AMATH 482 Assignment 3:

Principle Component Analysis

Yuqing Cui
February 24, 2021

Abstract

In this report, we are given twelve movie files which correspond to four cases (the ideal case, the noisy case, the case with horizontal displacement, and the case with both horizontal displacement and rotation), with each case described by three cameras recording at different positions. We used the principle component analysis (PCA) to analyze the behavior of the moving can by looking at movement of each component and the corresponding energy.

I. Introduction and Overview

We have four different cases and in each case, we are provided with three cameras at three different locations. The first one is the ideal case, in which the can is doing simple harmonic oscillations that it only has vertical displacement. The second case, the noisy case, is similar to the first case, except that our cameras are shaking, which would add noise into our data. In the third case, the can is released off-center so that besides vertical movement, it also has a displacement in the $x-y$ plane, which is similar to a pendulum. Based on the third case, we have our fourth case by adding rotation to the can so that it now has both displacement and rotation. We will do PCA, which will be more specifically introduced later, on each test. We will evaluate how accurate PCA performs on each test as well.

II. Theoretical Background

(a) Singular Value Decomposition (SVD)

The principle component analysis (PCA) is one of the significant applications of the singular value decomposition (SVD). SVD reforms an $m \times n$ matrix $\mathbf{A}$ into the following format:

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

in which $\mathbf{U}$ is an $m \times m$ unitary matrix that contains principle components. $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix that contains singular values. $\mathbf{V}$ is an $n \times n$ unitary matrix with time information. $\mathbf{U}$ and $\mathbf{V}$ are orthonormal to each other. All diagonal elements of $\mathbf{\Sigma}$ are non-negative and are sorted from the largest (upper-left) to the smallest (lower-right). The matrix multiplication will perform rotations and stretches. Besides, SVD enables every matrix to be diagonal with proper bases for the domain and the range.

(b) Principle Component Analysis (PCA)

The principle component analysis (PCA) assumes linearity. It filters redundancy in data and highlights signals with larger variances. Given an $m \times n$ input matrix $\mathbf{X}$ with m tests and n data in each test, we calculate the covariance matrix $\mathbf{C_X}$ by:

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^{\mathbf{T}}, \tag{2}$$

in which the diagonal elements are the variance of corresponding test, identifying significant components. The rest elements suggest the covariance between different tests, with a larger value indicating more redundancy in data. Therefore, our goal is to diagonalize the covariance matrix to extract the components, which is what can be performed by SVD. An important note is that when we do PCA and SVD, we must subtract the mean for each row and the PCA doesn't always give the best result.

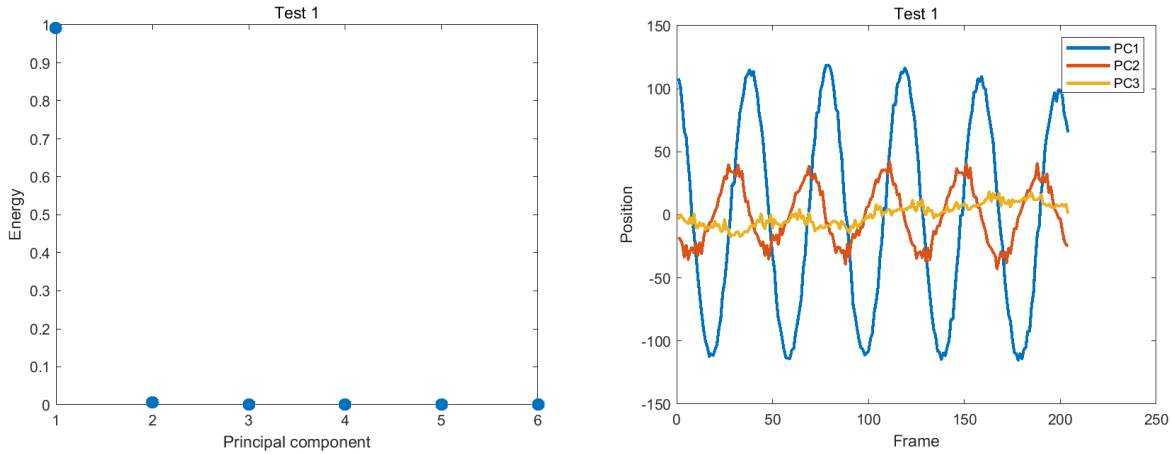
III. Algorithm Implementation and Development

The steps are the same for those four cases. We first loaded the data collected by each camera. Then we recorded the total number of frames. For each camera, we first played the video and briefly identified the range of the position of the can. Then we generated a filter to highlight those coordinates in order to make our results more accurate. Next, for each frame, we turned the video into grayscale to simplify the data so that we did not have to deal with complex RGB values. In grayscale, the pixel is more black if the value is closer to 0. So we looked for the pixel with a greater value, which indicates the white region on the can. We successfully captured the movement of the can by tracking its coordinates. We took the average of those coordinates, minimizing little shaking in movement, to better analyze the displacement. Once we finished these steps for all cameras, we had a problem that the three cameras didn't start recording at the same time. Our solution is finding when the y-coordinate first reached the maximum (since in the ideal case the can only moves vertically) and then took all the frames starting from it till the end.

We now obtained six vectors of two coordinates each recorded by three cameras. To form the matrix in order to apply SVD, again we adjust the length of those vectors to be equal to that of the shortest. We subtracted the mean of each row and performed the SVD on the covariance matrix and extracted its diagonal. We normalized the variances by dividing the square of the elements on the diagonal with their sums and we plotted the energy graph to see the percentage each principle component has. We also plotted the first three components, which are relatively significant than others, at each frame to analyze the motion.

IV. Computational Results

(a) Test 1: The Ideal Case



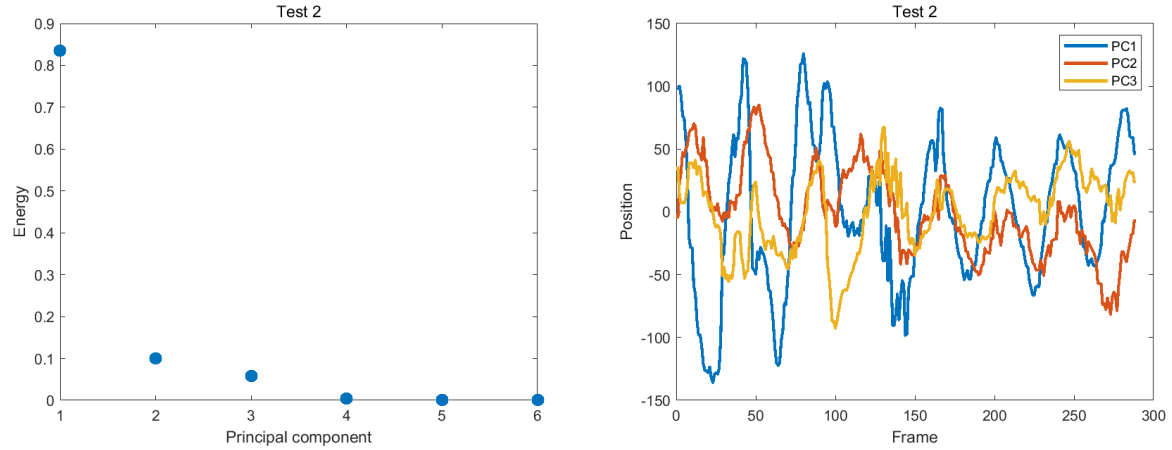
(a) Energy of principle components (b) Position of principle components

Figure 1. Result for test 1

Figure 1. (a) suggests that there is only one significant principle component that takes almost all the energy in the system. The result is plausible because in this case, we only have a simple harmonic motion, which means that the can only moves vertically. It's a 1-D motion, which is also proved by **Figure 1**. (b) that we can see an clean sinusoidal curve and obvious displacement of the first principle component which dominates the motion.

(b)Test 2: The Noisy Case

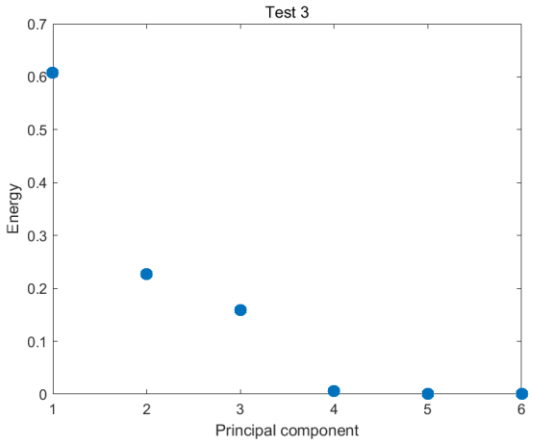
The shaking of cameras consequently become the noise in our data. **Figure 2**. (a) suggests that there is only one significant principle component in the system. However, the energy taken by the second and the third component are much greater than that in the ideal case. The curves in **Figure 2**. (b) are less smooth than those in **Figure 1**. (b) with the other two components are more amplified. Our PCA is roughly acceptable because of the energy graph, but it's not optimal in analyzing the displacement.



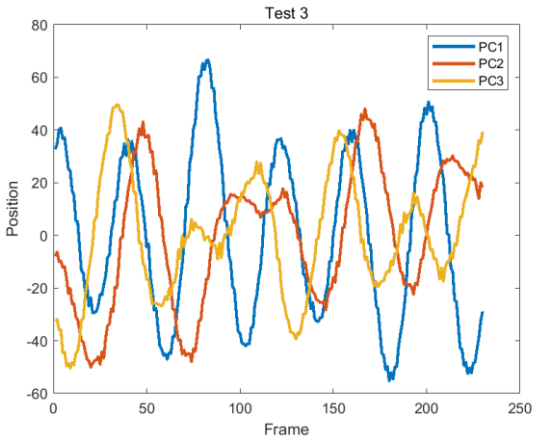
(a) Energy of principle components (b) Position of principle components

Figure 2. Result for test 2

(c) Test 3: Horizontal Displacement



(a) Energy of principle components

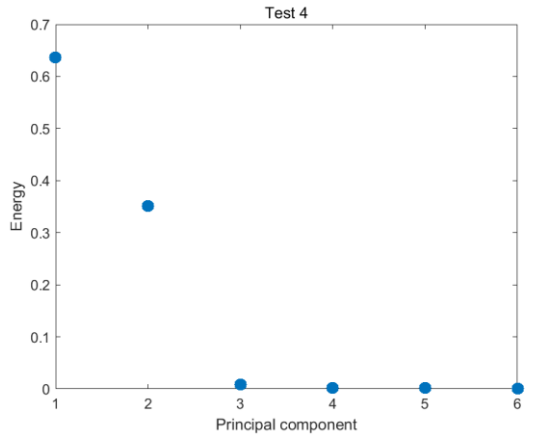


(b) Position of principle components

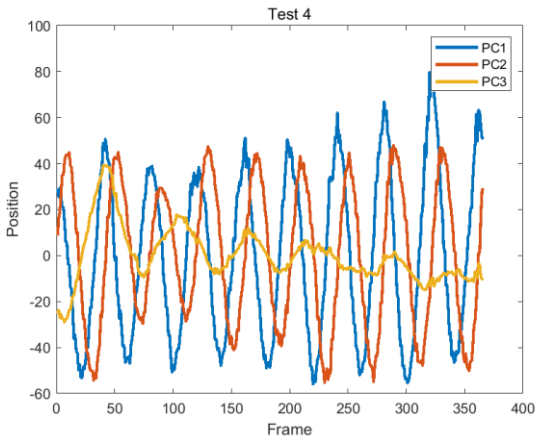
Figure 3. Result for test 3

This time, besides the simple harmonic motion in z-direction, we also have a pendulum motion in the $x-y$ plane because the can is released off-center. Therefore, we expect to have three major components that illustrating a motion in 3-D space. Our PCA did a good job in this case. We can see that in **Figure 3**. (a), the first three components take greater percent of energy, which is more obvious than the situation in case one and case 2, and the first component is the dominate one as usual. We have three relatively smooth sinusoidal curves in **Figure 3**. (b), which are reasonable for a can moving in all three directions.

(d) Test 4: Horizontal Displacement and Rotation



(a) Energy of principle components



(b) Position of principle components

Figure 4. Result for test 4

Finally, we added rotation to the can besides its 3-D displacement. Though we expect to see three or more principle components in this case, we only have two demonstrated by **Figure 4**. (a) that the first and the second components are the major ones. In **Figure 4**. (b), we can see that the curves of those two components are very obvious and swing more frequently than previous cases. The curve of the third component gradually decreases its amplitude as time goes. The PCA is not ideal in this case. One reason causing the problem might be that the rotation was accidentally cancelled with the movement in the $x - y$ plane when we were tracking the white area on the can.

V. Summary and Conclusions

We tracked the movement of the can in all four cases. We were able to perform PCA in each case to minimize redundancy in our data and extract the significant information, which allows us to analyze the behavior of each principle component including the percent of energy they took and their displacement. Though the last case illustrates that PCA does not always reflect the optimal result, it's still effective and useful in describing dynamics of a system in many cases.

Appendix A. MATLAB functions[1]

`load('filename'):` Loads data from the file `filename`. We used it to load our data.

`size(A,dim):` Returns the length of dimension `dim` when `dim` is a positive integer scalar. We used it to find the length of frames and number of columns of the matrices used for SVD.

`zeros(n1,n2)`: Returns an `n1`-by-`n2` array of zeros. We used it to create basic filters.

`rgb2gray(RGB)`: Converts the truecolor image RGB to the grayscale image. We used it to turn each frame into a gray-scaled picture to better capture the can.

`double(X)`: Converts the values in X to double precision. We used it to turn the grayscale value into doubles to apply our filter.

`[M,I] = max(X)`: Returns the maximum `M` and the corresponding index `I` of matrix `A`. We used it to find the maximum value in grayscale and the maximum index to cut our frames equally.

`[row,col] = find(__)`: Returns the row and column subscripts of each nonzero element in array X using any of the input arguments in previous syntaxes. We used it to track the position of the can that fulfills the condition.

`repmat(A,r1,r2)`: Specifies scalars `r1,r2`, that describe how copies of A are arranged in each dimension. We used find the mean of each row of the matrix.

`[u,s,v] = svd(A)`: Produces an economy-size decomposition of matrix `A,` such that `A = U*S*V'`. We used it to perform SVD.

`saveas(fig,filename):` saves the figure specified by `fig` to file `filename`. We used it to save out plots.

[1] Basic function descriptions from MathWorks Help Center.

Appendix B. MATLAB codes

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

numFrames11 = size(vidFrames1_1, 4);
numFrames21 = size(vidFrames2_1, 4);
numFrames31 = size(vidFrames3_1, 4);

%% Cam 1
% filter for mass
filter = zeros(480,640);
filter(170:430,300:350) = 1;

% track can
for i=1:numFrames11
    gray = rgb2gray(vidFrames1_1(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x11(i) = mean(X);
    y11(i) = mean(Y);
end

% cut frame
[a,b] = max(y11(1:30));
y11 = y11(b:end);
x11 = x11(b:end);

%% Cam 2
% filter for mass
filter = zeros(480,640);
filter(100:400,230:360) = 1;

% track can
for i=1:numFrames21
    gray = rgb2gray(vidFrames2_1(:,:,:,i));
    gray_f = double(gray).*filter;
```

```
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x21(i) = mean(X);
    y21(i) = mean(Y);
end

% cut frame
[a,b] = max(y21(1:30));
y21 = y21(b:end);
x21 = x21(b:end);

%% Cam 3
% filter for mass
filter = zeros(480,640);
filter(200:350,240:490) = 1;

% track can
for i=1:numFrames31
    gray = rgb2gray(vidFrames3_1(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x31(i) = mean(X);
    y31(i) = mean(Y);
end

% cut frame
[a,b] = max(y31(1:30));
y31 = y31(b:end);
x31 = x31(b:end);

%% PCA
minlength = min([length(x11),length(x21),length(x31)]);
x11 = x11(1:minlength);
x21 = x21(1:minlength);
x31 = x31(1:minlength);
y11 = y11(1:minlength);
y21 = y21(1:minlength);
y31 = y31(1:minlength);

% svd
```

```
X = [x11; y11; x21; y21; x31; y31];
X = X - repmat(mean(X,2),1,minlength);
[u,s,v] = svd(X*X'/sqrt(minlength-1));
lambda = diag(s);

figure()
plot(lambda.^2/sum(lambda.^2), '.','MarkerSize',30)
xlabel('Principal component'),ylabel('Energy')
title('Test 1')
saveas(gcf,'energy1.png')

Y = u'*X;
figure()
plot(Y(1,:),'Linewidth',2), hold on
plot(Y(2,:),'Linewidth',2)
plot(Y(3,:),'Linewidth',2)
xlabel('Frame'),ylabel('Position')
title('Test 1')
legend('PC1', 'PC2', 'PC3')
saveas(gcf,'position1.png')

%% Test 2
clear all; close all; clc;
load('cam1_2.mat');
load('cam2_2.mat');
load('cam3_2.mat');

numFrames12 = size(vidFrames1_2, 4);
numFrames22 = size(vidFrames2_2, 4);
numFrames32 = size(vidFrames3_2, 4);

%% Cam 1
% filter for mass
filter = zeros(480,640);
filter(170:430,300:400) = 1;

% track can
for i=1:numFrames12
    gray = rgb2gray(vidFrames1_2(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
```

```
    [X,Y] = find(gray_f > a*11/12);
    x12(i) = mean(X);
    y12(i) = mean(Y);
end

% cut frame
[a,b] = max(y12(1:30));
y12 = y12(b:end);
x12 = x12(b:end);

%% Cam 2
% filter for mass
filter = zeros(480,640);
filter(50:480,170:420) = 1;

% track can
for i=1:numFrames22
    gray = rgb2gray(vidFrames2_2(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x22(i) = mean(X);
    y22(i) = mean(Y);
end

% cut frame
[a,b] = max(y22(1:30));
y22 = y22(b:end);
x22 = x22(b:end);

%% Cam 3
% filter for mass
filter = zeros(480,640);
filter(200:380,240:490) = 1;

% track can
for i=1:numFrames32
    gray = rgb2gray(vidFrames3_2(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
```

```
    x32(i) = mean(X);
    y32(i) = mean(Y);
end

% cut frame
[a,b] = max(y32(1:30));
y32 = y32(b:end);
x32 = x32(b:end);

%% PCA
minlength = min([length(x12),length(x22),length(x32)]);
x12 = x12(1:minlength);
x22 = x22(1:minlength);
x32 = x32(1:minlength);
y12 = y12(1:minlength);
y22 = y22(1:minlength);
y32 = y32(1:minlength);

% svd
X = [x12; y12; x22; y22; x32; y32];
X = X - repmat(mean(X,2),1,minlength);
[u,s,v] = svd(X*X'/sqrt(minlength-1));
lambda = diag(s);

figure()
plot(lambda.^2/sum(lambda.^2), '.','MarkerSize',30)
xlabel('Principal component'),ylabel('Energy')
title('Test 2')
saveas(gcf,'energy2.png')

Y = u'*X;
figure()
plot(Y(1,:),'Linewidth',2), hold on
plot(Y(2,:),'Linewidth',2)
plot(Y(3,:),'Linewidth',2)
xlabel('Frame'),ylabel('Position')
title('Test 2')
legend('PC1', 'PC2', 'PC3')
saveas(gcf,'position2.png')

%% Test 3
```

```
clear all; close all; clc;
load('cam1_3.mat');
load('cam2_3.mat');
load('cam3_3.mat');

numFrames13 = size(vidFrames1_3, 4);
numFrames23 = size(vidFrames2_3, 4);
numFrames33 = size(vidFrames3_3, 4);

%% Cam 1
% filter for mass
filter = zeros(480,640);
filter(230:450,250:450) = 1;

% track can
for i=1:numFrames13
    gray = rgb2gray(vidFrames1_3(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x13(i) = mean(X);
    y13(i) = mean(Y);
end

% cut frame
[a,b] = max(y13(1:30));
y13 = y13(b:end);
x13 = x13(b:end);

%% Cam 2
% filter for mass
filter = zeros(480,640);
filter(100:420,170:420) = 1;

% track can
for i=1:numFrames23
    gray = rgb2gray(vidFrames2_3(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x23(i) = mean(X);
```

```
    y23(i) = mean(Y);
end

% cut frame
[a,b] = max(y23(1:30));
y23 = y23(b:end);
x23 = x23(b:end);

%% Cam 3
% filter for mass
filter = zeros(480,640);
filter(160:370,250:600) = 1;

% track can
for i=1:numFrames33
    gray = rgb2gray(vidFrames3_3(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x33(i) = mean(X);
    y33(i) = mean(Y);
end

% cut frame
[a,b] = max(y33(1:30));
y33 = y33(b:end);
x33 = x33(b:end);

%% PCA
minlength = min([length(x13),length(x23),length(x33)]);
x13 = x13(1:minlength);
x23 = x23(1:minlength);
x33 = x33(1:minlength);
y13 = y13(1:minlength);
y23 = y23(1:minlength);
y33 = y33(1:minlength);

% svd
X = [x13; y13; x23; y23; x33; y33];
X = X - repmat(mean(X,2),1,minlength);
[u,s,v] = svd(X*X'/sqrt(minlength-1));
```

```
lambda = diag(s);

figure()
plot(lambda.^2/sum(lambda.^2), '.','MarkerSize',30)
xlabel('Principal component'),ylabel('Energy')
title('Test 3')
saveas(gcf,'energy3.png')

Y = u'*X;
figure()
plot(Y(1,:),'Linewidth',2), hold on
plot(Y(2,:),'Linewidth',2)
plot(Y(3,:),'Linewidth',2)
xlabel('Frame'),ylabel('Position')
title('Test 3')
legend('PC1', 'PC2', 'PC3')
saveas(gcf,'position3.png')

%% Test 4
clear all; close all; clc;
load('cam1_4.mat');
load('cam2_4.mat');
load('cam3_4.mat');

numFrames14 = size(vidFrames1_4, 4);
numFrames24 = size(vidFrames2_4, 4);
numFrames34 = size(vidFrames3_4, 4);

%% Cam 1
% filter for mass
filter = zeros(480,640);
filter(220:450,270:470) = 1;

% track can
for i=1:numFrames14
    gray = rgb2gray(vidFrames1_4(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x14(i) = mean(X);
    y14(i) = mean(Y);
```

```
end

% cut frame
[a,b] = max(y14(1:30));
y14 = y14(b:end);
x14 = x14(b:end);

%% Cam 2
% filter for mass
filter = zeros(480,640);
filter(50:400,160:430) = 1;

% track can
for i=1:numFrames24
    gray = rgb2gray(vidFrames2_4(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x24(i) = mean(X);
    y24(i) = mean(Y);
end

% cut frame
[a,b] = max(y24(1:30));
y24 = y24(b:end);
x24 = x24(b:end);

%% Cam 3
% filter for mass
filter = zeros(480,640);
filter(100:300,270:500) = 1;

% track can
for i=1:numFrames34
    gray = rgb2gray(vidFrames3_4(:,:,:,i));
    gray_f = double(gray).*filter;
    [a,b] = max(gray_f(:));
    [X,Y] = find(gray_f > a*11/12);
    x34(i) = mean(X);
    y34(i) = mean(Y);
end
```

```
% cut frame
[a,b] = max(y34(1:30));
y34 = y34(b:end);
x34 = x34(b:end);

%% PCA
minlength = min([length(x14),length(x24),length(x34)]);
x14 = x14(1:minlength);
x24 = x24(1:minlength);
x34 = x34(1:minlength);
y14 = y14(1:minlength);
y24 = y24(1:minlength);
y34 = y34(1:minlength);

% svd
X = [x14; y14; x24; y24; x34; y34];
X = X - repmat(mean(X,2),1,minlength);
[u,s,v] = svd(X*X'/sqrt(minlength-1));
lambda = diag(s);

figure()
plot(lambda.^2/sum(lambda.^2), '.','MarkerSize',30)
xlabel('Principal component'),ylabel('Energy')
title('Test 4')
saveas(gcf,'energy4.png')

Y = u'*X;
figure()
plot(Y(1,:),'Linewidth',2), hold on
plot(Y(2,:),'Linewidth',2)
plot(Y(3,:),'Linewidth',2)
xlabel('Frame'),ylabel('Position')
title('Test 4')
legend('PC1', 'PC2', 'PC3')
saveas(gcf,'position4.png')
```