AMATH 482 - Winter Quarter
Homework 2: PCA

Vicky Norman

February 12, 2018

Abstract

This project uses principle component analysis to extract information about a series of dynamic systems from a several video recordings. The method and results are discussed in detail and some improvements to the data processing are discussed. The PCA successfully extracted the important dynamics from all cases, however it performed much better in the cases without noise.

1 Introduction and Overview

The purpose of this project is to use principle component analysis (PCA) to extract information from videos of a dynamic system. Three video recordings are taken of each system from three different locations. The data is processed using the PCA algorithm and the principle components are identified, this information is then used to reconstruct the dynamics of the system in the principle component basis.

The PCA algorithm has been implemented in Matlab to demonstrate how it can be used to extract the dynamics of a system. The process is repeated for four cases, the first case represents a system moving in simple harmonic motion without noise, the second represents the same system but includes noise. The third system introduces a second influential dynamic to the system in the form of horizontal displacement, and the fourth system has movement in the z-x plane and rotation in the y direction.

This report outlines the theoretical background behind the PCA algorithm and how it is implemented in Matlab. The results are presented with a series of figures and discussed in terms of how affective the PCA was on the data. Some ways of improving the process have been discussed in the conclusion.

2 Theoretical Background

Principle component analysis is an important procedure for determining the correlation between data sets. It uses an orthogonal transformation to convert a set of possibly correlated data to a set of linearly correlated variables called principle components. Before discussing how PCA works it is important to understand what is meant by 'correlated data'. Correlation describes the statistical relationship between variables, the correlation of variables is usually represented by a correlation coefficient between 0 and 1, the higher the coefficient the more correlated the data is.

Throughout this project we are going to be working with vectors of coordinates, we wish to determine the correlation between these sets of coordinates. The first stage in doing this is to represent them in matrix form as follows.

$$\mathbf{X} = \begin{vmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \\ \mathbf{x}_2 \\ \mathbf{y}_2 \\ \mathbf{x}_3 \\ \mathbf{y}_3 \end{vmatrix} \tag{1}$$

The six vectors of coordinates imply that we the system we are dealing with has six separate dynamics, however this is not necessarily the case. Since the data has been recorded about the same system but from a different viewpoint there will be some data that is redundant, i.e information that is recorded more than once. Our aim is to remove this redundancy and end up with as few vectors as possible that describe the dynamics of the system. To do this we must first understand the relationship between the data.

The easiest way to determine the strength of the relationship between two variables in coordinate systems is to calculate the covariance of the two vectors. Equations 2 and 3 show how these two values are calculated. When working with multiple coordinate vectors a covariance matrix can be produced which represents the covariance between each combination of coordinates and the variances of each vector. Equation 4 shows how this matrix is calculated with n pairs of coordinate vectors stored in matrix $\mathbf{X}$.

$$\sigma^2_{\mathbf{x}} = \frac{1}{n-1}\mathbf{x}\mathbf{x}^T \tag{2}$$

$$\sigma^2_{\mathbf{xy}} = \frac{1}{n-1}\mathbf{x}\mathbf{y}^T \tag{3}$$

$$C_{\mathbf{x}} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T = \begin{vmatrix} \sigma^2_{\mathbf{x}_1\mathbf{x}_1} & \sigma^2_{\mathbf{x}_1\mathbf{y}_1} & \sigma^2_{\mathbf{x}_1\mathbf{y}_2} & \cdots \\ \sigma^2_{\mathbf{x}_1\mathbf{y}_1} & \sigma^2_{\mathbf{y}_1\mathbf{y}_1} & & \\ \sigma^2_{\mathbf{x}_1\mathbf{x}_2} & & \ddots & \\ \vdots & & & \sigma^2_{\mathbf{y}_3\mathbf{y}_3} \end{vmatrix} \tag{4}$$

The values of the covariances in this matrix indicate how strongly related the sets of coordinates are, the higher the value the stronger the correlation is between the vectors. Our aim is to diagonalize this matrix so that the values are statistically independent, and thus the redundancy is removed. This is where the principle component basis comes in, we define a new basis $\mathbf{Y} = \mathbf{U}^*\mathbf{X}$ where $\mathbf{U}^*\mathbf{U} = \mathbf{I}$. The covariance matrix of $\mathbf{Y}$ is calculated as follows, using the SVD of matrix $\mathbf{X}$.

$$\begin{aligned} \mathbf{C}_{\mathbf{Y}} &= \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T \\ &= \frac{1}{n-1}\mathbf{U}^*\mathbf{X}(\mathbf{U}^*\mathbf{X})^T \\ &= \frac{1}{n-1}\mathbf{U}^*\mathbf{X}\mathbf{X}^T\mathbf{U} \\ &= \frac{1}{n-1}\mathbf{U}^*\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}\mathbf{U}^* \\ &= \frac{1}{n-1}\mathbf{\Sigma}^{\mathbf{2}}. \end{aligned} \tag{5}$$

Since $\mathbf{\Sigma}$ is a diagonal matrix, so is $\mathbf{\Sigma}^2$ which means we have succeeded in finding a diagonal covariance matrix. The ordered values of this diagonal matrix tell us which component has the strongest dynamic.//

3 Algorithm Implementation and Development

To implement the PCA algorithm in Matlab we must first extract the data we need from the videos. To track the position of the paint can it first needs to be located in each frame. Since the torch on top of the can is the brightest point in the video we can use this information to track it. Each video is truncated so that a small subset of the data is used to ensure that the possibility of error is reduced. Since the paint can is represented by a collection of points the median of these points is taken per frame and recorded in two vectors, $\mathbf{x}$ and $\mathbf{y}$. The median is used so that if there are any outliers they do not skew the results. Taking away the mean x and y value from each vector of coordinates helped to center all the coordinates so we could study the displacement of the can rather than the position in space. There are three videos per test, each providing two vectors of information. These vectors are collected together in a matrix $\mathbf{X}$ as in equation 1.

The SVD of this matrix is computed and the $\mathbf{\Sigma}$ value is used to calculate the covariance matrix of $\mathbf{Y}$ as in equation 5. Plotting these variance values shows us how many of the dynamics are important, the larger the variance the stronger the dynamic. Since we do not know the value of $\mathbf{U}*$ we can reconstruct the dynamics in the principle component basis as follows.

$$\begin{aligned} \mathbf{Y} &= \mathbf{U}^*\mathbf{X} \\ &= \mathbf{U}^*\mathbf{U}\mathbf{\Sigma}\mathbf{V}* \\ &= \mathbf{\Sigma}\mathbf{V}* \end{aligned} \tag{6}$$

To calculate the value of the first dynamic the first value of $\mathbf{\Sigma}$ will be used with the first column of $\mathbf{V}$ and so on for the other dynamics.

When implementing this in Matlab it became clear that it was important to have the videos well aligned in order to produce good results. Therefore the start times of the videos was adjusted so they all ran in time and in the same orientation. Plotting the displacement of the can in each direction helped to sync the videos and gave an idea of the dynamics of the system in each case.//

4 Computational Results

4.1 Test 1 - Ideal case

Figure 1 shows the displacement of the paint can in the Z direction over time for all three cameras. It can clearly be seen here that there is a relationship between the movement in all three sets of data, moreover one

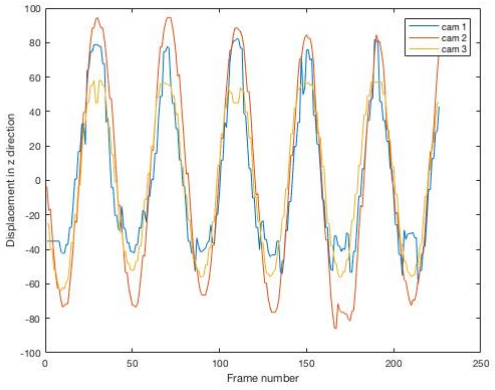

Figure 1: Displacement of paint can in the Z direction over time. Test 1.

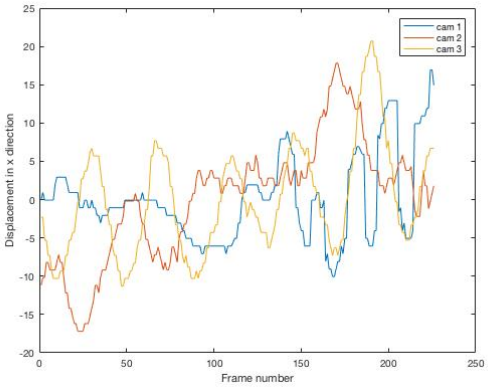

Figure 2: Displacement of paint can in the X direction over time. Test 1.

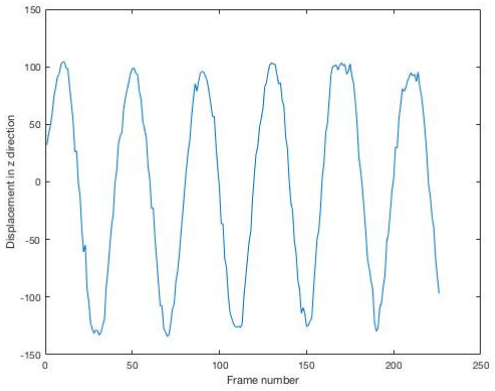

Figure 3: Displacement of paint can in the Z direction over time. Reconstructed in principle component basis. Test 1.

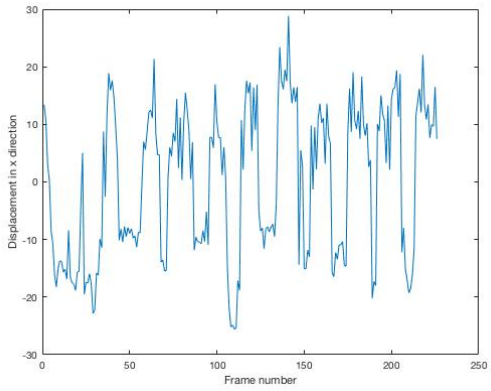

Figure 4: Displacement of paint can in the X direction over time. Reconstructed in principle component basis. Test 1.

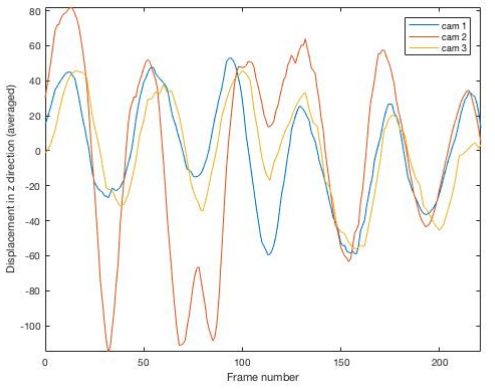

Figure 5: Displacement of paint can in the Z direction over time. Averaged using moving average filter. Test 2.

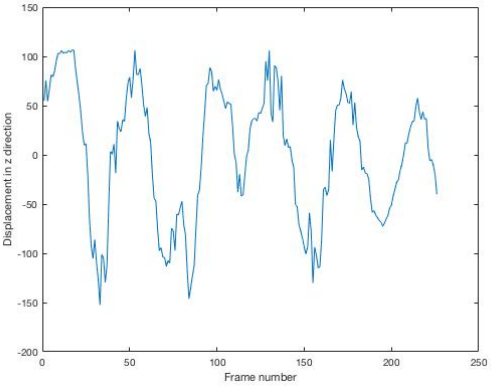

Figure 6: Displacement of paint can in the Z direction over time. Reconstructed in principle component basis. Test 2.

can see evidence of the simple harmonic oscillation of the paint can. Similarly figure 2 shows the displacement in the x direction, this movement is more erratic and also of a much smaller magnitude than that in the z direction, roughly 4 times less movement in the x direction can be observed. These figures strongly suggest that the most important description of the movement comes from measuring the displacement in the x direction, this hypothesis is confirmed by the results of PCA. Computing the covariance matrix in the principle component basis revealed that one of the variances was much larger than the other, suggesting that one of the components carries most of the data about the system, this result can be seen in figure 7. Thus reconstructing the displacement of the paint can using the strongest dynamic gives the result seen in figure 3, this is clearly showing the simple harmonic motion that we suspected from figure 1. Figure 4 also confirms what we suspected that the displacement in the x direction is a much weaker dynamic than that of the z direction.

4.2 Test 2 - Noisy case

We have shown how affective PCA can be at extracting the strongest dynamics from data that has a strong relationship, but what about if the data is full of noise? The set of videos for test 2 show the same movement of the paint can but they contain a lot of noise, mainly in the form of a shaky camera and movement in the

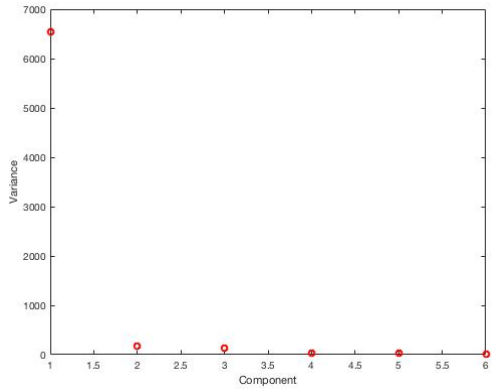

Figure 7: Ordered variance values of components. Test 1.

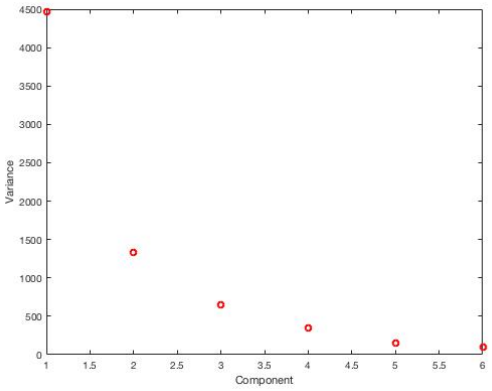

Figure 9: Ordered variance values of components. Test 2.

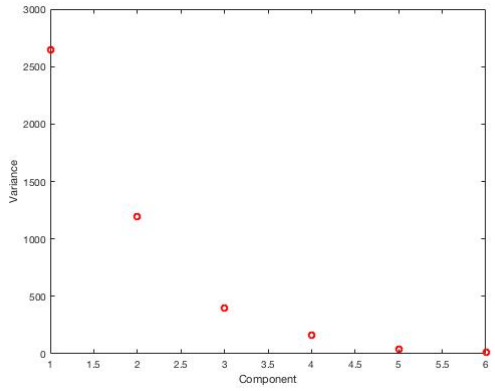

Figure 8: Ordered variance values of components. Test 3.

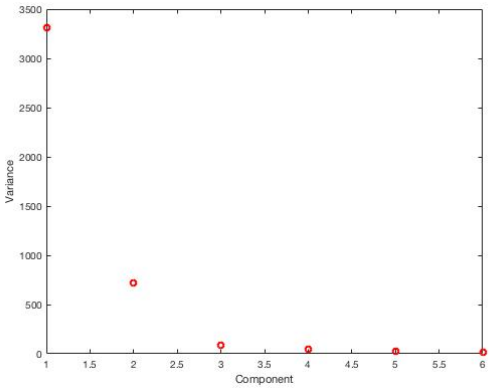

Figure 10: Ordered variance values of components. Test 4.

background. Extracting the displacement of the paint can in the Z direction did not produce the same clean result as in test 1 as seen in figure 5. Furthermore the result of the principle component reconstruction shown in figure 6 shows a similar pattern to that of test 1 but the signal is much less clear due to the amount of noise present. This is backed up by the plot of the variance values in figure 9 which shows that the recessive components are not as redundant as in test 1.

4.3 Test 3 - Horizontal case

The first two tests dealt with data that had one dominant dynamic, however systems often have more than one important dynamic. In case 3 the paint can also has some substantial movement in the x direction. The plot of variances shown in figure 8 shows the scale of the variances which implies that at least two of the dynamics are important for the system. The resulting graphs of the PCA are not included in the report here but can be obtained by running the code in appendix 5.3.

4.4 Test 4 - Horizontal displacement and rotation

PCA can be very helpful when analysing motion in more than two dimensions. Due to the nature of video, capturing morion in three dimensions is only possible by taking recordings in different locations. In test 4 the paint can has rotational movement in the y direction as well as the x-z plane. The variance of each component shown in figure 8 illustrates less of a difference between the values of each component. Reconstructing the

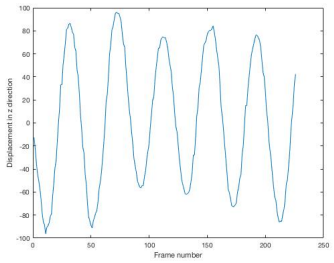

Figure 11: Displacement in Z direction in principle component basis over time. Test 4.

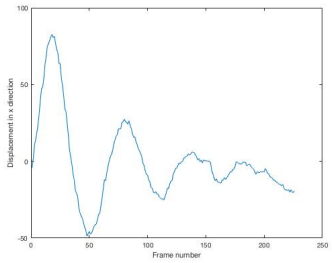

Figure 12: Displacement in X direction in principle component basis over time. Test 4.

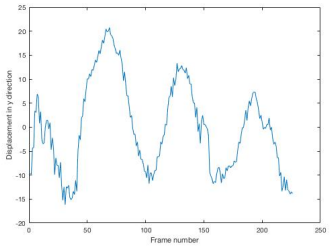

Figure 13: Displacement in Y direction in principle component basis over time. Test 4.

dynamics of the system using the principle component basis produces the results show in figures 11 - 13. These graphs do a good job of describing the motion of the paint can in this test, the strongest dynamic is still the motion in the x direction but there is definite displacement in the x and y directions.

5 Summary and Conclusion

Overall PCA was very effective in extracting the dynamics from the system. It worked best on when the data was not noisy and when the videos were well aligned. Aligning the videos so that they ran at the same time took quite a lot of time and effort so it could be useful to explore different signal processing techniques to ensure the data is in sync without having to do it manually.

The noise in test two negatively impacted the effectiveness of PCA, processing the video first to remove the noise could really help with the results of the PCA. Setting up the recording of the system to remove these obstacles would decrease the amount of computation required.

References

[1] *Computational Method for Data Analysis* Professor J. Nathan Kutz 2018.

[2] *mathworks.com*

Appendix

A

- [U,S,V] = svd(A, 'econ') - produces an economy-size decomposition of m-by-n matrix A, S containing the singular values.[2]
- y = filter(b,a,x) filters the input data x using a rational transfer function defined by the numerator and denominator coefficients b and a.[2]
- B = permute(A,order) rearranges the dimensions of A so that they are in the order specified by the vector order. B has the same values of A but the order of the subscripts needed to access any particular element is rearranged as specified by order. All the elements of order must be unique, real, positive, integer values.[2]
- eval(expression) evaluates the Matlab code represented by expression. If you use eval within an anonymous function, nested function, or function that contains a nested function, the evaluated expression cannot create a variable.[2]
- load(filename,variables) loads the specified variables from the MAT-file, filename.[2]
- k = find(X) returns a vector containing the linear indices of each nonzero element in array X.[2]

B

5.1 Test1 - Ideal case

```
clear all; close all;
% load videos
load cam1_1.mat vidFrames1_1
load cam2_1.mat vidFrames2_1
load cam3_1.mat vidFrames3_1

%%
num_frames = 226;
test = 1;
% initialising matrices
lightx = zeros(num_frames,1);
lighty = zeros(num_frames,1);

X = zeros(num_frames, 6);

for n = 1:3

cam = eval(sprintf('vidFrames%d_%d',n,test));
size(cam);

% truncating videos
if n == 1
    thresh = 250;
    ytrunk = 200:480;
    xtrunk = 180:450;
end
if n == 2
    thresh = 250;
    cam = cam(:,:,:,10:end);
    ytrunk = 100:400;
```

```
    xtrunk = 180:450;

end
if n == 3
    thresh = 230;
    cam = permute(cam,[2 1 3 4]);
    ytrunk = 210:460;
    xtrunk = 180:450;
end


for i = 1:num_frames
    [y x] = find(cam(ytrunk,xtrunk,2,i)>thresh);
    lightx(i) = median(x);
    lighty(i) = median(y);

%     imshow(cam(ytrunk,xtrunk,2,i)>thresh);
%     drawnow

end

% remove mean value of x and y to center around 0
lightx = lightx-mean(lightx);
lighty = lighty-mean(lighty);

X(:,2*n-1) = lightx;
X(:,2*n) = lighty;


end
figure

%plot z position
plot(1:num_frames, X(:,2), 1:num_frames, X(:,4), 1:num_frames, X(:,6))
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in z direction')

%plot x position
figure
plot(1:num_frames, X(:,1), 1:num_frames, X(:,3), 1:num_frames, X(:,5))
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in x direction')

%%

% svd on X
[U, S, V] = svd(X','econ');

covX = cov(X);
size(covX);

% calculte the covarance of PCA basis
```

```
covY = 1/(num_frames-1)*S.^2;

% plot covariance values
figure
plot(diag(covY),'ro', 'Linewidth', [2])
ylabel('Variance')
xlabel('Component')

% calculate first component
Y1 = S(1,1)*V(:,1);
figure
plot(Y1)
xlabel('Frame number')
ylabel('Displacement in z direction')

figure
% calculate second component
Y2 = S(2,2)*V(:,2);
plot(Y2)
xlabel('Frame number')
ylabel('Displacement in x direction')
```

5.2 Test 2 - Noisy case

```
clear all; close all;
% load videos
load cam1_2.mat vidFrames1_2
load cam2_2.mat vidFrames2_2
load cam3_2.mat vidFrames3_2


%%

num_frames = 226;
test = 2;
% initialising matrices
lightx = zeros(num_frames,1);
lighty = zeros(num_frames,1);

X = zeros(num_frames, 6);

for n = 1:3

cam = eval(sprintf('vidFrames%d_%d',n,test));
size(cam);

% truncating videos
if n == 1
    thresh = 250;
    ytrunk = 200:480;
    xtrunk = 180:450;
end
if n == 2
    thresh = 254;
```

```
    cam = cam(:,:,:,27:end);
    ytrunk = 70:400;
    xtrunk = 180:450;

end
if n == 3
    thresh = 245;
    cam = permute(cam,[2 1 3 4]);
    ytrunk = 250:460;
    xtrunk = 180:450;
end


for i = 1:num_frames
    [y x] = find(cam(ytrunk,xtrunk,2,i)>thresh);
    lightx(i) = median(x);
    lighty(i) = median(y);

%     imshow(cam(ytrunk,xtrunk,2,i)>thresh);
%     drawnow

end

% remove mean value of x and y to center around 0
lightx = lightx-mean(lightx);
lighty = lighty-mean(lighty);


X(:,2*n-1) = lightx;
X(:,2*n) = lighty;

end


%%

% average using filered peak average
coeff = ones(1,10)/10;
ave = filter(coeff, 1, X);
size(ave)
plot( 1-5:num_frames-5,[ave(:,2) ave(:,4) ave(:,6)])
axis([0 inf -inf inf])
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in z direction (averaged)')
figure
plot( 1-5:num_frames-5,[ave(:,1) ave(:,3) ave(:,5)])
axis([0 inf -inf inf])
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in x direction (averaged)')

%%
```

```
% svd on X
[U, S, V] = svd(X','econ');

covX = cov(X);
size(covX);

% calculte the covarance of PCA basis
covY = 1/(num_frames-1)*S.^2;

% plot covariance values
figure
plot(diag(covY),'ro', 'Linewidth', [2])
ylabel('Variance')
xlabel('Component')

% calculate first component
Y1 = S(1,1)*V(:,1);

figure
plot(Y1)
xlabel('Frame number')
ylabel('Displacement in z direction')

% calculate second component
Y2 = S(2,2)*V(:,2);

figure
plot(Y2)
xlabel('Frame number')
ylabel('Displacement in x direction')
```

5.3 Test 3 - Horizontal displacement

```
clear all; close all;
% load videos
load cam1_3.mat vidFrames1_3
load cam2_3.mat vidFrames2_3
load cam3_3.mat vidFrames3_3


%%

num_frames = 226;
test = 3;
% initialising matrices
lightx = zeros(num_frames,1);
lighty = zeros(num_frames,1);

X = zeros(num_frames, 6);

for n = 1:3

cam = eval(sprintf('vidFrames%d_%d',n,test));
size(cam);
```

```
% truncating videos
if n == 1
    thresh = 250;
    cam = cam(:,:,:,10:end);
    ytrunk = 200:480;
    xtrunk = 180:450;
end
if n == 2
    thresh = 254;
    cam = cam(:,:,:,:);
    ytrunk = 70:400;
    xtrunk = 180:450;

end
if n == 3
    thresh = 245;
    cam = permute(cam,[2 1 3 4]);
    cam = cam(:,:,:,5:end);
    ytrunk = 250:460;
    xtrunk = 180:450;
end


for i = 1:num_frames
    [y x] = find(cam(ytrunk,xtrunk,2,i)>thresh);
    lightx(i) = median(x);
    lighty(i) = median(y);

%     imshow(cam(ytrunk,xtrunk,2,i)>thresh);
%     drawnow

end

% remove mean value of x and y to center around 0
lightx = lightx-mean(lightx);
lighty = lighty-mean(lighty);

X(:,2*n-1) = lightx;
X(:,2*n) = lighty;


end


%%

% average using filered peak average
coeff = ones(1,10)/10;
ave = filter(coeff, 1, X);
plot( 1-5:num_frames-5,[ave(:,2) ave(:,4) ave(:,6)])
axis([0 inf -inf inf])
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
```

```
ylabel('Displacement in z direction (averaged)')

figure
plot( 1-5:num_frames-5,[ave(:,1) ave(:,3) ave(:,5)])
axis([0 inf -inf inf])
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in x direction (averaged)')



%%

% svd on X
[U, S, V] = svd(X','econ');

covX = cov(X);
size(covX);

% calculte the covarance of PCA basis
covY = 1/(num_frames-1)*S.^2

% plot covariance values
figure
plot(diag(covY),'ro', 'Linewidth', [2])
ylabel('Variance')
xlabel('Component')


% calculate first component
Y1 = S(1,1)*V(:,1);
figure;
plot(Y1)
xlabel('Frame number')
ylabel('Displacement in z direction')

figure;
% calculate second component
Y2 = S(2,2)*V(:,2);
plot(Y2)
xlabel('Frame number')
ylabel('Displacement in x direction')
```

5.4 Test 4 - Horizontal displacement and rotation

```
clear all; close all;
% load videos
load cam1_4.mat vidFrames1_4
load cam2_4.mat vidFrames2_4
load cam3_4.mat vidFrames3_4


%%
```

```
num_frames = 226;
test = 4;
% initialising matrices
lightx = zeros(num_frames,1);
lighty = zeros(num_frames,1);

X = zeros(num_frames, 6);

for n = 1:3

cam = eval(sprintf('vidFrames%d_%d',n,test));
size(cam);

% truncating videos
if n == 1
    thresh = 240;
    ytrunk = 200:480;
    xtrunk = 180:450;
end
if n == 2
    thresh = 245;
    cam = cam(:,:,:,8:end);
    ytrunk = 70:400;
    xtrunk = 180:450;

end
if n == 3
    thresh = 230;
    cam = permute(cam,[2 1 3 4]);
    ytrunk = 250:460;
    xtrunk = 100:350;
end


for i = 1:num_frames
    [y x] = find(cam(ytrunk, xtrunk,2,i)>thresh);
    lightx(i) = median(x);
    lighty(i) = median(y);

%     imshow(cam(ytrunk,xtrunk,2,i)>thresh);
%     drawnow

end

% remove mean value of x and y to center around 0
lightx = lightx-mean(lightx);
lighty = lighty-mean(lighty);

X(:,2*n-1) = lightx;
X(:,2*n) = lighty;


end
```

```
%%

% average using filered peak average
coeff = ones(1,10)/10;
ave = filter(coeff, 1, X);
plot( 1-5:num_frames-5,[ave(:,2) ave(:,4) ave(:,6)])
axis([0 inf -inf inf])
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in z direction (averaged)')

figure
plot( 1-5:num_frames-5,[ave(:,1) ave(:,3) ave(:,5)])
axis([0 inf -inf inf])
legend('cam 1','cam 2','cam 3')
xlabel('Frame number')
ylabel('Displacement in x direction (averaged)')



%%

% svd on X
[U, S, V] = svd(X','econ');

covX = cov(X);
size(covX);

% calculte the covarance of PCA basis
covY = 1/(num_frames-1)*S.^2;

% plot covariance values
figure
plot(diag(covY),'ro', 'Linewidth', [2])
ylabel('Variance')
xlabel('Component')


% calculate first component
Y1 = S(1,1)*V(:,1);
figure
plot(Y1)
xlabel('Frame number')
ylabel('Displacement in z direction')

% calculate second component
figure
Y2 = S(2,2)*V(:,2);
plot(Y2)
xlabel('Frame number')
ylabel('Displacement in x direction')

% calculate third component
```

```
figure
Y3 = S(3,3)*V(:,3);
plot(Y3)
xlabel('Frame number')
ylabel('Displacement in y direction')
```