Principal Component Analysis of Physical Systems

Chongyi Xu
University of Washington
AMATH 482/582 Winter Quarter 2018
chongyix@uw.edu

February 11, 2018

Abstract

The principal component analysis (PCA) is a kind of algorithms in biometrics. It is a statistics technical and used orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. PCA also is a tool to reduce multidimensional data to lower dimensions while retaining most of the information. It covers standard deviation, covariance, and eigenvectors.

1 Introduction and Overview

1.1 Introudction

In real life, it always happens that we would like to konw about a certain physical phenomena. However, without relative knowledge or theorm, it is hard to understand the phenomena by oneself. In order to understand our world, we sometimes tried to performe a certain experiment that could reproduce the real-life situation and recorded the data to study for it. And since we do not know the concept (or theorm) about the phenomena, the data we collected from experiments will always be redundant and noisy. The principal component analysis (PCA) provides statistical helps to rearrange and optimize the data such that the result we got from the data could be more convincing and effective.

1.2 Overview

We are going to explore 4 cases to illustrate various aspects of PCA and its practical usefulness and the effects of noise on the PCA algorithm.

- Ideal case
- Noisy case
- Horizontal displacement
- Horizontal displacement and rotation

We have the videos for every cases from 3 different cameras and the purpose is to understand the system.

2 Theoretical Background

PCA was invented in 1901 by Karl Pearson, as an analogue of the principal axis theorem in mechanics; it was later independently developed and named by Harold Hotelling in the 1930s. Depending on the field of application, it is also named the discrete Karhunen–Loève transform (KLT) in signal processing, the Hotelling transform in multivariate quality control, proper orthogonal decomposition (POD) in mechanical engineering, singular value decomposition (SVD) of X (Golub and Van Loan, 1983), eigenvalue decomposition (EVD) of XTX in linear algebra, factor analysis (for a discussion of the differences between PCA and factor analysis see Ch. 7 of [3]), Eckart–Young theorem (Harman, 1960), or Schmidt–Mirsky theorem in psychometrics, empirical orthogonal functions (EOF) in meteorological science, empirical eigenfunction decomposition (Sirovich, 1987), empirical component analysis (Lorenz, 1956), quasiharmonic modes (Brooks et al., 1988), spectral decomposition in noise and vibration, and empirical modal analysis in structural dynamics.

3 Algorithm Implementation and Development

The main idea of principal component analysis is to get the variance and covariance of the data to see if the data we have collected is either useful or redundant/noisy. Considering we have two data sets

$$\vec{a} = [a_1\ a_2\ a_3\ \dots a_n]$$
$$\vec{b} = [b_1\ b_2\ b_3\ \dots b_n]$$

Then, the variance of vectors $\vec{a}$ and $\vec{b}$ are defined to be

$$\sigma_a^2 = \frac{1}{n-1}\vec{a}\vec{a^T}$$
$$\sigma_b^2 = \frac{1}{n-1}\vec{b}\vec{b^T}$$

Also, we have the covariance between data sets $\vec{a}$ and $\vec{b}$

$$\sigma_{ab}^2 = \frac{1}{n-1}\vec{a}\vec{b^T}$$

This score tells that how much does dataset $\vec{a}$ depend on the other data set $\vec{b}$. In the other words, it tells how much are $\vec{a}$ and $\vec{b}$ in the same direction. Generally speaking, consider a data matrix X^{n*n}. The covariance among all pairs of data vectors could be genralized as

$$C_X = \frac{1}{n-1}XX^T = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \sigma_{13}^2 & \dots & \sigma_{1n}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & \sigma_{23}^2 & \dots & \sigma_{2n}^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1}^2 & \sigma_{n2}^2 & \sigma_{n3}^2 & \dots & \sigma_{nn}^2 \end{bmatrix}$$

And it can be seen that on the diagonal, we have the variances of $x_1 \dots x_n$. In order to find the data that are important to the system, we simply change the basis of C_X to diagonalize it. And $(C_X)_{diagonalized}$ tells that which direction matters and order the importance(energy) from highest to lowest on the diagonal. According to PCA, we changed the basis of the data matrix X to $Y = U^*X$ where U is the result matrix of XX^T from Singular Value Decomposition ($XX^T = U\Sigma V^*$). So, the covariance matrix of the new data matrix Y will be

$$\begin{aligned} C_Y &= \frac{1}{n-1}YY^T \\ &= \frac{1}{n-1}U^*X(U^*X)^T \\ &= \frac{1}{n-1}U^*XX^*U \\ &= \frac{1}{n-1}U^*U\Sigma V^*V\Sigma U^*U \\ &= \frac{\Sigma^2}{n-1} \end{aligned}$$

The Σ^2 here is exactly the diagonal matrix we are looking for.

4 Computational Results

In this specific experiment, we would like to know the motion of painted can to understand the physical system. In order to record the motion of the painted can from the given videos, I tried to catch the motion of the light on the painted can. The method I came up with is that store the frames into a matrices for every video. And then find the maximum value on x and y axis to figure out the position of the light. For example, for the first camera in test 1 (ideal case), on one certain frame, the position of the light I found is Figure 1

Figure 1: A frame of light I found(black box)

Then for convenience of comparing and computing, I made all data I got to be in the same size (length). With this step, we could then compare the displacement on every direction. In the first test (ideal case), the displacement vs. time graph for both x-axis and y-axis I got is Figure 2 In order to have a better under-

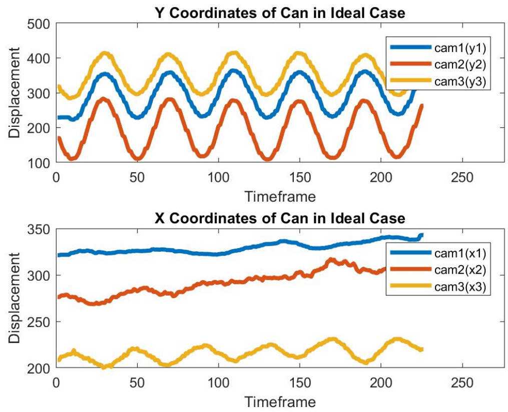

Figure 2: Ideal Case, Dispalcement vs Time

standing on the differences of displacement among different cameras, I normalized the displacement to have a more readable plot like Figure 3 Then it is time

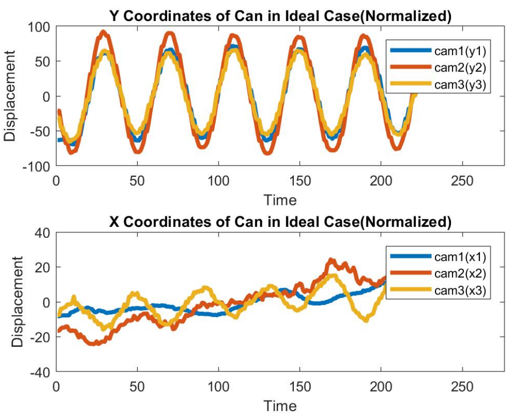

Figure 3: Ideal Case, Dispalcement vs Time (Normalized)

to discuss how many directions are playing roles in this experiment. The most simple way is to proceed a singular value decomposition (SVD) and have a look at the diagonal matrix. With doing that, I got Figure 4 and conclude that there is only one direction mattering the phenomena. In the other word, it is a 1D motion.

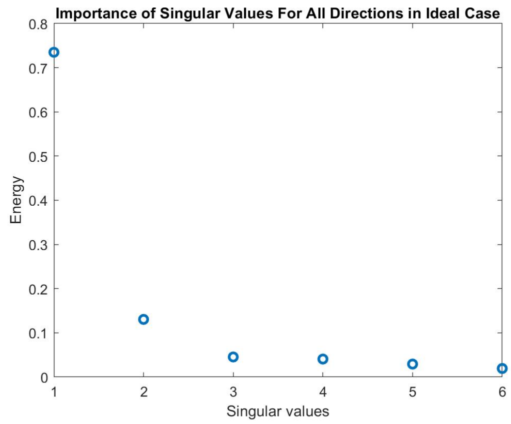

Figure 4: Ideal Case, Importance of Each Direction

Similarly, we could do the same analysis to other cases (noisy, horizontal displacement, horizontal dispalcement and rotation) From the graphs, we can

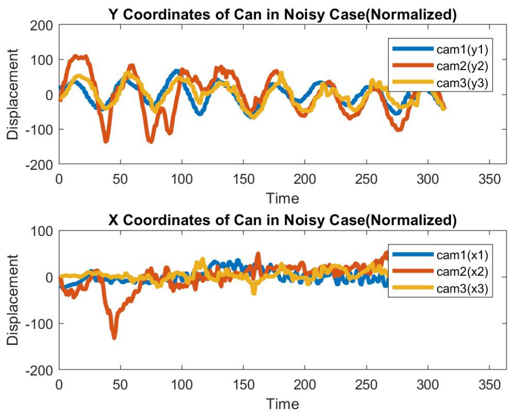

Figure 5: Noisy Case, Dispalcement vs Time (Normalized)

see that the noisy data affects our determination a lot according to Figure 5. In this figure, the second camera has its curve that is kind of off-track due to noise. However, from the importance graph (Figure 6), we can still conclude that there is only one single direction that matters since the other singular values

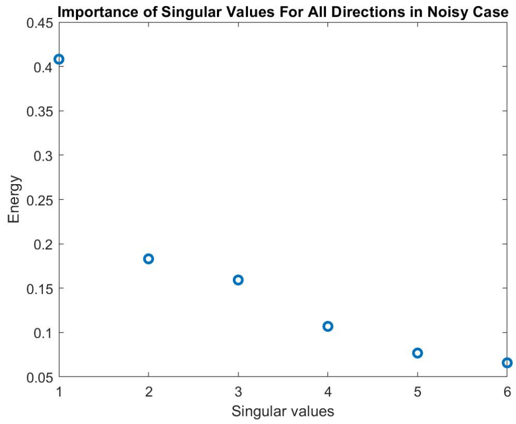

Figure 6: Noisy Case, Importance of Each Direction

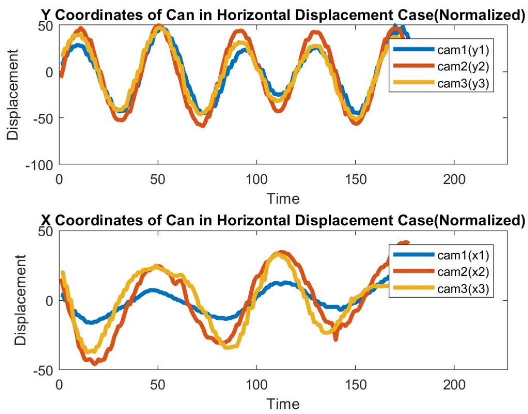

Figure 7: Horizontal Case, Dispalcement vs Time (Normalized)

has much lower importance (energy) comparing to the first singular value.
And third test (horizontal displacement) results in that 2 directions play important roles in its motion, which make sense since it moves horizonally (x-direction) and vertically (z-direction). We made the same conclusion due to Figure 8. And the last test is somehow tricky since there are two singular values that have values but not too big. I concluded that it is a 3D motion in test 4.

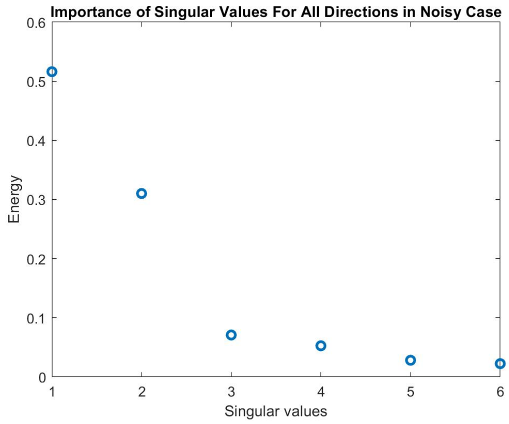

Figure 8: Horizontal Case, Importance of Each Direction

5 Summary and Conclusions

The Principal Component Analysis is useful and effective. Using this method, we are able to illustrate real-life phenomena from data perspective by simply reproduce the situation and study for it. However, it is very important to elinimate noisy data as much as possible since noise will have huge effect during the process of analysis. It is okay to take redundant data since PCA will take out those by chaging the basis. But it is hard to detect noise and those noise will affect our conclusion.

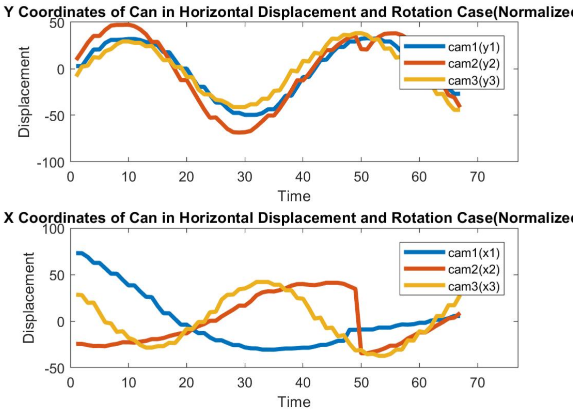

Figure 9: Horizontal and Rotation Case, Dispalcement vs Time (Normalized)

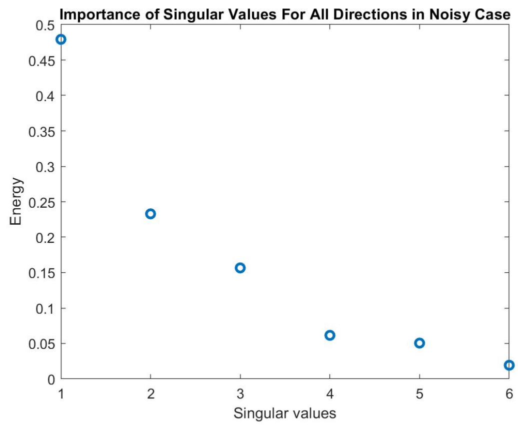

Figure 10: Horizontal and Rotation Case, Importance of Each Direction

```
%% TEST 1
clear all; close all; clc;
%% Import cam data
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');
%% Get rows/columns and dimensions
[row1, col1, dim1_1, dim2_1] = size(vidFrames1_1);
[row2, col2, dim1_2, dim2_2] = size(vidFrames2_1);
[row3, col3, dim1_3, dim2_3] = size(vidFrames3_1);
%% Store the video by frames
for k = 1:dim2_1
    img1{k} = vidFrames1_1(:, :, :, k);
end
for k = 1:dim2_2
    img2{k} = vidFrames2_1(:, :, :, k);
end

for k = 1:dim2_3
    img3{k} = vidFrames3_1(:, :, :, k);
end
%% Ideal Case
x1 = zeros(1, dim2_1);
y1 = zeros(1, dim2_1);
boxX1 = [300, 350];
boxY1 = [200, 250];
for k = 1:dim2_1
    img = vidFrames1_1(:, :, 3, k);
    box = double(img(boxY1(1):boxY1(2), boxX1(1):boxX1(2)));
    [row, col] = find(box == max(max(box)));
    y1(k) = mean(row) + boxY1(1);
    x1(k) = mean(col) + boxX1(1);
    boxX1 = [round(x1(k) - 20), round(x1(k) + 20)];
    boxY1 = [round(y1(k) - 20), round(y1(k) + 20)];
end

x2 = zeros(1, dim2_2);
y2 = zeros(1, dim2_2);
boxX2 = [250, 300];
boxY2 = [250, 300];
for k = 1:dim2_2
    img = vidFrames2_1(:, :, 3, k);
    box = double(img(boxY2(1):boxY2(2), boxX2(1):boxX2(2)));
    [row, col] = find(box == max(max(box)));
    y2(k) = mean(row) + boxY2(1);
    x2(k) = mean(col) + boxX2(1);
    boxX2 = [round(x2(k) - 20), round(x2(k) + 20)];
    boxY2 = [round(y2(k) - 20), round(y2(k) + 20)];
end

x3 = zeros(1, dim2_3);
y3 = zeros(1, dim2_3);
boxX3 = [310, 340];
boxY3 = [265, 295];
for k = 1:dim2_3
    img = vidFrames3_1(:, :, 3, k);
    box = double(img(boxY3(1):boxY3(2), boxX3(1):boxX3(2)));
    [row, col] = find(box == max(max(box)));
    y3(k) = mean(row) + boxY3(1);
    x3(k) = mean(col) + boxX3(1);
    boxX3 = [round(x3(k) - 15), round(x3(k) + 15)];
    boxY3 = [round(y3(k) - 15), round(y3(k) + 15)];
end

% %% Video1
```

```
% figure(1);
% for i = 1:length(img1)
%     imshow(img1{i});
%     hold on;
%     rectangle('position', [x1(i) - 10, y1(i) - 10, 20, 20], 'Linewidth', 2);
%     hold off;
%     pause(0.001);
% end
% %% Video2
% figure(2);
% for i = 1:length(img2)
%     imshow(img2{i});
%     hold on;
%     rectangle('position', [x2(i) - 10, y2(i) - 10, 20, 20], 'Linewidth', 2);
%     hold off;
%     pause(0.001)
% end
% %% Video3
% figure(3);
% for i = 1:length(img3)
%     imshow(img3{i});
%     hold on;
%     rectangle('position', [x3(i) - 15, y3(i) - 15, 30, 30], 'Linewidth', 2);
%     hold off;
%     pause(0.001)
% end

%% Force everything to be in the same size
x1; y1; % Reference
x2 = x2(11:dim2_1 + 10);
y2 = y2(11:dim2_1 + 10);
x3 = x3(1:dim2_1);
y3 = 480 - y3(1:dim2_1);
%% Plot
figure(4);
subplot(2,1,1);
plot(y1,'Linewidth', 3); hold on;
plot(y2,'Linewidth', 3)
plot(x3,'Linewidth', 3)
title('Y Coordinates of Can in Ideal Case');
xlabel('Timeframe');
ylabel('Displacement');
xlim([0 length(y1) + 50]);
legend('cam1(y1)','cam2(y2)','cam3(y3)');

subplot(2,1,2);
plot(x1,'Linewidth', 3); hold on;
plot(x2,'Linewidth', 3)
plot(y3,'Linewidth', 3)
title('X Coordinates of Can in Ideal Case');
xlabel('Timeframe');
ylabel('Displacement');
xlim([0 length(y1) + 50]);
legend('cam1(x1)','cam2(x2)','cam3(x3)');

%% Normalize the Data
figure(5)
subplot(2,1,1);
plot(y1 - mean(y1),'Linewidth', 3); hold on;
plot(y2 - mean(y2),'Linewidth', 3)
plot(x3 - mean(x3),'Linewidth', 3)
title('Y Coordinates of Can in Ideal Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
xlim([0 length(y1) + 50]);
```

```
legend('cam1(y1)','cam2(y2)','cam3(y3)');

subplot(2,1,2);
plot(x1 - mean(x1),'Linewidth', 3); hold on;
plot(x2 - mean(x2),'Linewidth', 3)
plot(y3 - mean(y3),'Linewidth', 3)
title('X Coordinates of Can in Ideal Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
legend('cam1(x1)','cam2(x2)','cam3(x3)');
xlim([0 length(y1) + 50]);

%% SVD
X = [x1 - mean(x1); y1 - mean(y1); x2 - mean(x2); y2 - mean(y2); ...
     x3 - mean(x3); y3 - mean(y3)];
[u, s, v] = svd(X, 'econ');

%% Plot
figure;
plot(diag(s)/sum(diag(s)),'o', 'Linewidth', 2);
title('Importance of Singular Values For All Directions in Ideal Case');
xlabel('Singular values');
ylabel('Energy');
%%
%---------------------------------------------------------------------------
%% TEST 2
clc; clear all; close all;
%% noisy case
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
%%
[row1, col1, dim1_1, dim2_1] = size(vidFrames1_2);
[row2, col2, dim1_2, dim2_2] = size(vidFrames2_2);
[row3, col3, dim1_3, dim2_3] = size(vidFrames3_2);
%%
for k = 1:dim2_1
    img1{k} = vidFrames1_2(:, :, :, k);
end
for k = 1:dim2_2
    img2{k} = vidFrames2_2(:, :, :, k);
end

for k = 1:dim2_3
    img3{k} = vidFrames3_2(:, :, :, k);
end
%% case 2
x1 = zeros(1, dim2_1);
y1 = zeros(1, dim2_1);
boxX = [300 350];
boxY = [300 350];
for i = 1:dim2_1
    img = vidFrames1_2(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y1(i) = mean(row) + boxY(1);
    x1(i) = mean(col) + boxX(1);
    boxX = [round(x1(i) - 20), round(x1(i) + 20)];
    boxY = [round(y1(i) - 20), round(y1(i) + 20)];
end

x2 = zeros(1, dim2_2);
y2 = zeros(1, dim2_2);
boxX = [290 340];
boxY = [330 380];
```

```matlab
for i = 1:dim2_2
    img = vidFrames2_2(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y2(i) = mean(row) + boxY(1);
    x2(i) = mean(col) + boxX(1);
    boxX = [round(x2(i) - 30), round(x2(i) + 30)];
    boxY = [round(y2(i) - 30), round(y2(i) + 30)];
end


x3 = zeros(1, dim2_3);
y3 = zeros(1, dim2_3);
boxX = [335 365];
boxY = [240 270];
%[335, 240, 20, 20]
count = 0;
for i = 1:dim2_3
    count = count + 1;
    img = vidFrames3_2(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y3(i) = mean(row) + boxY(1);
    x3(i) = mean(col) + boxX(1);
    boxX = [round(x3(i) - 30), round(x3(i) + 30)];
    boxY = [round(y3(i) - 30), round(y3(i) + 30)];
end
% %%
% figure(1);
% for i = 1:length(img1)
%    imshow(img1{i});
%    hold on;
%    rectangle('position', [x1(i) - 20, y1(i) - 20, 40, 40], 'Linewidth', 2);
%    hold off;
%    pause(0.001)
% end
% %%
% figure(2);
% for i = 1:length(img2)
%    imshow(img2{i});
%    hold on;
%    rectangle('position', [x2(i) - 15, y2(i) - 15, 30, 30], 'Linewidth', 2);
%    hold off;
%    pause(0.001)
% end
% %%
% figure(3);
% for i = 1:length(img3)
%    imshow(img3{i});
%    hold on;
%    rectangle('position', [x3(i) - 15, y3(i) - 15, 30, 30], 'Linewidth', 2);
%    hold off;
%    pause(0.001)
% end
%%
x1; y1;
x2 = x2(23:dim2_1 + 22);
y2 = y2(23:dim2_1 + 22);
x3 = x3(1:dim2_1);
y3 = y3(1:dim2_1);
%%
figure(2)
subplot(2,1,1);
plot(y1 - mean(y1),'Linewidth', 3); hold on;
plot(y2 - mean(y2),'Linewidth', 3)
```

```
plot(x3 - mean(x3),'Linewidth', 3)
title('Y Coordinates of Can in Noisy Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
xlim([0 length(y1) + 50]);
legend('cam1(y1)','cam2(y2)','cam3(y3)');

subplot(2,1,2);
plot(x1 - mean(x1),'Linewidth', 3); hold on;
plot(x2 - mean(x2),'Linewidth', 3)
plot(y3 - mean(y3),'Linewidth', 3)
title('X Coordinates of Can in Noisy Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
legend('cam1(x1)','cam2(x2)','cam3(x3)');
xlim([0 length(y1) + 50]);
%% SVD
X = [x1- mean(x1); y1- mean(y1); x2- mean(x2); y2- mean(y2); y3- mean(y3); x3- mean(x3);];
[u, s, v] = svd(X, 'econ');
%%
figure(3)
plot(diag(s)/sum(diag(s)), 'o','Linewidth', 2);
title('Importance of Singular Values For All Directions in Noisy Case');
xlabel('Singular values');
ylabel('Energy');
%%
%---------------------------------------------------------------------------
%% TEST 3
clc; clear all; close all;
%% horizontal displacement
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
%%
[row1, col1, dim1_1, dim2_1] = size(vidFrames1_3);
[row2, col2, dim1_2, dim2_2] = size(vidFrames2_3);
[row3, col3, dim1_3, dim2_3] = size(vidFrames3_3);
%%
for k = 1:dim2_1
    img1{k} = vidFrames1_3(:, :, :, k);
end
for k = 1:dim2_2
    img2{k} = vidFrames2_3(:, :, :, k);
end

for k = 1:dim2_3
    img3{k} = vidFrames3_3(:, :, :, k);
end
%% case 2
x1 = zeros(1, dim2_1);
y1 = zeros(1, dim2_1);
boxX = [310 330];
boxY = [280 300];
for i = 1:dim2_1
    img = vidFrames1_3(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y1(i) = mean(row) + boxY(1);
    x1(i) = mean(col) + boxX(1);
    boxX = [round(x1(i) - 10), round(x1(i) + 10)];
    boxY = [round(y1(i) - 10), round(y1(i) + 10)];
end

x2 = zeros(1, dim2_2);
y2 = zeros(1, dim2_2);
```

```
boxX = [230 260];
boxY = [280 310];
for i = 1:dim2_2
    img = vidFrames2_3(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y2(i) = mean(row) + boxY(1);
    x2(i) = mean(col) + boxX(1);
    boxX = [round(x2(i) - 10), round(x2(i) + 10)];
    boxY = [round(y2(i) - 10), round(y2(i) + 10)];
end


x3 = zeros(1, dim2_3);
y3 = zeros(1, dim2_3);
boxX = [345 375];
boxY = [220 250];
count = 0;
for i = 1:dim2_3
    count = count + 1;
    img = vidFrames3_3(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y3(i) = mean(row) + boxY(1);
    x3(i) = mean(col) + boxX(1);
    boxX = [round(x3(i) - 20), round(x3(i) + 20)];
    boxY = [round(y3(i) - 20), round(y3(i) + 20)];
end
% %%
% figure(1);
% for i = 1:length(img1)
%     imshow(img1{i});
%     hold on;
%     rectangle('position', [x1(i) - 10, y1(i) - 10, 20, 20], 'Linewidth', 2);
%     hold off;
%     pause(0.001)
% end
%%
% figure(2);
% for i = 1:length(img2)
%     imshow(img2{i});
%     hold on;
%     rectangle('position', [x2(i) - 10, y2(i) - 10, 20, 20], 'Linewidth', 2);
%     hold off;
%     pause(0.001)
% end
%%
% figure(3);
% for i = 1:length(img3)
%     imshow(img3{i});
%     hold on;
%     rectangle('position', [x3(i) - 10, y3(i) - 10, 20, 20], 'Linewidth', 2);
%     hold off;
%     pause(0.001)
% end
%%
x1 = x1(8:177 + 7);
y1 = y1(8:177 + 7);
x2 = x2(36:177+35);
y2 = y2(36:177+35);
x3 = x3(1:177);
y3 = 480 - y3(1:177);
%%
figure(2)
subplot(2,1,1);
```

```
plot(y1 - mean(y1),'Linewidth', 3); hold on;
plot(y2 - mean(y2),'Linewidth', 3)
plot(x3 - mean(x3),'Linewidth', 3)
title('Y Coordinates of Can in Horizontal Displacement Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
xlim([0 length(y1) + 50]);
legend('cam1(y1)','cam2(y2)','cam3(y3)');

subplot(2,1,2);
plot(x1 - mean(x1),'Linewidth', 3); hold on;
plot(x2 - mean(x2),'Linewidth', 3)
plot(y3 - mean(y3),'Linewidth', 3)
title('X Coordinates of Can in Horizontal Displacement Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
legend('cam1(x1)','cam2(x2)','cam3(x3)');
xlim([0 length(y1) + 50]);
%% SVD
X = [x1- mean(x1); y1- mean(y1); x2- mean(x2); y2- mean(y2); y3- mean(y3); x3- mean(x3);];
[u, s, v] = svd(X, 'econ');
%%
figure(3)
plot(diag(s)/sum(diag(s)), 'o','Linewidth', 2);
title('Importance of Singular Values For All Directions in Noisy Case');
xlabel('Singular values');
ylabel('Energy');
%%
%-------------------------------------------------------------------------
%% TEST 4
clc; clear all; close all;
%% horizontal displacement and rotation
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
%%
[row1, col1, dim1_1, dim2_1] = size(vidFrames1_4);
[row2, col2, dim1_2, dim2_2] = size(vidFrames2_4);
[row3, col3, dim1_3, dim2_3] = size(vidFrames3_4);
%%
for k = 1:dim2_1
    img1{k} = vidFrames1_4(:, :, :, k);
end
for k = 1:dim2_2
    img2{k} = vidFrames2_4(:, :, :, k);
end

for k = 1:dim2_3
    img3{k} = vidFrames3_4(:, :, :, k);
end
%%
boxX = [430 460];
boxY = [260 290];
for i = 20:70
    img = vidFrames1_4(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    if (max(max(box)) == max(max(img)))
        [row, col] = find(box == max(max(box)));
        y1(i) = mean(row) + boxY(1);
        x1(i) = mean(col) + boxX(1);
        boxX = [round(x1(i) - 15), round(x1(i) + 15)];
        boxY = [round(y1(i) - 15), round(y1(i) + 15)];
    end
end
```

```
boxX = [340 370];
boxY = [320 350];
for i = 150:200
    img = vidFrames1_4(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    if (max(max(box)) == max(max(img)))
        [row, col] = find(box == max(max(box)));
        y1(i) = mean(row) + boxY(1);
        x1(i) = mean(col) + boxX(1);
        boxX = [round(x1(i) - 15), round(x1(i) + 15)];
        boxY = [round(y1(i) - 15), round(y1(i) + 15)];
    end
end
x1 = x1(y1>0);
y1 = y1(y1>0);

%%
boxX = [255 285];
boxY = [180 210];
for i = 152:200
    img = vidFrames2_4(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    if (max(max(box)) == max(max(img)))
        [row, col] = find(box == max(max(box)));
        y2(i) = mean(row) + boxY(1);
        x2(i) = mean(col) + boxX(1);
        boxX = [round(x2(i) - 15), round(x2(i) + 15)];
        boxY = [round(y2(i) - 15), round(y2(i) + 15)];
    end
end

boxX = [248 278];
boxY = [198 228];
for i = 235:274
    img = vidFrames2_4(:,:,3,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    if (max(max(box)) == max(max(img)))
        [row, col] = find(box == max(max(box)));
        y2(i) = mean(row) + boxY(1);
        x2(i) = mean(col) + boxX(1);
        boxX = [round(x2(i) - 15), round(x2(i) + 15)];
        boxY = [round(y2(i) - 15), round(y2(i) + 15)];
    else
        y2(i) = y2(i-1);
        x2(i) = x2(i-1);
        boxX = [round(x2(i) - 50), round(x2(i) + 50)];
        boxY = [round(y2(i) - 50), round(y2(i) + 50)];
    end
end
x2 = x2(y2>0);
y2 = y2(y2>0);
%%
boxX = [310 340];
boxY = [210 240];
for i = 50:100
    img = vidFrames3_4(:,:,2,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y3(i) = mean(row) + boxY(1);
    x3(i) = mean(col) + boxX(1);
    boxX = [round(x3(i) - 15), round(x3(i) + 15)];
    boxY = [round(y3(i) - 15), round(y3(i) + 15)];
end
boxX = [365 395];
boxY = [220 250];
```

```
for i = 120:170
    img = vidFrames3_4(:,:,2,i);
    box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
    [row, col] = find(box == max(max(box)));
    y3(i) = mean(row) + boxY(1);
    x3(i) = mean(col) + boxX(1);
    boxX = [round(x3(i) - 15), round(x3(i) + 15)];
    boxY = [round(y3(i) - 15), round(y3(i) + 15)];
end
y3 = x3(x3>0);
x3 = x3(x3>0);
y3 = [y3(10:40) y3(63:end)];
x3 = [x3(10:40) x3(63:end)];
%%
x1 = x1(5:length(x3));
y1 = y1(5:length(x3));
x2 = x2(1:length(x3) - 4);
y2 = y2(1:length(x3) - 4);
x3 = x3(5:length(x3));
y3 = 480 - y3(1:length(x3));
%%
figure(2)
subplot(2,1,1);
plot(y1 - mean(y1),'Linewidth', 3); hold on;
plot(y2 - mean(y2),'Linewidth', 3)
plot(x3 - mean(x3),'Linewidth', 3)
title('Y Coordinates of Can in Horizontal Displacement and Rotation Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
xlim([0 length(y1) + 10]);
legend('cam1(y1)','cam2(y2)','cam3(y3)');

subplot(2,1,2);
plot(x1 - mean(x1),'Linewidth', 3); hold on;
plot(x2 - mean(x2),'Linewidth', 3)
plot(y3 - mean(y3),'Linewidth', 3)
title('X Coordinates of Can in Horizontal Displacement and Rotation Case(Normalized)');
xlabel('Time');
ylabel('Displacement');
legend('cam1(x1)','cam2(x2)','cam3(x3)');
xlim([0 length(y1) + 10]);
%% SVD
X = [x1- mean(x1); y1- mean(y1); x2- mean(x2); y2- mean(y2); y3- mean(y3); x3- mean(x3);];
[u, s, v] = svd(X, 'econ');
%%
figure(3)
plot(diag(s)/sum(diag(s)), 'o','Linewidth', 2);
title('Importance of Singular Values For All Directions in Noisy Case');
xlabel('Singular values');
ylabel('Energy');
```