

# 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.

## 1.2 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.

## 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 theorem, 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 theorem) 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.

## 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  $XX^T$  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

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

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

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

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  $X^{n \times n}$ . The covariance among all pairs of data vectors could be generalized as

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

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  $(C_X)_{\text{diagonalized}}$  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

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

The  $\Sigma^2$  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-

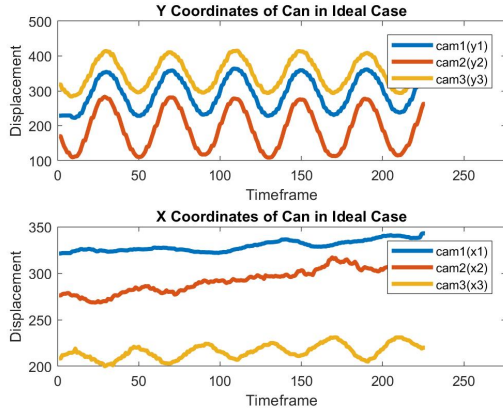


Figure 2: Ideal Case, Displacement vs Time

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

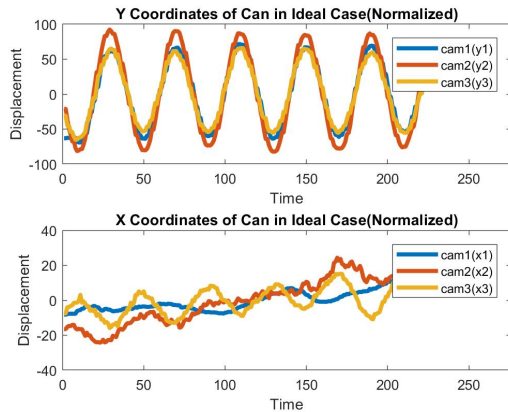


Figure 3: Ideal Case, Displacement vs Time (Normalized)

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.

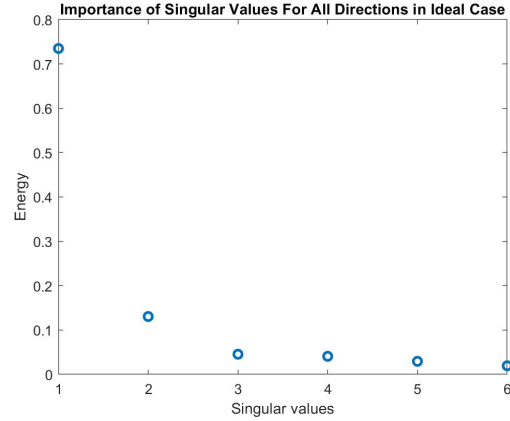


Figure 4: Ideal Case, Importance of Each Direction

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

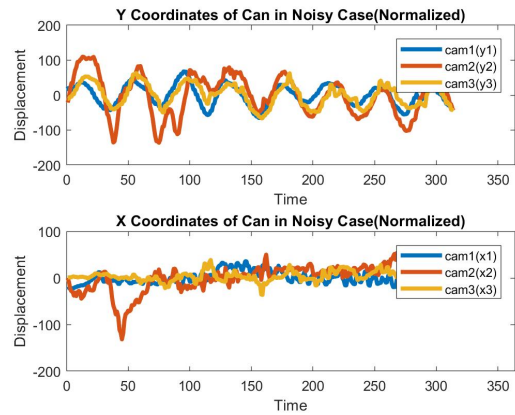


Figure 5: Noisy Case, Displacement vs Time (Normalized)

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

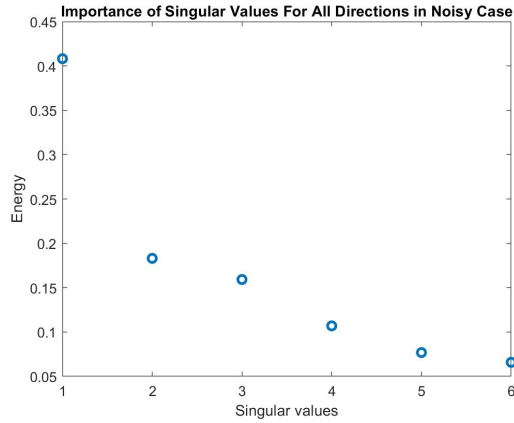


Figure 6: Noisy Case, Importance of Each Direction

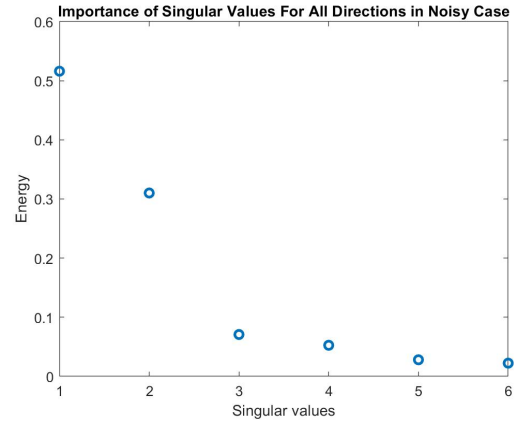


Figure 8: Horizontal Case, Importance of Each Direction

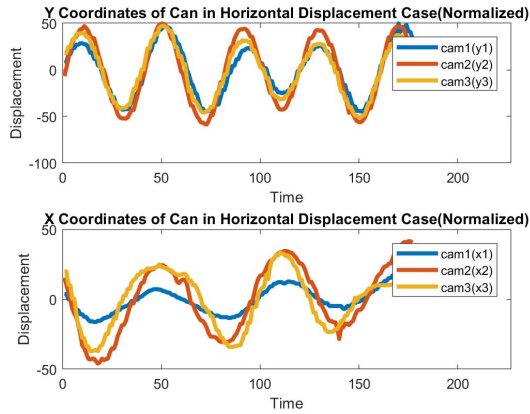


Figure 7: Horizontal Case, Displacement vs Time (Normalized)

## 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 eliminate 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 changing the basis. But it is hard to detect noise and those noise will affect our conclusion.

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 horizontally (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.

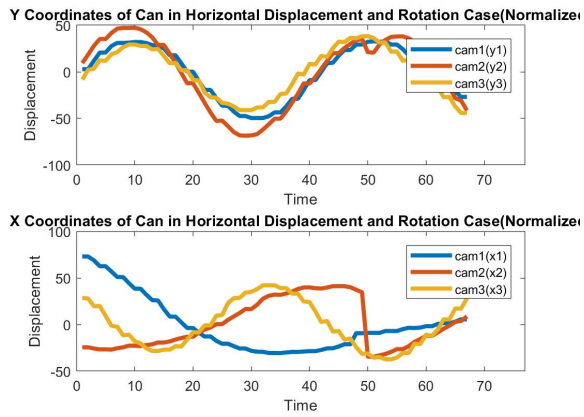


Figure 9: Horizontal and Rotation Case, Displacment vs Time (Normalized)

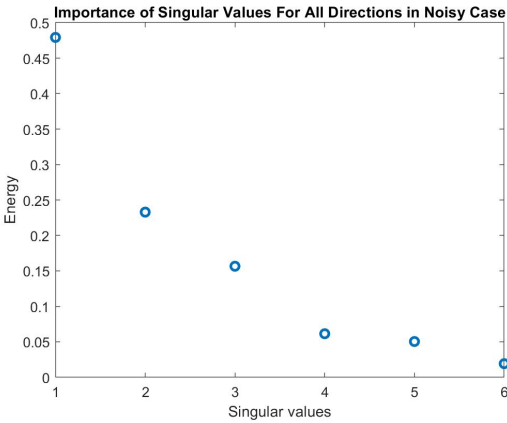


Figure 10: Horizontal and Rotation Case, Importance of Each Direction

```

1  %% TEST 1
2  clear all; close all; clc;
3  %% Import cam data
4  load('cam1_1.mat');
5  load('cam2_1.mat');
6  load('cam3_1.mat');
7  %% Get rows/columns and dimensions
8  [row1, col1, dim1_1, dim2_1] = size(vidFrames1_1);
9  [row2, col2, dim1_2, dim2_2] = size(vidFrames2_1);
10 [row3, col3, dim1_3, dim2_3] = size(vidFrames3_1);
11 %% Store the video by frames
12 for k = 1:dim2_1
13     img1{k} = vidFrames1_1(:, :, :, k);
14 end
15 for k = 1:dim2_2
16     img2{k} = vidFrames2_1(:, :, :, k);
17 end
18
19 for k = 1:dim2_3
20     img3{k} = vidFrames3_1(:, :, :, k);
21 end
22 %% Ideal Case
23 x1 = zeros(1, dim2_1);
24 y1 = zeros(1, dim2_1);
25 boxX1 = [300, 350];
26 boxY1 = [200, 250];
27 for k = 1:dim2_1
28     img = vidFrames1_1(:, :, 3, k);
29     box = double(img(boxY1(1):boxY1(2), boxX1(1):boxX1(2)));
30     [row, col] = find(box == max(max(box)));
31     y1(k) = mean(row) + boxY1(1);
32     x1(k) = mean(col) + boxX1(1);
33     boxX1 = [round(x1(k) - 20), round(x1(k) + 20)];
34     boxY1 = [round(y1(k) - 20), round(y1(k) + 20)];
35 end
36
37 x2 = zeros(1, dim2_2);
38 y2 = zeros(1, dim2_2);
39 boxX2 = [250, 300];
40 boxY2 = [250, 300];
41 for k = 1:dim2_2
42     img = vidFrames2_1(:, :, 3, k);
43     box = double(img(boxY2(1):boxY2(2), boxX2(1):boxX2(2)));
44     [row, col] = find(box == max(max(box)));
45     y2(k) = mean(row) + boxY2(1);
46     x2(k) = mean(col) + boxX2(1);
47     boxX2 = [round(x2(k) - 20), round(x2(k) + 20)];
48     boxY2 = [round(y2(k) - 20), round(y2(k) + 20)];
49 end
50
51 x3 = zeros(1, dim2_3);
52 y3 = zeros(1, dim2_3);
53 boxX3 = [310, 340];
54 boxY3 = [265, 295];
55 for k = 1:dim2_3
56     img = vidFrames3_1(:, :, 3, k);
57     box = double(img(boxY3(1):boxY3(2), boxX3(1):boxX3(2)));
58     [row, col] = find(box == max(max(box)));
59     y3(k) = mean(row) + boxY3(1);
60     x3(k) = mean(col) + boxX3(1);
61     boxX3 = [round(x3(k) - 15), round(x3(k) + 15)];
62     boxY3 = [round(y3(k) - 15), round(y3(k) + 15)];
63 end
64
65 % %% Video1

```

```

66 % figure(1);
67 % for i = 1:length(img1)
68 %     imshow(img1{i});
69 %     hold on;
70 %     rectangle('position', [x1(i) - 10, y1(i) - 10, 20, 20], 'Linewidth', 2);
71 %     hold off;
72 %     pause(0.001);
73 % end
74 % %% Video2
75 % figure(2);
76 % for i = 1:length(img2)
77 %     imshow(img2{i});
78 %     hold on;
79 %     rectangle('position', [x2(i) - 10, y2(i) - 10, 20, 20], 'Linewidth', 2);
80 %     hold off;
81 %     pause(0.001)
82 % end
83 % %% Video3
84 % figure(3);
85 % for i = 1:length(img3)
86 %     imshow(img3{i});
87 %     hold on;
88 %     rectangle('position', [x3(i) - 15, y3(i) - 15, 30, 30], 'Linewidth', 2);
89 %     hold off;
90 %     pause(0.001)
91 % end
92
93 %% Force everything to be in the same size
94 x1; y1; % Reference
95 x2 = x2(11:dim2_1 + 10);
96 y2 = y2(11:dim2_1 + 10);
97 x3 = x3(1:dim2_1);
98 y3 = 480 - y3(1:dim2_1);
99 %% Plot
100 figure(4);
101 subplot(2,1,1);
102 plot(y1,'Linewidth', 3); hold on;
103 plot(y2,'Linewidth', 3)
104 plot(y3,'Linewidth', 3)
105 title('Y Coordinates of Can in Ideal Case');
106 xlabel('Timeframe');
107 ylabel('Displacement');
108 xlim([0 length(y1) + 50]);
109 legend('cam1(y1)', 'cam2(y2)', 'cam3(y3)');
110
111 subplot(2,1,2);
112 plot(x1,'Linewidth', 3); hold on;
113 plot(x2,'Linewidth', 3)
114 plot(y3,'Linewidth', 3)
115 title('X Coordinates of Can in Ideal Case');
116 xlabel('Timeframe');
117 ylabel('Displacement');
118 xlim([0 length(y1) + 50]);
119 legend('cam1(x1)', 'cam2(x2)', 'cam3(x3)');
120
121 %% Normalize the Data
122 figure(5)
123 subplot(2,1,1);
124 plot(y1 - mean(y1),'Linewidth', 3); hold on;
125 plot(y2 - mean(y2),'Linewidth', 3)
126 plot(x3 - mean(x3),'Linewidth', 3)
127 title('Y Coordinates of Can in Ideal Case(Normalized)');
128 xlabel('Time');
129 ylabel('Displacement');
130 xlim([0 length(y1) + 50]);

```

```

131 legend('cam1(y1)', 'cam2(y2)', 'cam3(y3)');
132
133 subplot(2,1,2);
134 plot(x1 - mean(x1), 'Linewidth', 3); hold on;
135 plot(x2 - mean(x2), 'Linewidth', 3);
136 plot(y3 - mean(y3), 'Linewidth', 3);
137 title('X Coordinates of Can in Ideal Case(Normalized)');
138 xlabel('Time');
139 ylabel('Displacement');
140 legend('cam1(x1)', 'cam2(x2)', 'cam3(x3)');
141 xlim([0 length(y1) + 50]);
142
143 %% SVD
144 X = [x1 - mean(x1); y1 - mean(y1); x2 - mean(x2); y2 - mean(y2); ...
145      x3 - mean(x3); y3 - mean(y3)];
146 [u, s, v] = svd(X, 'econ');
147
148 %% Plot
149 figure;
150 plot(diag(s)/sum(diag(s)), 'o', 'Linewidth', 2);
151 title('Importance of Singular Values For All Directions in Ideal Case');
152 xlabel('Singular values');
153 ylabel('Energy');
154 %%
155 %-----
156 %% TEST 2
157 clc; clear all; close all;
158 %% noisy case
159 load('cam1_2.mat')
160 load('cam2_2.mat')
161 load('cam3_2.mat')
162 %%
163 [row1, col1, dim1_1, dim2_1] = size(vidFrames1_2);
164 [row2, col2, dim1_2, dim2_2] = size(vidFrames2_2);
165 [row3, col3, dim1_3, dim2_3] = size(vidFrames3_2);
166 %%
167 for k = 1:dim2_1
168     img1{k} = vidFrames1_2(:, :, :, k);
169 end
170 for k = 1:dim2_2
171     img2{k} = vidFrames2_2(:, :, :, k);
172 end
173
174 for k = 1:dim2_3
175     img3{k} = vidFrames3_2(:, :, :, k);
176 end
177 %% case 2
178 x1 = zeros(1, dim2_1);
179 y1 = zeros(1, dim2_1);
180 boxX = [300 350];
181 boxY = [300 350];
182 for i = 1:dim2_1
183     img = vidFrames1_2(:, :, 3, i);
184     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
185     [row, col] = find(box == max(max(box)));
186     y1(i) = mean(row) + boxY(1);
187     x1(i) = mean(col) + boxX(1);
188     boxX = [round(x1(i) - 20), round(x1(i) + 20)];
189     boxY = [round(y1(i) - 20), round(y1(i) + 20)];
190 end
191
192 x2 = zeros(1, dim2_2);
193 y2 = zeros(1, dim2_2);
194 boxX = [290 340];
195 boxY = [330 380];

```



```

196 for i = 1:dim2_2
197     img = vidFrames2_2(:,:,3,i);
198     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
199     [row, col] = find(box == max(max(box)));
200     y2(i) = mean(row) + boxY(1);
201     x2(i) = mean(col) + boxX(1);
202     boxX = [round(x2(i) - 30), round(x2(i) + 30)];
203     boxY = [round(y2(i) - 30), round(y2(i) + 30)];
204 end
205
206
207 x3 = zeros(1, dim2_3);
208 y3 = zeros(1, dim2_3);
209 boxX = [335 365];
210 boxY = [240 270];
211 %[335, 240, 20, 20]
212 count = 0;
213 for i = 1:dim2_3
214     count = count + 1;
215     img = vidFrames3_2(:,:,3,i);
216     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
217     [row, col] = find(box == max(max(box)));
218     y3(i) = mean(row) + boxY(1);
219     x3(i) = mean(col) + boxX(1);
220     boxX = [round(x3(i) - 30), round(x3(i) + 30)];
221     boxY = [round(y3(i) - 30), round(y3(i) + 30)];
222 end
223 % %%
224 % figure(1);
225 % for i = 1:length(img1)
226 %     imshow(img1{i});
227 %     hold on;
228 %     rectangle('position', [x1(i) - 20, y1(i) - 20, 40, 40], 'Linewidth', 2);
229 %     hold off;
230 %     pause(0.001)
231 % end
232 % %%
233 % figure(2);
234 % for i = 1:length(img2)
235 %     imshow(img2{i});
236 %     hold on;
237 %     rectangle('position', [x2(i) - 15, y2(i) - 15, 30, 30], 'Linewidth', 2);
238 %     hold off;
239 %     pause(0.001)
240 % end
241 % %%
242 % figure(3);
243 % for i = 1:length(img3)
244 %     imshow(img3{i});
245 %     hold on;
246 %     rectangle('position', [x3(i) - 15, y3(i) - 15, 30, 30], 'Linewidth', 2);
247 %     hold off;
248 %     pause(0.001)
249 % end
250 %%
251 x1; y1;
252 x2 = x2(23:dim2_1 + 22);
253 y2 = y2(23:dim2_1 + 22);
254 x3 = x3(1:dim2_1);
255 y3 = y3(1:dim2_1);
256 %%
257 figure(2)
258 subplot(2,1,1);
259 plot(y1 - mean(y1), 'Linewidth', 3); hold on;
260 plot(y2 - mean(y2), 'Linewidth', 3)

```

```

261 plot(x3 - mean(x3),'Linewidth', 3)
262 title('Y Coordinates of Can in Noisy Case(Normalized)');
263 xlabel('Time');
264 ylabel('Displacement');
265 xlim([0 length(y1) + 50]);
266 legend('cam1(y1)', 'cam2(y2)', 'cam3(y3)');
267
268 subplot(2,1,2);
269 plot(x1 - mean(x1),'Linewidth', 3); hold on;
270 plot(x2 - mean(x2),'Linewidth', 3)
271 plot(y3 - mean(y3),'Linewidth', 3)
272 title('X Coordinates of Can in Noisy Case(Normalized)');
273 xlabel('Time');
274 ylabel('Displacement');
275 legend('cam1(x1)', 'cam2(x2)', 'cam3(x3)');
276 xlim([0 length(y1) + 50]);
277 %% SVD
278 X = [x1- mean(x1); y1- mean(y1); x2- mean(x2); y2- mean(y2); y3- mean(y3); x3- mean(x3)];
279 [u, s, v] = svd(X, 'econ');
280 %%
281 figure(3)
282 plot(diag(s)/sum(diag(s)), 'o','Linewidth', 2);
283 title('Importance of Singular Values For All Directions in Noisy Case');
284 xlabel('Singular values');
285 ylabel('Energy');
286 %%
287 %-----
288 %% TEST 3
289 clc; clear all; close all;
290 %% horizontal displacement
291 load('cam1_3.mat')
292 load('cam2_3.mat')
293 load('cam3_3.mat')
294 %%
295 [row1, col1, dim1_1, dim2_1] = size(vidFrames1_3);
296 [row2, col2, dim1_2, dim2_2] = size(vidFrames2_3);
297 [row3, col3, dim1_3, dim2_3] = size(vidFrames3_3);
298 %%
299 for k = 1:dim2_1
300     img1{k} = vidFrames1_3(:, :, :, k);
301 end
302 for k = 1:dim2_2
303     img2{k} = vidFrames2_3(:, :, :, k);
304 end
305
306 for k = 1:dim2_3
307     img3{k} = vidFrames3_3(:, :, :, k);
308 end
309 %% case 2
310 x1 = zeros(1, dim2_1);
311 y1 = zeros(1, dim2_1);
312 boxX = [310 330];
313 boxY = [280 300];
314 for i = 1:dim2_1
315     img = vidFrames1_3(:, :, 3, i);
316     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
317     [row, col] = find(box == max(max(box)));
318     y1(i) = mean(row) + boxY(1);
319     x1(i) = mean(col) + boxX(1);
320     boxX = [round(x1(i) - 10), round(x1(i) + 10)];
321     boxY = [round(y1(i) - 10), round(y1(i) + 10)];
322 end
323
324 x2 = zeros(1, dim2_2);
325 y2 = zeros(1, dim2_2);

```

```

326 boxX = [230 260];
327 boxY = [280 310];
328 for i = 1:dim2_2
329     img = vidFrames2_3(:,:,3,i);
330     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
331     [row, col] = find(box == max(max(box)));
332     y2(i) = mean(row) + boxY(1);
333     x2(i) = mean(col) + boxX(1);
334     boxX = [round(x2(i) - 10), round(x2(i) + 10)];
335     boxY = [round(y2(i) - 10), round(y2(i) + 10)];
336 end
337
338
339 x3 = zeros(1, dim2_3);
340 y3 = zeros(1, dim2_3);
341 boxX = [345 375];
342 boxY = [220 250];
343 count = 0;
344 for i = 1:dim2_3
345     count = count + 1;
346     img = vidFrames3_3(:,:,3,i);
347     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
348     [row, col] = find(box == max(max(box)));
349     y3(i) = mean(row) + boxY(1);
350     x3(i) = mean(col) + boxX(1);
351     boxX = [round(x3(i) - 20), round(x3(i) + 20)];
352     boxY = [round(y3(i) - 20), round(y3(i) + 20)];
353 end
354 %%
355 % figure(1);
356 % for i = 1:length(img1)
357 %     imshow(img1{i});
358 %     hold on;
359 %     rectangle('position', [x1(i) - 10, y1(i) - 10, 20, 20], 'Linewidth', 2);
360 %     hold off;
361 %     pause(0.001)
362 % end
363 %%
364 % figure(2);
365 % for i = 1:length(img2)
366 %     imshow(img2{i});
367 %     hold on;
368 %     rectangle('position', [x2(i) - 10, y2(i) - 10, 20, 20], 'Linewidth', 2);
369 %     hold off;
370 %     pause(0.001)
371 % end
372 %%
373 % figure(3);
374 % for i = 1:length(img3)
375 %     imshow(img3{i});
376 %     hold on;
377 %     rectangle('position', [x3(i) - 10, y3(i) - 10, 20, 20], 'Linewidth', 2);
378 %     hold off;
379 %     pause(0.001)
380 % end
381 %%
382 x1 = x1(8:177 + 7);
383 y1 = y1(8:177 + 7);
384 x2 = x2(36:177+35);
385 y2 = y2(36:177+35);
386 x3 = x3(1:177);
387 y3 = 480 - y3(1:177);
388 %%
389 figure(2)
390 subplot(2,1,1);

```

```

391 plot(y1 - mean(y1), 'Linewidth', 3); hold on;
392 plot(y2 - mean(y2), 'Linewidth', 3)
393 plot(x3 - mean(x3), 'Linewidth', 3)
394 title('Y Coordinates of Can in Horizontal Displacement Case(Normalized)');
395 xlabel('Time');
396 ylabel('Displacement');
397 xlim([0 length(y1) + 50]);
398 legend('cam1(y1)', 'cam2(y2)', 'cam3(y3)');
399
400 subplot(2,1,2);
401 plot(x1 - mean(x1), 'Linewidth', 3); hold on;
402 plot(x2 - mean(x2), 'Linewidth', 3)
403 plot(y3 - mean(y3), 'Linewidth', 3)
404 title('X Coordinates of Can in Horizontal Displacement Case(Normalized)');
405 xlabel('Time');
406 ylabel('Displacement');
407 legend('cam1(x1)', 'cam2(x2)', 'cam3(x3)');
408 xlim([0 length(y1) + 50]);
409 %% SVD
410 X = [x1- mean(x1); y1- mean(y1); x2- mean(x2); y2- mean(y2); y3- mean(y3); x3- mean(x3)];
411 [u, s, v] = svd(X, 'econ');
412 %%
413 figure(3)
414 plot(diag(s)/sum(diag(s)), 'o', 'Linewidth', 2);
415 title('Importance of Singular Values For All Directions in Noisy Case');
416 xlabel('Singular values');
417 ylabel('Energy');
418 %%
419 %-----
420 %% TEST 4
421 clc; clear all; close all;
422 %% horizontal displacement and rotation
423 load('cam1_4.mat')
424 load('cam2_4.mat')
425 load('cam3_4.mat')
426 %%
427 [row1, col1, dim1_1, dim2_1] = size(vidFrames1_4);
428 [row2, col2, dim1_2, dim2_2] = size(vidFrames2_4);
429 [row3, col3, dim1_3, dim2_3] = size(vidFrames3_4);
430 %%
431 for k = 1:dim2_1
432     img1{k} = vidFrames1_4(:, :, :, k);
433 end
434 for k = 1:dim2_2
435     img2{k} = vidFrames2_4(:, :, :, k);
436 end
437
438 for k = 1:dim2_3
439     img3{k} = vidFrames3_4(:, :, :, k);
440 end
441 %%
442 boxX = [430 460];
443 boxY = [260 290];
444 for i = 20:70
445     img = vidFrames1_4(:, :, 3, i);
446     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
447     if (max(max(box)) == max(max(img)))
448         [row, col] = find(box == max(max(box)));
449         y1(i) = mean(row) + boxY(1);
450         x1(i) = mean(col) + boxX(1);
451         boxX = [round(x1(i) - 15), round(x1(i) + 15)];
452         boxY = [round(y1(i) - 15), round(y1(i) + 15)];
453     end
454 end
455

```

```

456 boxX = [340 370];
457 boxY = [320 350];
458 for i = 150:200
459     img = vidFrames1_4(:,:,3,i);
460     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
461     if (max(max(box)) == max(max(img)))
462         [row, col] = find(box == max(max(box)));
463         y1(i) = mean(row) + boxY(1);
464         x1(i) = mean(col) + boxX(1);
465         boxX = [round(x1(i) - 15), round(x1(i) + 15)];
466         boxY = [round(y1(i) - 15), round(y1(i) + 15)];
467     end
468 end
469 x1 = x1(y1>0);
470 y1 = y1(y1>0);
471
472 %%
473 boxX = [255 285];
474 boxY = [180 210];
475 for i = 152:200
476     img = vidFrames2_4(:,:,3,i);
477     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
478     if (max(max(box)) == max(max(img)))
479         [row, col] = find(box == max(max(box)));
480         y2(i) = mean(row) + boxY(1);
481         x2(i) = mean(col) + boxX(1);
482         boxX = [round(x2(i) - 15), round(x2(i) + 15)];
483         boxY = [round(y2(i) - 15), round(y2(i) + 15)];
484     end
485 end
486
487 boxX = [248 278];
488 boxY = [198 228];
489 for i = 235:274
490     img = vidFrames2_4(:,:,3,i);
491     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
492     if (max(max(box)) == max(max(img)))
493         [row, col] = find(box == max(max(box)));
494         y2(i) = mean(row) + boxY(1);
495         x2(i) = mean(col) + boxX(1);
496         boxX = [round(x2(i) - 15), round(x2(i) + 15)];
497         boxY = [round(y2(i) - 15), round(y2(i) + 15)];
498     else
499         y2(i) = y2(i-1);
500         x2(i) = x2(i-1);
501         boxX = [round(x2(i) - 50), round(x2(i) + 50)];
502         boxY = [round(y2(i) - 50), round(y2(i) + 50)];
503     end
504 end
505 x2 = x2(y2>0);
506 y2 = y2(y2>0);
507 %%
508 boxX = [310 340];
509 boxY = [210 240];
510 for i = 50:100
511     img = vidFrames3_4(:,:,2,i);
512     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
513     [row, col] = find(box == max(max(box)));
514     y3(i) = mean(row) + boxY(1);
515     x3(i) = mean(col) + boxX(1);
516     boxX = [round(x3(i) - 15), round(x3(i) + 15)];
517     boxY = [round(y3(i) - 15), round(y3(i) + 15)];
518 end
519 boxX = [365 395];
520 boxY = [220 250];

```

```

521 for i = 120:170
522     img = vidFrames3_4(:,:,2,i);
523     box = double(img(boxY(1):boxY(2), boxX(1):boxX(2)));
524     [row, col] = find(box == max(max(box)));
525     y3(i) = mean(row) + boxY(1);
526     x3(i) = mean(col) + boxX(1);
527     boxX = [round(x3(i) - 15), round(x3(i) + 15)];
528     boxY = [round(y3(i) - 15), round(y3(i) + 15)];
529 end
530 y3 = x3(x3>0);
531 x3 = x3(x3>0);
532 y3 = [y3(10:40) y3(63:end)];
533 x3 = [x3(10:40) x3(63:end)];
534 %%
535 x1 = x1(5:length(x3));
536 y1 = y1(5:length(x3));
537 x2 = x2(1:length(x3) - 4);
538 y2 = y2(1:length(x3) - 4);
539 x3 = x3(5:length(x3));
540 y3 = 480 - y3(1:length(x3));
541 %%
542 figure(2)
543 subplot(2,1,1);
544 plot(y1 - mean(y1), 'Linewidth', 3); hold on;
545 plot(y2 - mean(y2), 'Linewidth', 3);
546 plot(x3 - mean(x3), 'Linewidth', 3);
547 title('Y Coordinates of Can in Horizontal Displacement and Rotation Case(Normalized)');
548 xlabel('Time');
549 ylabel('Displacement');
550 xlim([0 length(y1) + 10]);
551 legend('cam1(y1)', 'cam2(y2)', 'cam3(y3)');
552
553 subplot(2,1,2);
554 plot(x1 - mean(x1), 'Linewidth', 3); hold on;
555 plot(x2 - mean(x2), 'Linewidth', 3);
556 plot(y3 - mean(y3), 'Linewidth', 3);
557 title('X Coordinates of Can in Horizontal Displacement and Rotation Case(Normalized)');
558 xlabel('Time');
559 ylabel('Displacement');
560 legend('cam1(x1)', 'cam2(x2)', 'cam3(x3)');
561 xlim([0 length(y1) + 10]);
562 %% SVD
563 X = [x1- mean(x1); y1- mean(y1); x2- mean(x2); y2- mean(y2); y3- mean(y3); x3- mean(x3)];
564 [u, s, v] = svd(X, 'econ');
565 %%
566 figure(3)
567 plot(diag(s)/sum(diag(s)), 'o', 'Linewidth', 2);
568 title('Importance of Singular Values For All Directions in Noisy Case');
569 xlabel('Singular values');
570 ylabel('Energy');

```