**Homework 2: Principal Component Analysis – Applied to Video**

Daniel Nguyen

February 5, 2018

**Abstract**

We apply principal component analysis (PCA) on a set of videos depicting a paint can oscillating in various ways. We explore how factors such as noise (shakiness of the camera) and the type of motion done by the paint can (pendulum vs. spring behavior) affect how the PCA performs.

**Contents**

**1 Introduction**

**2 Theory**

**3 MATLAB Procedure**

**4 Results**

**5 Summary and Conclusions**

**Appendix A MATLAB Functions Used and Extra Graphs**

**Appendix B Additional Figures**

**Appendix C MATLAB Code**

**1 Introduction**

The field of data analysis has led to many groundbreaking discoveries about the world, such as the successful search of the cystic fibrosis gene back in 1989. In order to make such discoveries, we have to have a way to find the hidden patterns and relationships in sometimes very large sets of data, which aren’t always obvious to see in its original form. But if we can filter and reduce the noise and dimensionality of our datasets into 2-D or 3-D subspaces, it would be a huge boon in our quest to find some key insights into our data. Principal Component Analysis allows us to do this, and we will explore its effect on a set of videos depicting an oscillating paint can.

**2 Theory**

The idea behind PCA is that it analyzes the variance within one set of data and the covariance between multiple set of data in order to compute the unitary matrices and the singular values that we get using Singular Value Decomposition. As mentioned in my last write-up, we find the covariance matrix with the following algorithm:

The covariance matrix looks as such:

Where the diagonal represents the variances for various individual sets measurements, and the off-diagonals contain the covariance between each different individual set of measurements. A useful quality of this matrix is that it is square and symmetrical. These properties will be very useful when it comes to the diagonalization of this matrix using eigenvalue decomposition. Diagonalization is useful because it gives us a low-rank approximation of the data, after removing all redundancies. The direct SVD onto a data matrix will give us the same thing, so for this report, I will be using that instead of computing the covariance.

**3 MATLAB Procedure**

We are given a set of twelve videos taken by three different cameras placed at different angles. Each camera records four cases:

1. An ideal case, where there is minimal noise and the paint can undergoes simple oscillatory behavior in the z-direction
2. A noisy case, in which shakiness is applied to the cameras
3. Horizontal displacement, where the paint can moves like a pendulum across the x-y plane in addition to oscillation in the z-direction
4. Case 3 but with the added motion of rotation around the paint can’s axis.

In order to analyze the motion of the paint can through the use of PCA, the position of the paint can in each video have to be extracted, so an algorithm must be contrived to do so. My algorithm to get the motion of the paint can in one individual video works as follows:

1. Convert the video into grayscale so that we don’t have to worry about the red, green and blue components of the video (which are ultimately unimportant since all we need is the paint can position), and we get the added benefit of going from a 4-dimensional data structure to a 3-dimensional one.
2. For every frame in the video, filter out every pixel below a certain grayscale value. Since the paint can is whiter than everything else in the video, if we set every pixel darker than the paint can to have a grayscale value of 0, it will help us a lot in the following calculations. The standard threshold I chose was 200, but for some videos I had to increase this value to between 230 to 240, since other objects such as the whiteboard or the wall have grayscale values above 200 as well, and these objects are not what we are interested in.

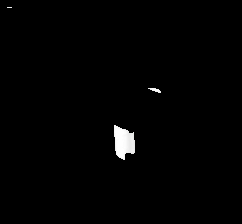
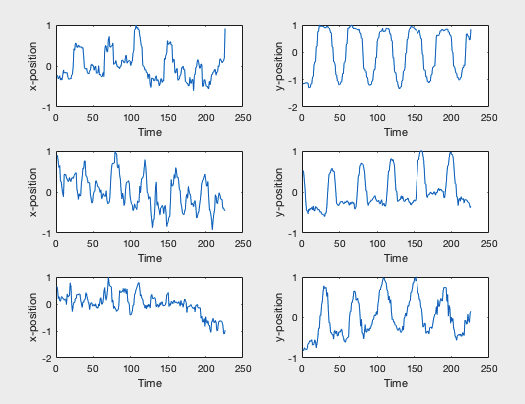


Figure 1: What a video looks like after the background objects are filtered out.

1. Now for each frame, we have to find the average pixel range of the non-filtered graphics in the video. I did this by going through the columns of each frame and returning the index of a non-zero value into an array, and then taking the mean of the indices to get the average vertical position of the paint can. I repeat this same process for each row to get the average horizontal position of the paint can. Since the mass of non-filtered pixels are changing position with each frame, the average row and column index of the paint can will change with time, so I can take the average values that I calculated for each frame to represent the x-y position of the paint can in time. (Note: MATLAB will return NaN for columns that don’t have any non-zero values since it won’t have any indexes to return, so for any NaN values encountered, I set it to zero in order to make calculation possible.) Concatenating our x-y values will ultimately result in a 2xN position matrix.
2. Before we can do any sort of accurate analysis on our extracted data, we will have to standardize our set. The range of pixel motion varies according to each video, so what I did was that I calculated the mean of each x or y vector, and then I subtracted that mean from every value in the vector, in order to center my position vectors around zero. But this doesn’t solve the problem that the maximum amplitudes of each vector varies with the video as well, so I divided all values in a given vector by its maximum value, which now represents position as a decimal between -1 to 1, which will be my standard metric for measuring position.

When we apply this algorithm to any video, we will have rough data of the position of the paint can throughout the video. After doing this for all videos in a certain case, we can concatenate all of their resulting position matrices on top of one another in order to get a 6xN matrix (containing x and y measurements at three different angles). We can then apply the SVD in order to visualize the most dominant modes in the videos, and what they can tell us about the motion of the paint can.

**4 Results**

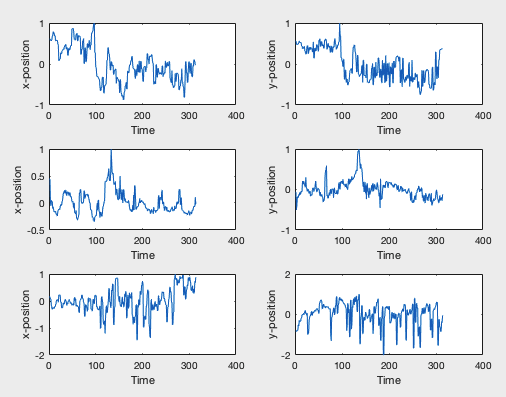
****

Case 1:

(Ideal)

Figure 2: x and y positions for videos 1\_1, 2\_1 and 3\_1 (ordered top to bottom).

When analyzing the y-positions, we get a very clean of the up-down oscillatory behavior displayed by the paint can due to the lack of noise in our videos. With the x-positions however, the graphs aren’t as clean. This is due to the slight noise of the videos (shakiness) and the very slight horizontal displacement of the paint can in the videos.

****

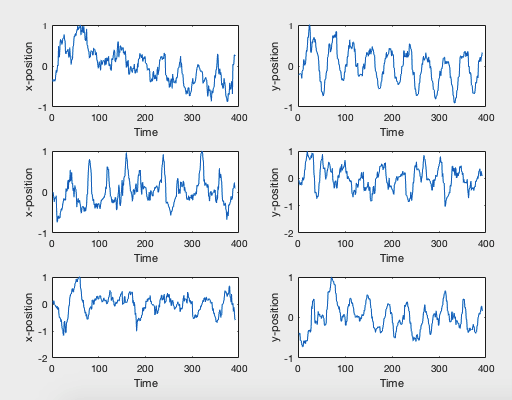
Case 2:

(Noisy)

Figure 3: x and y positions for videos of the second case (1\_2, 2\_2, 3\_2).

In this case, we can see the limitations of PCA when noise is a dominant factor in our data set. Though we can still see oscillatory behavior being exhibited in the y-position, there are a lot of spikes and the movement through time is very erratic.

(I will omit case 3 since I did not obtain any useful insight with my algorithm applied on it, but the graph and explanation for this case can be found in Appendix B.)

****

Case 4:

(Added

Pendulum &

Rotational

Behavior)

Figure 4: x and y positions for videos of the fourth case.

Once again, we get very clear oscillatory behavior given to us by the position we extracted from the paint can. Unlike case 1, if we look at the x-position of the paint can in each video this time, we will get a clearer sense of oscillatory behavior as well as the y-position. This is very consistent with the paint can’s pendulum motion.

Here is what the first column of our V matrix tells us about the position of the paint can in each case over time:

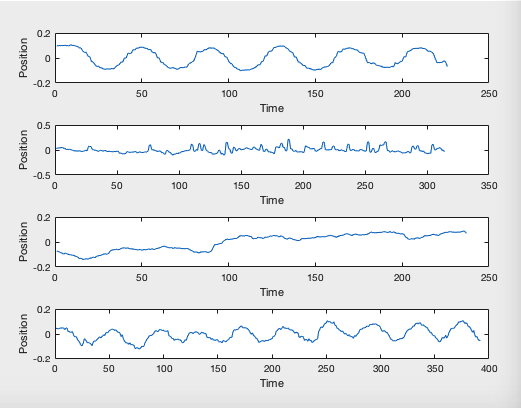
****

Figure 3: The visualizations of our paint cans given by the first mode, plotted using the first column of our V matrix.

For cases 1 and 4, we can see a relatively smooth low-rank representation of our paint can’s up-down motion throughout time. For cases 2 and 4, we are not as lucky. The raw position graphs that we extracted did not come out very cleanly, and so PCA had a very difficult time giving us an accurate representation of the actual motion of the paint can.

I should like to discuss the singular values that were found for each case now. In an ideal case, we should have gotten only one dominant singular value with all the other singular values being zero. This is to be expected if all of the x and y measurements of our data matrix were strongly correlated to each other (high covariance). But the covariance between the x and y measurements in our case were not as perfect as we wanted them to be, and so, though the additional singular values were less significant than the first, they were still high above zero. The graph of the singular values can be found in appendix A.

**5 Summary and Conclusions**

Principal Component Analysis is an invaluable tool as it is very effective in the analysis of datasets. We have successfully used it on our videos to simplify our data into a form that we can actually visualize in time. However, we won’t always be so fortunate, since PCA’s effectiveness is reduced when noise is added to our datasets, as we saw in video 2. In these cases, further investigation is needed in order to get what we want.

**Appendix A MATLAB Functions Used**

rgb2gray();

find();

isnan();

sum();

length();

mean();

min();

svd();

uint8();

**Note**: When working on this assignment, I couldn’t use the suptitle(); command anymore. This is what I used for the last assignment in order to label my subplots, but the command doesn’t exist now, even when I tried looking for help in the documentation.

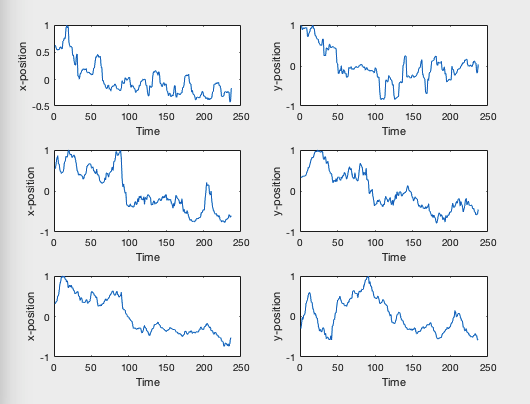
**Appendix B Additional Figures**

Figure B1: x and y positions for videos of the third case. Though we see oscillatory behavior exhibited for both the horizontal and vertical components, it is not consistent. We see a strange tendency for the range of motion of the paint can to drop down with time, as we see for all x and y graphs that the values are positive and then gradually slopes down to the negative values.

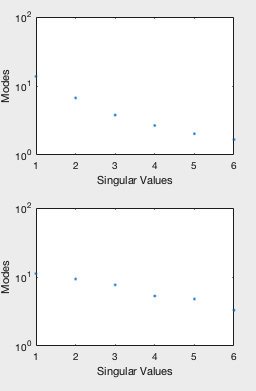
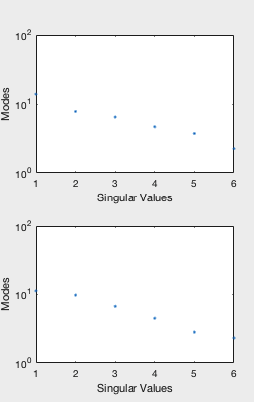


Figure B2: The singular values for each case (ordered top to bottom then left to right). Notice how high values for the less dominant modes are.

**Appendix C MATLAB Code**

%% Daniel Nguyen, AMATH 482

% Assignment 2

clear all;

close all;

clc;

load('cam1\_1.mat')

load('cam1\_2.mat')

load('cam1\_3.mat')

load('cam1\_4.mat')

load('cam2\_1.mat')

load('cam2\_2.mat')

load('cam2\_3.mat')

load('cam2\_4.mat')

load('cam3\_1.mat')

load('cam3\_2.mat')

load('cam3\_3.mat')

load('cam3\_4.mat')

%find a way to truncate all in loop later

%b = vidFrames1\_1(47:434,:,:,:);

%%

v1\_1 = [];

for i = 1:length(vidFrames1\_1(1,1,1,:))

v = rgb2gray(vidFrames1\_1(:,:,:,i));

v1\_1(:,:,i) = v;

end

v1\_1 = uint8(v1\_1);

for i = 1:length(v1\_1(1,1,:))

v = v1\_1(:,:,i);

v(v<200) = 0;

v1\_1(:,:,i) = v;

end

xy1\_1 = [];

for i = 1:length(v1\_1(1,1,:))

frame = v1\_1(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy1\_1 = [xy1\_1 xy];

end

%%

v2\_1 = [];

for i = 1:length(vidFrames2\_1(1,1,1,:))

v = rgb2gray(vidFrames2\_1(:,:,:,i));

v2\_1(:,:,i) = v;

end

v2\_1 = uint8(v2\_1);

for i = 1:length(v2\_1(1,1,:))

v = v2\_1(:,:,i);

v(v<225) = 0;

v2\_1(:,:,i) = v;

end

xy2\_1 = [];

for i = 1:length(v2\_1(1,1,:))

frame = v2\_1(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy2\_1 = [xy2\_1 xy];

end

%%

v3\_1 = [];

for i = 1:length(vidFrames3\_1(1,1,1,:))

v = rgb2gray(vidFrames3\_1(:,:,:,i));

v3\_1(:,:,i) = v;

end

v3\_1 = uint8(v3\_1);

for i = 1:length(v3\_1(1,1,:))

v = v3\_1(:,:,i);

v(v<225) = 0;

v3\_1(:,:,i) = v;

end

xy3\_1 = [];

for i = 1:length(v3\_1(1,1,:))

frame = v3\_1(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy3\_1 = [xy3\_1 xy];

end

%%

min\_len = min([length(xy1\_1(1,:)) length(xy2\_1(1,:)) length(xy3\_1(1,:))]);

xy1 = [xy1\_1(:,1:min\_len); xy2\_1(:,1:min\_len); xy3\_1(:,1:min\_len)];

xy1\_new = xy1;

for i = 1:length(xy1\_new(:,1))

mean\_val = mean(xy1\_new(i,:));

xy1\_new(i,:) = xy1\_new(i,:) - mean\_val;

max\_val = max(xy1\_new(i,:));

xy1\_new(i,:) = xy1\_new(i,:) / max\_val;

end

[u1,s1,v1] = svd(xy1\_new,'econ');

xy1\_recon = u1(:,1)\*s1(1,1)\*v1(:,1).';

%%

v1\_2 = [];

for i = 1:length(vidFrames1\_2(1,1,1,:))

v = rgb2gray(vidFrames1\_2(:,:,:,i));

v1\_2(:,:,i) = v;

end

v1\_2 = uint8(v1\_2);

for i = 1:length(v1\_2(1,1,:))

v = v1\_2(:,:,i);

v(v<220) = 0;

v1\_2(:,:,i) = v;

end

xy1\_2 = [];

for i = 1:length(v1\_2(1,1,:))

frame = v1\_2(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy1\_2 = [xy1\_2 xy];

end

%%

v2\_2 = [];

for i = 1:length(vidFrames2\_2(1,1,1,:))

v = rgb2gray(vidFrames2\_2(:,:,:,i));

v2\_2(:,:,i) = v;

end

v2\_2 = uint8(v2\_2);

for i = 1:length(v2\_2(1,1,:))

v = v2\_2(:,:,i);

v(v<240) = 0;

v2\_2(:,:,i) = v;

end

xy2\_2 = [];

for i = 1:length(v2\_2(1,1,:))

frame = v2\_2(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy2\_2 = [xy2\_2 xy];

end

%%

v3\_2 = [];

for i = 1:length(vidFrames3\_2(1,1,1,:))

v = rgb2gray(vidFrames3\_2(:,:,:,i));

v3\_2(:,:,i) = v;

end

v3\_2 = uint8(v3\_2);

for i = 1:length(v3\_2(1,1,:))

v = v3\_2(:,:,i);

v(v<240) = 0;

v3\_2(:,:,i) = v;

end

xy3\_2 = [];

for i = 1:length(v3\_2(1,1,:))

frame = v3\_2(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy3\_2 = [xy3\_2 xy];

end

%%

min\_len = min([length(xy1\_2(1,:)) length(xy2\_2(1,:)) length(xy3\_2(1,:))]);

xy2 = [xy1\_2(:,1:min\_len); xy2\_2(:,1:min\_len); xy3\_2(:,1:min\_len)];

xy2\_new = xy2;

for i = 1:length(xy2\_new(:,1))

mean\_val = mean(xy2\_new(i,:));

xy2\_new(i,:) = xy2\_new(i,:) - mean\_val;

max\_val = max(xy2\_new(i,:));

xy2\_new(i,:) = xy2\_new(i,:) / max\_val;

end

[u2,s2,v2] = svd(xy2\_new,'econ');

xy2\_recon = u2(:,1)\*s2(1,1)\*v2(:,1).';

%%

v1\_3 = [];

for i = 1:length(vidFrames1\_3(1,1,1,:))

v = rgb2gray(vidFrames1\_3(:,:,:,i));

v1\_3(:,:,i) = v;

end

v1\_3 = uint8(v1\_3);

for i = 1:length(v1\_3(1,1,:))

v = v1\_3(:,:,i);

v(v<235) = 0;

v1\_3(:,:,i) = v;

end

xy1\_3 = [];

for i = 1:length(v1\_3(1,1,:))

frame = v1\_3(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy1\_3 = [xy1\_3 xy];

end

%%

v2\_3 = [];

for i = 1:length(vidFrames2\_3(1,1,1,:))

v = rgb2gray(vidFrames2\_3(:,:,:,i));

v2\_3(:,:,i) = v;

end

v2\_3 = uint8(v2\_3);

for i = 1:length(v2\_3(1,1,:))

v = v2\_3(:,:,i);

v(v<240) = 0;

v2\_3(:,:,i) = v;

end

xy2\_3 = [];

for i = 1:length(v2\_3(1,1,:))

frame = v2\_3(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy2\_3 = [xy2\_3 xy];

end

%%

v3\_3 = [];

for i = 1:length(vidFrames3\_3(1,1,1,:))

v = rgb2gray(vidFrames3\_3(:,:,:,i));

v3\_3(:,:,i) = v;

end

v3\_3 = uint8(v3\_3);

for i = 1:length(v3\_3(1,1,:))

v = v3\_3(:,:,i);

v(v<235) = 0;

v3\_3(:,:,i) = v;

end

xy3\_3 = [];

for i = 1:length(v3\_3(1,1,:))

frame = v3\_3(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy3\_3 = [xy3\_3 xy];

end

%%

min\_len = min([length(xy1\_3(1,:)) length(xy2\_3(1,:)) length(xy3\_3(1,:))]);

xy3 = [xy1\_3(:,1:min\_len); xy2\_3(:,1:min\_len); xy3\_3(:,1:min\_len)];

xy3\_new = xy3;

for i = 1:length(xy3\_new(:,1))

mean\_val = mean(xy3\_new(i,:));

xy3\_new(i,:) = xy3\_new(i,:) - mean\_val;

max\_val = max(xy3\_new(i,:));

xy3\_new(i,:) = xy3\_new(i,:) / max\_val;

end

[u3,s3,v3] = svd(xy3\_new,'econ');

xy3\_recon = u3(:,1)\*s3(1,1)\*v3(:,1).';

%%

v1\_4 = [];

for i = 1:length(vidFrames1\_4(1,1,1,:))

v = rgb2gray(vidFrames1\_4(:,:,:,i));

v1\_4(:,:,i) = v;

end

v1\_4 = uint8(v1\_4);

for i = 1:length(v1\_4(1,1,:))

v = v1\_4(:,:,i);

v(v<200) = 0;

v1\_4(:,:,i) = v;

end

xy1\_4 = [];

for i = 1:length(v1\_4(1,1,:))

frame = v1\_4(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy1\_4 = [xy1\_4 xy];

end

%%

v2\_4 = [];

for i = 1:length(vidFrames2\_4(1,1,1,:))

v = rgb2gray(vidFrames2\_4(:,:,:,i));

v2\_4(:,:,i) = v;

end

v2\_4 = uint8(v2\_4);

for i = 1:length(v2\_4(1,1,:))

v = v2\_4(:,:,i);

v(v<240) = 0;

v2\_4(:,:,i) = v;

end

xy2\_4 = [];

for i = 1:length(v2\_4(1,1,:))

frame = v2\_4(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy2\_4 = [xy2\_4 xy];

end

%%

v3\_4 = [];

for i = 1:length(vidFrames3\_4(1,1,1,:))

v = rgb2gray(vidFrames3\_4(:,:,:,i));

v3\_4(:,:,i) = v;

end

v3\_4 = uint8(v3\_4);

for i = 1:length(v3\_4(1,1,:))

v = v3\_4(:,:,i);

v(v<200) = 0;

v3\_4(:,:,i) = v;

end

xy3\_4 = [];

for i = 1:length(v3\_4(1,1,:))

frame = v3\_4(:,:,i);

x = [];

for j = 1:length(frame(1,:))

indices\_x = find(frame(:,j), length(frame(:,1)), 'first');

avg\_index\_x = sum(indices\_x)/length(indices\_x);

x = [x avg\_index\_x];

end

x(isnan(x)) = 0;

x\_val = sum(x)/length(x);

y = [];

for k = 1:length(frame(:,1))

indices\_y = find(frame(k,:), length(frame(1,:)), 'first');

avg\_index\_y = sum(indices\_y)/length(indices\_y);

y = [y avg\_index\_y];

end

y(isnan(y)) = 0;

y\_val = sum(y)/length(y);

xy = [x\_val; y\_val];

xy3\_4 = [xy3\_4 xy];

end

%%

min\_len = min([length(xy1\_4(1,:)) length(xy2\_4(1,:)) length(xy3\_4(1,:))]);

xy4 = [xy1\_4(:,1:min\_len); xy2\_4(:,1:min\_len); xy3\_4(:,1:min\_len)];

xy4\_new = xy4;

for i = 1:length(xy4\_new(:,1))

mean\_val = mean(xy4\_new(i,:));

xy4\_new(i,:) = xy4\_new(i,:) - mean\_val;

max\_val = max(xy4\_new(i,:));

xy4\_new(i,:) = xy4\_new(i,:) / max\_val;

end

[u4,s4,v4] = svd(xy4\_new,'econ');

xy4\_recon = u4(:,1)\*s4(1,1)\*v4(:,1).';

%%

figure(1)

for i = 1:6

subplot(3,2,i)

plot(xy1\_new(i,:))

axis = '';

if mod(i, 2) == 1

axis = 'x-position';

else

axis = 'y-position';

end

xlabel('Time')

ylabel(axis)

end

figure(2)

for i = 1:6

subplot(3,2,i)

plot(xy2\_new(i,:))

axis = '';

if mod(i, 2) == 1

axis = 'x-position';

else

axis = 'y-position';

end

xlabel('Time')

ylabel(axis)

end

figure(3)

for i = 1:6

subplot(3,2,i)

plot(xy3\_new(i,:))

axis = '';

if mod(i, 2) == 1

axis = 'x-position';

else

axis = 'y-position'

end

xlabel('Time')

ylabel(axis)

end

figure(4)

for i = 1:6

subplot(3,2,i)

plot(xy4\_new(i,:))

axis = '';

if mod(i, 2) == 1

axis = 'x-position';

else

axis = 'y-position';

end

xlabel('Time')

ylabel(axis)

end

figure(5)

subplot(4,1,1)

plot(v1(:,1))

xlabel('Time')

ylabel('Position')

subplot(4,1,2)

plot(v2(:,1))

xlabel('Time')

ylabel('Position')

subplot(4,1,3)

plot(v3(:,1))

xlabel('Time')

ylabel('Position')

subplot(4,1,4)

plot(v4(:,1))

xlabel('Time')

ylabel('Position')

figure(6)

subplot(4,1,1)

semilogy(diag(s1),'.')

xlabel('Singular Values')

ylabel('Modes')

subplot(4,1,2)

semilogy(diag(s2),'.')

xlabel('Singular Values')

ylabel('Modes')

subplot(4,1,3)

semilogy(diag(s3),'.')

xlabel('Singular Values')

ylabel('Modes')

subplot(4,1,4)

semilogy(diag(s4),'.')

xlabel('Singular Values')

ylabel('Modes')