplay)
    % LOADMOVIE Converts each frame of a video to grayscale
    % Filters grayscale to focus on white components to track bucket
    % movement. Finds coordinate locations of focused white components to
    % track movement. Takes the mean of x and y coordinates and puts them
    % into one large matrix of the averaged coordinates, which the function
    % returns. Also returns the length of the final matrix for use in
    % further analysis.

    % implay(tplay); % optional method to play the video
    all_means = [];
    numFrames = size(tplay,4);
    for j = 1:numFrames
        X = tplay(:,:,:,j);
        X = double(rgb2gray(X)); % convert each video frame to grayscale
        white = X > 240;
        location = find(white);
        [y, x]= ind2sub(size(white), location);
        y_mean = mean(y);
        x_mean = mean(x);
        all_means = [all_means; y_mean, x_mean];
        all_means = rmmissing(all_means); % remove NaN values from data
       % imshow(white)
       % drawnow
    end
    [rows, ~] = size(all_means);
    length = rows;

end
```