Principal Component Analysis

Yuxin Jiang

AMATH 482 HW3

Abstract

Principal Component Analysis (PCA) is one of the most effective tools in data analytics. In this report, we will illustrate a variety of different properties of PCA and different influence of noises by conducting an experiment on movie clips created from different cameras focusing on motions of a moving paint. The experiment mainly involves four different scenarios, the ideal case, the noisy case, the horizontal displacement case, and the horizontal displacement and rotation case. We will also examine the principal components with different energy levels and comparative results from the data with and without PCA to explore the usefulness.

I. Introduction and Overview

In this experiment, we are given data created by three different cameras in four different cases. The ideal case includes a moving paint that has a displacement in the z direction, which is clear, harmonic and perfect for dynamics extraction with PCA. The noisy case involves the same movements of the paint, but with the introduction of camera shaking and disrupted harmonic motion. In the horizontal displacement case, the paint is released off-center, causing not only movements in z direction, but also movements in x-y plane. In the horizontal displacement and rotation case, rotation is added to the previous horizontal case, which represents addition of rotation on the movements of z direction and x-y plane. In order to perform PCA, we tracked the motion of the paint in the first step. By looking at different cases, we will be able to explore different aspects of PCA and different effects of noises on the results.

II. Theoretical Background

A. Singular Value Decomposition (SVD)

A Singular Value Decomposition (SVD) is a factorization that converts a matrix of size $m \times n$ as following:

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

where $U \in \mathbb{C}^{m\times n}$, $V \in \mathbb{C}^{n\times n}$, $\Sigma \in \mathbb{R}^{m\times n}$. It is also assumed that the diagonal entries of Σ are nonnegative and ordered from largest to smallest so that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ where $p = \min(m, n)$.

With SVD, the matrix first applies a unitary transformation preserving the unit sphere with V^*. Then it is stretched by Σ that creates an ellipse with principal semiaxes. In the end, the generated hyperellipse is rotated by the unitary transformation U. Every matrix has an SVD and singular values σ_j are distinct, which enables us to diagonalize every matrix.

B. Principal Component Analysis (PCA)

One of the most effective tools of analysis utilizing SVD is the Principal Component Analysis (PCA). PCA is able to be used for quantification of low-dimensional dynamics, where the data is unknown. Matrices often contain redundant information that are repetitive and complex. With PCA, redundancy and noise can be reduced by extracting the dominant components which then allow us to observe the simplified behaviors.

PCA first requires us to standardize and normalize the data to put into the form

$$X = \begin{bmatrix} X_a \\ Y_a \\ X_b \\ Y_b \\ X_c \\ Y_c \end{bmatrix} \tag{2}$$

where the row columns represent different measurements. Covariance then can help us address the issues of noise and redundancy. We compute the covariance matrix as following:[1]

[1] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

$$C_X = \frac{1}{n-1} XX^T \quad (3)$$

It is a square and symmetric matrix whose diagonal represents the variance of particular measurements. The off-diagonal terms are the covariances between different measurements, which would be equal to diagonal terms if two data sets are identically redundant. The large off-diagonal terms thus catch the redundancy from correlations between the possible pairs of measurements while small off-diagonal terms indicate low redundancy. The large off-diagonal terms also indicate the dynamics of interest since it suggests strong fluctuations. We then would be able to remove the redundancy by utilizing SVD to diagonalize the covariance matrix. The matrix Σ shows us the energy and we could identify the largest one for simplified behavior. [2]

III. Algorithm Implementation and Development

For the ideal case, I first load the data and find the number of frames with the command **size**. In order to apply PCA, I first used a for loop to track the motions of the paint. I converted each frame into a grayscale using **rgb2gray** and **double** so that distractions in the video can be reduced. After examining the video and data, I set all of the areas that do not contain the object we wish to observe, the paint, to total black by setting the value to zero, which represents the darkest color in the mode. Since we are looking for the lightest location given that there is a light attached to the top of the paint, setting the parts we do not need to zero can help us locate the biggest value more accurately. Therefore, after masking the areas, I used **max** command to find the largest value. I then set a threshold of 92% of the biggest value to extract the location of top of the paint since the light seems to expand while moving, storing the data into separate vectors. I put the average locations into separate vectors in the end. I did the same for the all three cameras and the other three cases with adjustment in the area that I set to black and number of frames.

However, I have found that the videos had different number of frames and do not start at the same positions at the same time. Hence, I plotted the movements and observed their patterns. In the clip for camera 3, I plotted **y3** while plotting **x1** and **x2** for the other cameras since the video by camera 3 is rotated. After observing the patterns, I adjusted the data for better alignment for all four cases

[2] J.Nathan Kutz, *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*

by taking parts of specific data using the command **x2(n:end)**. In order to put all three sets of data into one matrix for PCA, I used the **min** command to find the smallest size of all vectors and then put the cut vectors according to the smallest length into a matrix in the form of equation (2) in Section II. I then subtracted the mean of each row found by the command **mean** so that the values are centered at zero. Subsequently, I took the economy-sized SVD of the matrix using **svd(A, 'econ')** and plotted the normalized energy stored in Σ. In the end, I plotted **v*s** that represents the locations of the principal components multiplied by corresponding energy. In order to compare the usefulness of PCA, I plotted the original movements as well. In this way, we are able to visualize clearly the dominant components and its corresponding movements. I applied the same strategies for the other three cases except for changing the number of frames and adjusting the length of vectors with different constants.

IV. Computational Results

A. Ideal Case

As we can see in top of figure 1, there is no obvious movements of the paint in x-y plane but obvious and simple oscillation in z direction. In figure 2, it is easy to observe that there is only one dominant component whose energy level is above 75% while the others are relatively small and considered as redundancies. We could see the movement of the dominant component showing the expected motion in bottom of figure 1, thus proving that PCA did a great job in this case.

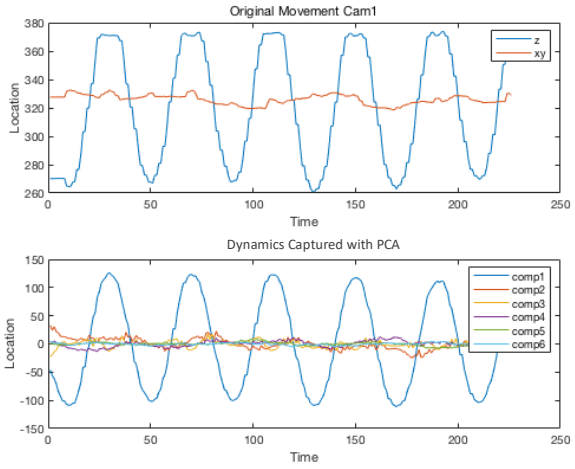

Figure 1: Plot of Original Movements in Camera 1 (top) vs Dynamics Captured with PCA in Ideal Case (bottom)

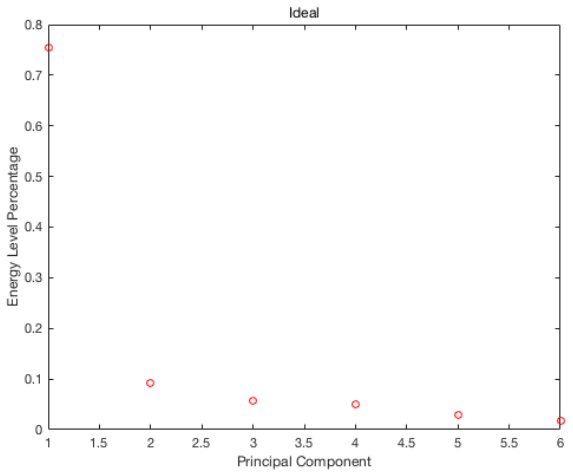

Figure 2: Plot of Principal Components in Ideal Case

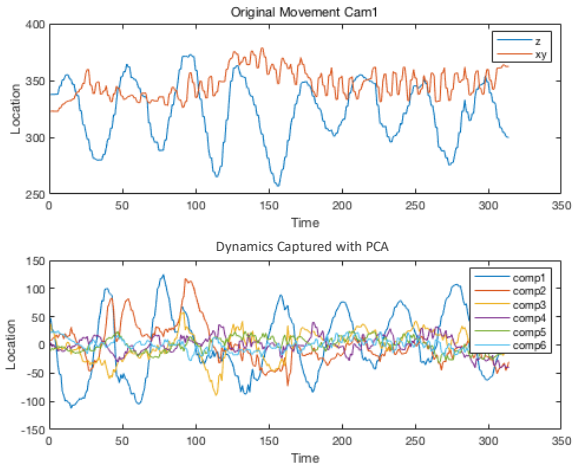

Figure 3: Plot of Original Movements in Camera 1 (top) vs Dynamics Captured with PCA in Noisy Case (bottom)

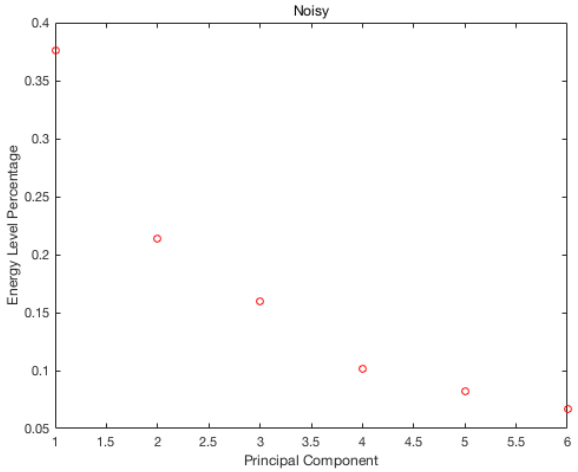

Figure 4: Plot of Principal Components in Noisy Case

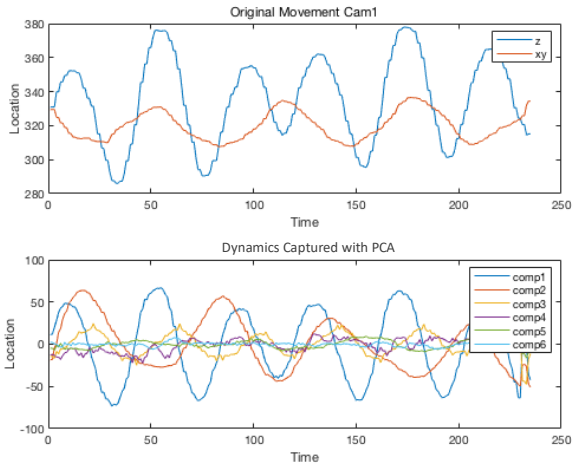

Figure 5: Plot of Original Movements in Camera 1 (top) vs Dynamics Captured with PCA in Horizontal Case (bottom)

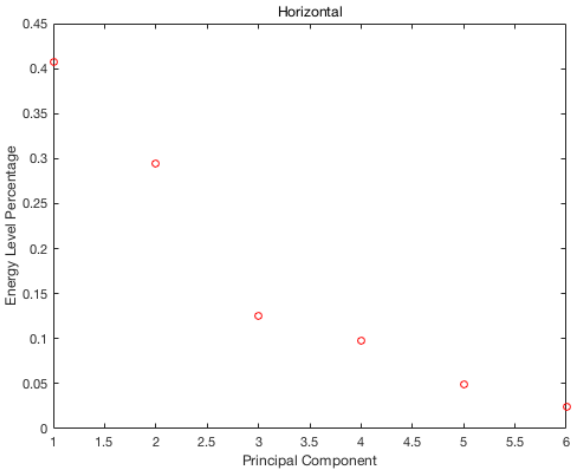

Figure 6: Plot of Principal Components in Horizontal Case

B. Noisy Case

Since the camera is shaking, lots of noises are introduced in this case. We can see increased movements in x-y plane as shown in top of figure 3 due to the camera shaking. Although there is still one dominant component, the energy level is much lower comparing to the ideal case, and energy levels of the other components are higher than the ideal case as shown in figure 4. Therefore, it takes around two to three components to capture most of the behaviors. The relatively significant energy level of component 2 also produces movements resulting from the noises as shown in bottom of figure 3. With the introduction of noise, PCA constantly regarded the noise as our targeted data and did not produce the expected results. However, we could still see the oscillating movements, but not as harmonic and clear.

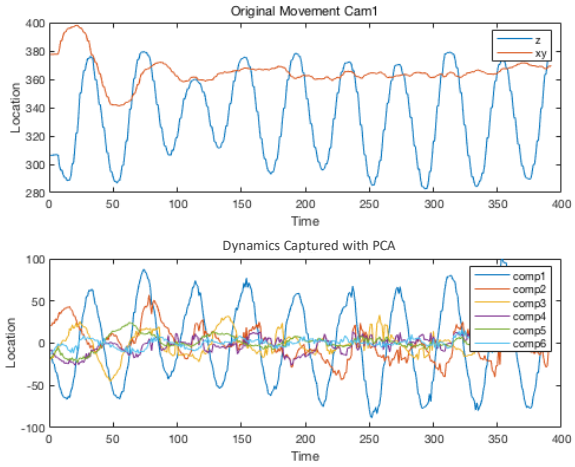

Figure 7: Plot of Original Movements in Camera 1 (top) vs Dynamics Captured with PCA in Horizontal and Rotational Case (bottom)

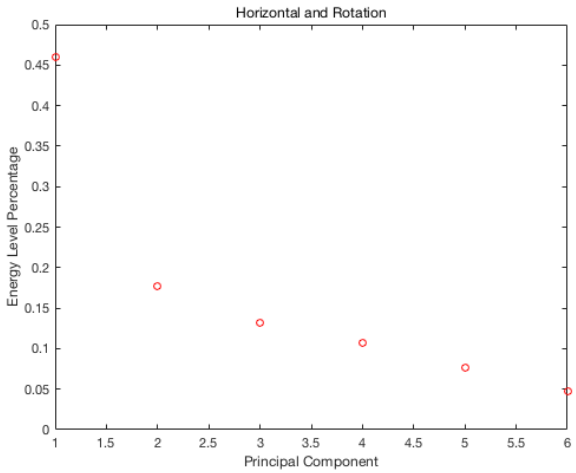

Figure 8: Plot of Principal Components in Horizontal and Rotational Case

C. Horizontal Displacement

In this case, we can see oscillating movements in both z direction and in x-y plane in top of figure 5 due to the pendulum motion and harmonic oscillation. From figure 6, we saw that the energy levels show a similar pattern as in the noisy case. There are two relatively dominant components and the others have relatively small energy percentage. The horizontal displacement results in addition to the movements in z direction, thus causing the results that there are two components showing obvious motions as shown in bottom of figure 5. The motions of the two components illustrate the expected patterns, and thus PCA did a better job in this case than in the noisy case.

D. Horizontal Displacement and Rotation

It is hard for us to capture any rotational behaviors through the plot since there is only harmonic oscillating motion showing in z direction in top of figure 7. From figure 8, there is one dominant component and the others are relatively smaller than in horizontal case. Only one of the components show a clear picture of oscillation while the others are not as clear as expected to show horizontal and rotational behaviors according to bottom of figure 7. Therefore, PCA did not offer us a clear picture in this case.

V. Summary and Conclusions

With this experiment, we could clearly see the practical usefulness of PCA and also factors that will potentially affect its accuracy. By comparing different cases, I found that PCA could effectively determine the dominant components and then track the behaviors associated without redundancy in most cases. The ideal case shows the most obvious results PCA can perform, removing the redundancy and tracking the motions clearly according to the dominant component. In the noisy case, however, accuracy and effectiveness are greatly harmed. According to that, we could conclude that noises have negative effect on the performance of PCA. In order to apply PCA, we will need to reduce the noises as many as possible to obtain greater results. The PCA perform well in the horizontal case, where both pendulum and oscillation behaviors are captured. However, weakness of PCA in processing rotational data has been shown in the last case. In conclusion, PCA provides us with an efficient way to remove the redundancy and reveal the dynamics by finding dominant components in cases where noises are not large and rotational behaviors are not highly involved.

Appendix A: MATLAB Functions Used and Brief Implementation Explanation

[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.

[U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix A. The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, S, along with the columns in either U or V that multiply those zeros in the expression A = U*S*V'

B = repmat(A,n) returns an array containing n copies of A in the row and column dimensions. The size of B is size(A)*n when A is a matrix.

double(s) converts the symbolic value s to double precision.

M = max (Y) returns the value of largest element in the vector.

M = mean(A) returns the mean of the elements of A along the first array dimension whose size does not equal 1.

plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

rgb2gray(RGB) converts the true-color image RGB to the grayscale image.[3]

[3] "Makers of MATLAB and Simulink." MathWorks, www.mathworks.com/

Appendix B: MATLAB Code

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

f1=size(vidFrames1_1,4);
f2=size(vidFrames2_1,4);
f3=size(vidFrames3_1,4);

%track the motion
x1=[];
y1=[];
for i=1:f1
    x=double(rgb2gray(vidFrames1_1(:,:,:,i)));
    x(400:end,:)=0;
    x(1:210,:)=0;
    x(:,400:end)=0;
    x(:,1:300)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y1(i)=mean(b);
    x1(i)=mean(a);
end

x2=[];
y2=[];
for i=1:f2
    x=double(rgb2gray(vidFrames2_1(:,:,:,i)));
    x(1:110,:)=0;
    x(370:end,:)=0;
    x(:,350:end)=0;
    x(:,1:200)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y2(i)=mean(b);
    x2(i)=mean(a);
end

x3=[];
y3=[];
for i=1:f3
    x=double(rgb2gray(vidFrames3_1(:,:,:,i)));
    x(1:250,:)=0;
    x(350:end,:)=0;
    x(:,500:end)=0;
    x(:,1:200)=0;
    M=max(x(:));
    [a,b]=find(x>=M*0.92);
    y3(i)=mean(b);
    x3(i)=mean(a);
end
x2=x2(10:end);
y2=y2(10:end);%aligh the second one
```

```
%plot(x1);hold on
%plot(x2);
%plot(y3)

%PCA

l=min([length(x1),length(x2),length(y3)]);
A=[x1(1:l);y1(1:l);x2(1:l);y2(1:l);x3(1:l);y3(1:l)];

[m,n]=size(A);
mn=mean(A,2);
A=A-repmat(mn,1,n);

[u,s,v]=svd(A,'econ');
figure(1)
plot(diag(s)/sum(diag(s)),'ro')
xlabel('Principal Component');
ylabel('Energy Level Percentage');
title('Ideal');

figure(2)
subplot(2,1,1)
plot(1:l,x1(1:l), 1:l, y1(1:l))
legend('z','xy')
xlabel('Time')
ylabel('Location')
title('Original Movement Cam1')
subplot(2,1,2)
plot(v*s)
legend('comp1','comp2','comp3','comp4','comp5','comp6')
xlabel('Time')
ylabel('Location')
title('Dynamic Captured with PCA');
%% Case 2

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

f1=size(vidFrames1_2,4);
f2=size(vidFrames2_2,4);
f3=size(vidFrames3_2,4);

%track the motion
x1=[];
y1=[];
for i=1:f1
    x=double(rgb2gray(vidFrames1_2(:,:,:,i)));
    x(400:end,:)=0;
    x(1:210,:)=0;
    x(:,400:end)=0;
    x(:,1:300)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
```

```
    y1(i)=mean(b);
    x1(i)=mean(a);
end

x2=[];
y2=[];
for i=1:f2
    x=double(rgb2gray(vidFrames2_2(:,:,:,i)));
    x(1:100,:)=0;
    x(370:end,:)=0;
    x(:,450:end)=0;
    x(:,1:200)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y2(i)=mean(b);
    x2(i)=mean(a);
end

x3=[];
y3=[];
for i=1:f3
    x=double(rgb2gray(vidFrames3_2(:,:,:,i)));
    x(1:210,:)=0;
    x(315:end,:)=0;
    x(:,500:end)=0;
    x(:,1:220)=0;
    M=max(x(:));
    [a,b]=find(x>=M*0.92);
    y3(i)=mean(b);
    x3(i)=mean(a);
end
x2=x2(20:end);
y2=y2(20:end);%aligh the second one
%plot(x1);hold on
%plot(x2);
%plot(y3)

%PCA

l=min([length(x1),length(x2),length(y3)]);
A=[x1(1:l);y1(1:l);x2(1:l);y2(1:l);x3(1:l);y3(1:l)];

[m,n]=size(A);
mn=mean(A,2);
A=A-repmat(mn,1,n);

[u,s,v]=svd(A,'econ');
figure(1)
plot(diag(s)/sum(diag(s)),'ro')
xlabel('Principal Component');
ylabel('Energy Level Percentage');
title('Noisy');

figure(2)
subplot(2,1,1)
```

```
plot(1:l,x1(1:l), 1:l, y1(1:l))
legend('z','xy')
xlabel('Time')
ylabel('Location')
title('Original Movement Cam1')
subplot(2,1,2)
plot(v*s)
legend('comp1','comp2','comp3','comp4','comp5','comp6')
xlabel('Time')
ylabel('Location')
title('Dynamic Captured with PCA');
%% Horizontal
clear all; close all; clc;
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')

f1=size(vidFrames1_3,4);
f2=size(vidFrames2_3,4);
f3=size(vidFrames3_3,4);

%track the motion
x1=[];
y1=[];
for i=1:f1
    x=double(rgb2gray(vidFrames1_3(:,:,:,i)));
    x(400:end,:)=0;
    x(1:210,:)=0;
    x(:,400:end)=0;
    x(:,1:300)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y1(i)=mean(b);
    x1(i)=mean(a);
end

x2=[];
y2=[];
for i=1:f2
    x=double(rgb2gray(vidFrames2_3(:,:,:,i)));
    x(1:150,:)=0;
    x(370:end,:)=0;
    x(:,450:end)=0;
    x(:,1:200)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y2(i)=mean(b);
    x2(i)=mean(a);
end

x3=[];
y3=[];
for i=1:f3
    x=double(rgb2gray(vidFrames3_3(:,:,:,i)));
    x(1:210,:)=0;
```

```
    x(315:end,:)=0;
    x(:,500:end)=0;
    x(:,1:220)=0;
    M=max(x(:));
    [a,b]=find(x>=M*0.92);
    y3(i)=mean(b);
    x3(i)=mean(a);
end

x1=x1(5:end);
y1=y1(5:end)
x2=x2(35:end);
y2=y2(35:end);%aligh the second one
%plot(x1);hold on
%plot(x2);
%plot(y3)

%PCA

l=min([length(x1),length(x2),length(y3)]);
A=[x1(1:l);y1(1:l);x2(1:l);y2(1:l);x3(1:l);y3(1:l)];

[m,n]=size(A);
mn=mean(A,2);
A=A-repmat(mn,1,n);

[u,s,v]=svd(A,'econ');
figure(1)
plot(diag(s)/sum(diag(s)),'ro')
xlabel('Principal Component');
ylabel('Energy Level Percentage');
title('Horizontal');

figure(2)
subplot(2,1,1)
plot(1:l,x1(1:l), 1:l, y1(1:l))
legend('z','xy')
xlabel('Time')
ylabel('Location')
title('Original Movement Cam1')
subplot(2,1,2)
plot(v*s)
legend('comp1','comp2','comp3','comp4','comp5','comp6')
xlabel('Time')
ylabel('Location')
title('Dynamic Captured with PCA');
%% Horizontal and Rotation
clear all; close all; clc;
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')

f1=size(vidFrames1_4,4);
f2=size(vidFrames2_4,4);
f3=size(vidFrames3_4,4);
```

```
%track the motion
x1=[];
y1=[];
for i=1:f1
    x=double(rgb2gray(vidFrames1_4(:,:,:,i)));
    x(400:end,:)=0;
    x(1:210,:)=0;
    x(:,400:end)=0;
    x(:,1:300)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y1(i)=mean(b);
    x1(i)=mean(a);
end

x2=[];
y2=[];
for i=1:f2
    x=double(rgb2gray(vidFrames2_4(:,:,:,i)));
    x(1:100,:)=0;
    x(370:end,:)=0;
    x(:,450:end)=0;
    x(:,1:200)=0;
    M=max(x(:));
    [a,b]=find(x>=(M*0.92));
    y2(i)=mean(b);
    x2(i)=mean(a);
end

x3=[];
y3=[];
for i=1:f3
    x=double(rgb2gray(vidFrames3_4(:,:,:,i)));
    x(1:150,:)=0;
    x(315:end,:)=0;
    x(:,500:end)=0;
    x(:,1:280)=0;
    M=max(x(:));
    [a,b]=find(x>=M*0.92);
    y3(i)=mean(b);
    x3(i)=mean(a);
end

x2=x2(5:end);
y2=y2(5:end);%aligh the second one
%plot(x1);hold on
%plot(x2);
%plot(y3)

%PCA

l=min([length(x1),length(x2),length(y3)]);
A=[x1(1:l);y1(1:l);x2(1:l);y2(1:l);x3(1:l);y3(1:l)];
```

```
[m,n]=size(A);
mn=mean(A,2);
A=A-repmat(mn,1,n);

[u,s,v]=svd(A,'econ');
figure(1)
plot(diag(s)/sum(diag(s)),'ro')
xlabel('Principal Component');
ylabel('Energy Level Percentage');
title('Horizontal and Rotation');

figure(2)
subplot(2,1,1)
plot(1:l,x1(1:l), 1:l, y1(1:l))
legend('z','xy')
xlabel('Time')
ylabel('Location')
title('Original Movement Cam1')
subplot(2,1,2)
plot(v*s)
legend('comp1','comp2','comp3','comp4','comp5','comp6')
xlabel('Time')
ylabel('Location')
title('Dynamic Captured with PCA');
```