AMATH 482 Homework 3

Clayton Heath

February 24, 2021

Abstract

We use the *Singular Value Decomposition* (SVD) and *Principal Component Analysis* (PCA) in order to understand the motion of a spring-mass system. We do this by recording a video of the spring-mass system from three different cameras, each from different angles. We do this for four different tests, an ideal case where system just has displacement in the z direction, a noisy case, where it is the same as the ideal case expect the cameras shake, a horizontal displacement case where there is movement in x-y direction, and a case with both horizontal displacement and rotation. We then will evaluate the information that the SVD and PCA are able to produce from the different tests.

1 Introduction and Overview

We will first import the videos into MATLAB. We will then group each of the video sets for each test together, and cut them so that they all start at the same time and are the same length. Then crop the videos so that to cut out as much as we can without cutting off the can. We then process each frame of the videos to track the position of the can for each of the 3 cameras per test. We then build a matrix out of these coordinates and use SVD and PCA in order to plot the movement of the can in each test.

2 Theoretical Background

The *Singular Value Decomposition* is a method of decomposing some $m \times n$ matrix A such that

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

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m\times n}$. The values σ_n on the diagonal of Σ are called the *singular values* of A. The column vectors of U are called the *left singular vectors* of A and the column vectors of V are called the are called the *right singular vectors* of A.
Principal Component Analysis (PCA) is a low dimension reduction of data. Let X be a matrix composed of row vectors of equal length a, b, c, d. Then, we can compute the covariances between the rows of X with

$$C_X = \frac{1}{n-1}XX^T = \begin{bmatrix} \sigma_a^2 & \sigma_{ab}^2 & \sigma_{ac}^2 & \sigma_{ad}^2 \\ \sigma_{ba}^2 & \sigma_b^2 & \sigma_{bc}^2 & \sigma_{bd}^2 \\ \sigma_{ca}^2 & \sigma_{cb}^2 & \sigma_c^2 & \sigma_{cd}^2 \\ \sigma_{da}^2 & \sigma_{db}^2 & \sigma_{dc}^2 & \sigma_d^2 \end{bmatrix} \tag{2}$$

This is called the *covariance matrix*. This brings us closer to the goal of PCA, which is to find a change of basis so that the variables are uncorrelated. Meaning each variable will contain information so that there are no redundancies between them. We also want to know which variables have the largest variance since that means they contain the most information. Thus we diagonalize the matrix:

$$C_X = V\Lambda V^{-1}$$

Now we have that the basis of eigenvectors in V are the principal components, these are uncorrelated since they must be orthogonal. The diagonal elements of Λ are the variances of these variables.
Now, connecting SVD and PCA, let $A = \frac{1}{\sqrt{n-1}}X$, then from Equation 2 we get

$$C_X = \frac{1}{n-1}XX^T = AA^T \tag{3}$$

and from Equation (1) we get

$$AA^T = U\Sigma^2 U^T \tag{4}$$

where U is the orthogonal matrix of left singular vectors and Σ is the diagonal matrix of singular values. Thus, the eigenvalues of the covariance matrix are the squares of the singular values. These singular values were scaled by a factor of $\sqrt{n-1}$ to get to the correct units.

3 Algorithm Implementation and Development

The first thing we need to do, is to crop and align all of the video. We do this by watching all three videos for a given test at the same time, we then realize they are all different lengths and start at different times. So we manually cut each video such that they start at the same time, and cut off the ends of videos so that they are all the same length. We then crop each video down to as small as possible so that the can and the flashlight on the can stay visible for the duration of the video. This makes it so that there is less information in each video in hopes for cleaner data when tracking the can. We do these cuts and crops for all 4 of the tests and can be seen at the beginning of each `Video N Setup` section of the MATLAB code where $N = 1, 2, 3, 4$.

We then create row vectors to hold the x coordinates and y coordinates for the can at each frame of the 3 videos, thus we have 6 row vectors. Then we loop over each frame of the 3 videos, turning the frames into black and white, then finding the x and y coordinates of the max value of the frame, which we are hoping is the flashlight at the top of the can. We then put the coordinates into its corresponding row vector at the current frame. We again do this for each of the 4 cases under their respective sections in the MATLAB code.

Then, we take our 6 row vectors and build a matrix out of them. Thus we have 6 dimensions of movement for our can, 2 from each camera. But, clearly this video is based in reality, and our can could only actually move in 3 dimensions. So the SVD will weed out redundancies as described in the Theoretical Background section. First, we subtract the mean of each row from itself so we can standardize the variables and find the true variance between them. Then we use PSA and SVD as described in the Theoretical Analysis section, specifically, we use the `svd()` function on the transpose of our matrix divided by $\sqrt{n-1}$. Then we plot the singular values as a percent of the total energy, along with the first 3 modes of the position of the can. This again is done for all 4 videos, and these figures and their analysis can be seen under the Computational Results section.

4 Computational Results

First, looking at the results from Test 1, we go to Figure 1. We see that over 60% of the energy is in the first mode. And looking at the first mode, we see some very clear and pretty smooth oscillations. This is clearly the can moving up and down in the z direction over the duration of the recordings. Then looking at the second and third nodes that contain about 18% and 10% of the energy respectively, we don't really get anything interesting, it seems like these all come from noise generated from the slight shakiness of the cameras and possibly some failures of tracking the can correctly. This is what we expect since the can was only moving in the z direction in the video, so it makes sense the other modes would not contain very useful information.

Next, looking at the results from Test 2, we go to Figure 2. The first thing we notice is that in the graph of the singular values, the first singular value only contains just over 40% of the information with the second having over 20% and the third about 15%. I would assume this is because the recordings are so shaky, that we lose a lot of the information about movement in the z direction, and making it seem like there is movement in other directions, when there really is not. This is why we still see some pretty good oscillations in the first mode, but not nearly as smooth as the one from test one. This is because of that 'lost' information from the noise. Then, we still see a lot of meaningless spikes and movement in the second and third modes. This confirms that there was not a loss of information in the first mode because of actual movement of the can, but because of random movement of the cameras.

Next, looking at the results from Test 3, we go to Figure 3. For test three, the information is even more spread among the singular values than the first two tests. The first three singular values contain about 37%,

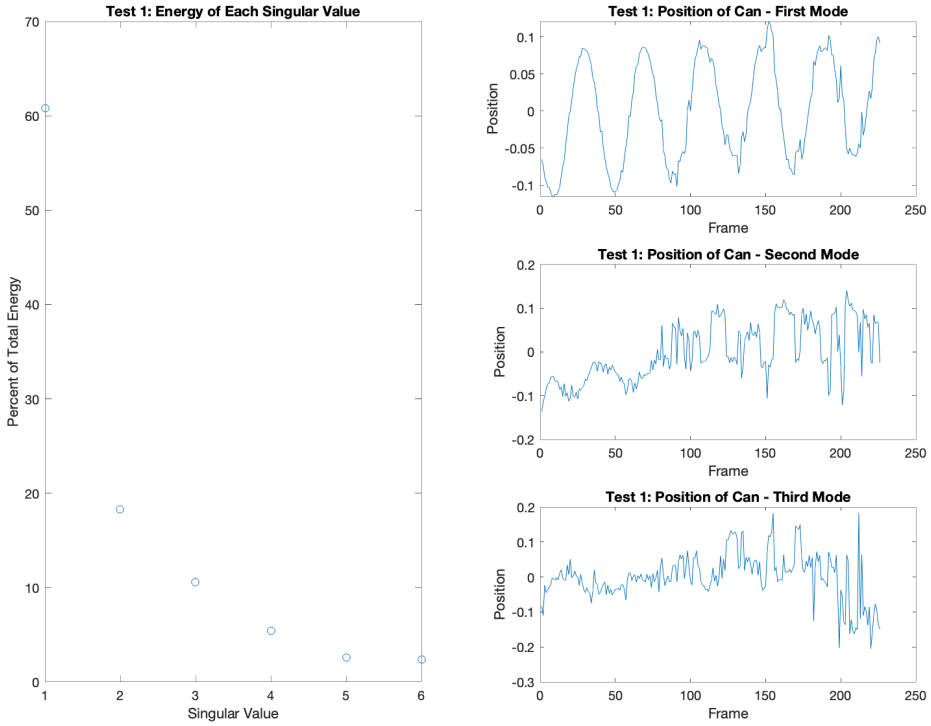

Figure 1: Here is the plot of the energy of each singular value as a percent of the total energy, along with the first three modes of the position of the can from Test 1.

25%, and 18% of the data respectively. Though this expected and hope for in this test since the can was let go from off-center, meaning there is movement in the x, y, and z axes, thus there should be less information in a single dimension. The plots of the first three modes confirms this. We see the first mode probably represents movement in the z axis since in the video most of the movement was up and down, very clear oscillations. Then the second and third modes we can arbitrarily assign the x and y axis respectively. We see that the second mode is much smoother and has less noise than the third mode. This is probably due to the cameras having better angles to capture the movement in that direction. Something else we notice is that the oscillations in the second mode are wider than those in the first mode. This represents the swinging back and forth happening taking longer than the moving up and down. Even with the noise in our plots, it is clear that the can had oscillations in all three directions.

Finally, looking at the results from Test 4, we go to Figure 4. The first thing we notice is that the energy is distributed similarly to that of Test 2, with the first three singular values containing about 41%, 18%, and 15% respectively. Since the movement of the can in the recording was like that of Test 3, we would have hoped that this would have been similar to Test 3, but it is not. This is because of the can spinning, which would not be a big deal except for the fact that we track the position of the can by a flashlight taped to the top. Thus, when the can spins so that the flashlight is not pointing to the camera, the coordinates of the can becomes inaccurate. Thus the spinning of the can causes the tracking of the can in the videos to be off, leading to 'lost' data. This is similar to how we 'lost' data in Test 2 because of the shakiness of the camera, thus we see a similar result in the plots. The first mode has clear oscillations in the z axis, but not as smooth as previous tests, due to the spinning of the can. While the second and third modes are full of hard to interpret noise. Since the can had movement in the x and y direction, we would have hoped to see oscillations here, but the data lost from spinning was too much and we were not able to retain that movement in those directions.

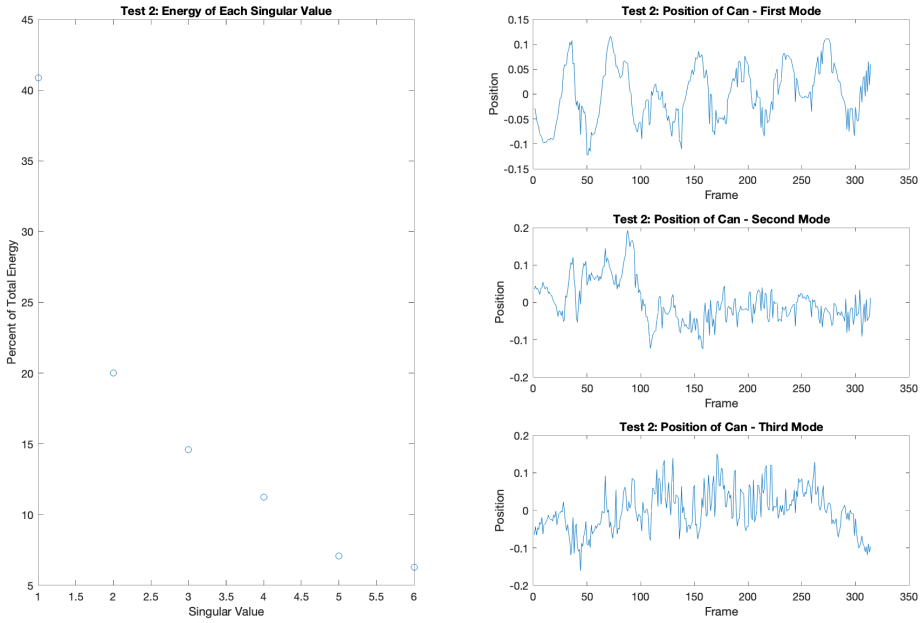

Figure 2: Here is the plot of the energy of each singular value as a percent of the total energy, along with the first three modes of the position of the can from Test 2.

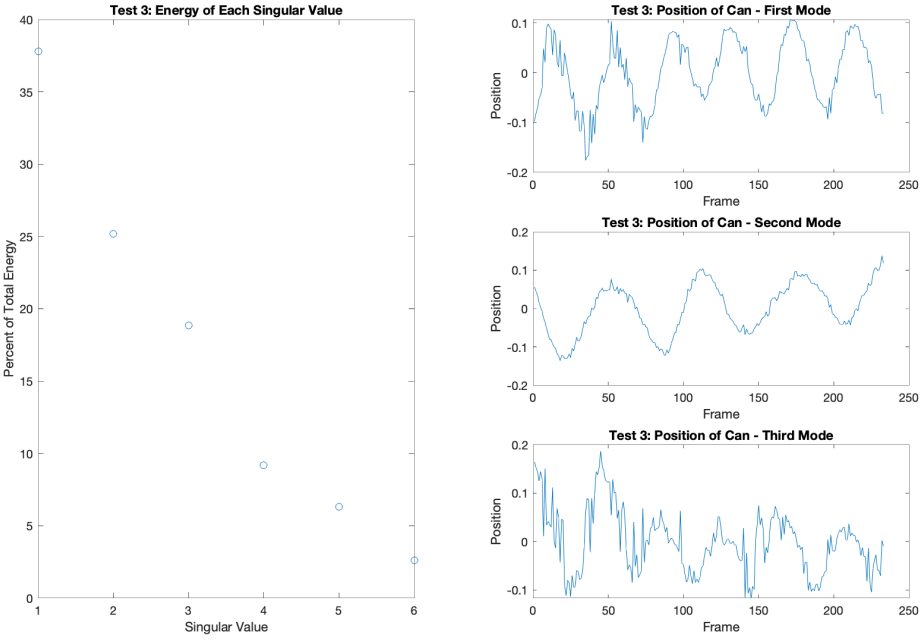

Figure 3: Here is the plot of the energy of each singular value as a percent of the total energy, along with the first three modes of the position of the can from Test 3.

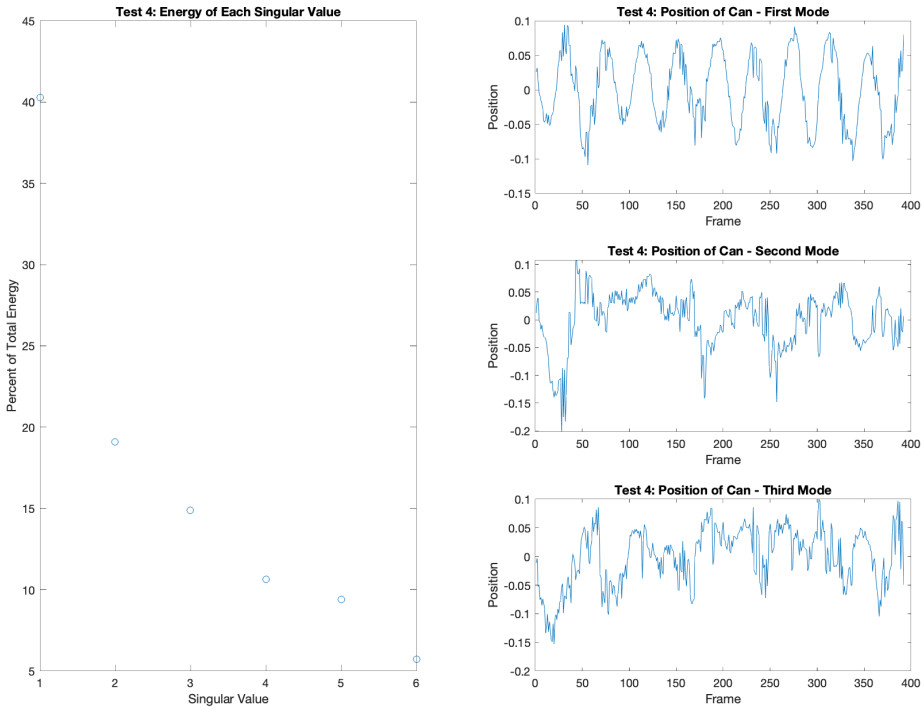

Figure 4: Here is the plot of the energy of each singular value as a percent of the total energy, along with the first three modes of the position of the can from Test 4.

5 Summary and Conclusions

We were able to use SVD and PCA in order track the movement of the can in all 4 tests. We see how effective PCA can be even in poor situations like Test 2. And in Test 4, the loss of information was not due to the PCA, but because of how we tracked the can in the videos. Thus, tracking the can a different way may have led to better results, such as tracking the can by its color. Though in every case, we were able to at least obtain obvious oscillations in the z axis.

Appendix A MATLAB Functions

- `svd(A,'econ')` returns the SVD of matrix `A`, $[U, \Sigma, V^*]$, without the rows of zeros padding the Σ matrix.

Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

```
%% Prep
clear all; close all; clc
load cam1_4.mat;
load cam2_4.mat;
load cam3_4.mat;
load cam1_3.mat;
load cam2_3.mat;
load cam3_3.mat;
load cam1_2.mat;
load cam2_2.mat;
load cam3_2.mat;
load cam1_1.mat;
load cam2_1.mat;
load cam3_1.mat;
%% Test 1 Setup
vidCrop1_1 = vidFrames1_1(220:390,315:385,:,:);
vidCrop2_1 = vidFrames2_1(100:320,260:340,:,12:237);
vidCrop3_1 = vidFrames3_1(245:310,275:440,:,1:226);
vid1_1x = zeros(1,226);
vid1_1y = zeros(1,226);
vid2_1x = zeros(1,226);
vid2_1y = zeros(1,226);
vid3_1x = zeros(1,226);
vid3_1y = zeros(1,226);
for j = 1:226
    bwFrame = rgb2gray(vidCrop1_1(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid1_1x(j), vid1_1y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop2_1(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid2_1x(j), vid2_1y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop3_1(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid3_1x(j), vid3_1y(j)] = ind2sub(size(bwFrame), ind);
end
%% Test 1 SVD
ses1 = [vid1_1x; vid1_1y; vid2_1x; vid2_1y; vid3_1x; vid3_1y];
[m,n] = size(ses1);
avg = mean(ses1,2);
ses1 = ses1-repmat(avg,1,n);
[U,S,V] = svd(ses1'/sqrt(n-1),'econ');
sig = diag(S);

figure(1)
subplot(1,2,1)
plot(100*sig/sum(sig),'O')
title("Test 1: Energy of Each Singular Value")
xlabel("Singular Value")
ylabel("Percent of Total Energy")
subplot(3,2,2)
plot(U(:,1))
title("Test 1: Position of Can - First Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,4)
plot(U(:,2))
title("Test 1: Position of Can - Second Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,6)
plot(U(:,3))
title("Test 1: Position of Can - Third Mode")
xlabel("Frame")
ylabel("Position")
```

```
%% Test 2 Setup
vidCrop1_2 = vidFrames1_2(225:400,300:400,:,:);
vidCrop2_2 = vidFrames2_2(65:350,200:375,:,26:341);
vidCrop3_2 = vidFrames3_2(200:325,275:450,:,6:319);
vid1_2x = zeros(1,314);
vid1_2y = zeros(1,314);
vid2_2x = zeros(1,314);
vid2_2y = zeros(1,314);
vid3_2x = zeros(1,314);
vid3_2y = zeros(1,314);
for j = 1:314
    bwFrame = rgb2gray(vidCrop1_2(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid1_2x(j), vid1_2y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop2_2(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid2_2x(j), vid2_2y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop3_2(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid3_2x(j), vid3_2y(j)] = ind2sub(size(bwFrame), ind);
end
%% Test 2 SVD
ses2 = [vid1_2x; vid1_2y; vid2_2x; vid2_2y; vid3_2x; vid3_2y];
[m,n] = size(ses2);
avg = mean(ses2,2);
ses2 = ses2-repmat(avg,1,n);
[U,S,V] = svd(ses2'/sqrt(n-1),'econ');
sig = diag(S);

figure(2)
subplot(1,2,1)
plot(100*sig/sum(sig),'O')
title("Test 2: Energy of Each Singular Value")
xlabel("Singular Value")
ylabel("Percent of Total Energy")
subplot(3,2,2)
plot(U(:,1))
title("Test 2: Position of Can - First Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,4)
plot(U(:,2))
title("Test 2: Position of Can - Second Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,6)
plot(U(:,3))
title("Test 2: Position of Can - Third Mode")
xlabel("Frame")
ylabel("Position")
%% Test 3 Setup
vidCrop1_3 = vidFrames1_3(225:380,280:380,:,7:239);
vidCrop2_3 = vidFrames2_3(135:350,210:390,:,32:264);
vidCrop3_3 = vidFrames3_3(200:325,275:400,:,1:233);
vid1_3x = zeros(1,233);
vid1_3y = zeros(1,233);
vid2_3x = zeros(1,233);
vid2_3y = zeros(1,233);
vid3_3x = zeros(1,233);
vid3_3y = zeros(1,233);
for j = 1:233
    bwFrame = rgb2gray(vidCrop1_3(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid1_3x(j), vid1_3y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop2_3(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid2_3x(j), vid2_3y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop3_3(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid3_3x(j), vid3_3y(j)] = ind2sub(size(bwFrame), ind);
end
```

```
%% Test 3 SVD
ses3 = [vid1_3x; vid1_3y; vid2_3x; vid2_3y; vid3_3x; vid3_3y];
[m,n] = size(ses3);
avg = mean(ses3,2);
ses3 = ses3-repmat(avg,1,n);
[U,S,V] = svd(ses3'/sqrt(n-1),'econ');
sig = diag(S);

figure(3)
subplot(1,2,1)
plot(100*sig/sum(sig),'O')
title("Test 3: Energy of Each Singular Value")
xlabel("Singular Value")
ylabel("Percent of Total Energy")
subplot(3,2,2)
plot(U(:,1))
title("Test 3: Position of Can - First Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,4)
plot(U(:,2))
title("Test 3: Position of Can - Second Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,6)
plot(U(:,3))
title("Test 3: Position of Can - Third Mode")
xlabel("Frame")
ylabel("Position")
%% Test 4 Setup
vidCrop1_4 = vidFrames1_4(225:360,320:450,:,:);
vidCrop2_4 = vidFrames2_4(95:310,220:410,:,6:397);
vidCrop3_4 = vidFrames3_4(175:275,300:460,:,1:392);
vid1_4x = zeros(1,392);
vid1_4y = zeros(1,392);
vid2_4x = zeros(1,392);
vid2_4y = zeros(1,392);
vid3_4x = zeros(1,392);
vid3_4y = zeros(1,392);
for j = 1:392
    bwFrame = rgb2gray(vidCrop1_4(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid1_4x(j), vid1_4y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop2_4(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid2_4x(j), vid2_4y(j)] = ind2sub(size(bwFrame), ind);

    bwFrame = rgb2gray(vidCrop3_4(:,:,:,j));
    [M, ind] = max(bwFrame(:));
    [vid3_4x(j), vid3_4y(j)] = ind2sub(size(bwFrame), ind);
end
```

```
%% Test 4 SVD
ses4 = [vid1_4x; vid1_4y; vid2_4x; vid2_4y; vid3_4x; vid3_4y];
[m,n] = size(ses4);
avg = mean(ses4,2);
ses4 = ses4-repmat(avg,1,n);
[U,S,V] = svd(ses4'/sqrt(n-1),'econ');
sig = diag(S);

figure(4)
subplot(1,2,1)
plot(100*sig/sum(sig),'O')
title("Test 4: Energy of Each Singular Value")
xlabel("Singular Value")
ylabel("Percent of Total Energy")
subplot(3,2,2)
plot(U(:,1))
title("Test 4: Position of Can - First Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,4)
plot(U(:,2))
title("Test 4: Position of Can - Second Mode")
xlabel("Frame")
ylabel("Position")
subplot(3,2,6)
plot(U(:,3))
title("Test 4: Position of Can - Third Mode")
xlabel("Frame")
ylabel("Position")
```