

AMATH 582 Homework 3

Christopher Liu

February 21, 2020

Abstract

Principal component analysis (PCA) is a powerful tool in allowing us to identify dynamics of interest in a system. We have been presented with recordings of a paint can moving on a spring from 3 different cameras for 4 different test scenarios. PCA will be employed to empirically extract the governing equations of motion of the paint can by studying the linear proper orthogonal modes of the system. By analyzing the footage, we will highlight the strengths of PCA in identifying key dynamics as well as demonstrate how PCA can fail when the data has high levels of noise.

1 Introduction and Overview

For this assignment, we have been tasked with analyzing the movement of a paint can on an spring recorded on 3 different cameras from 3 distinct angles. Utilizing the principal component analysis (PCA), we will then be able to empirically extract the governing equations of motion of the paint can.

The behavior of the paint can is altered in 4 different tests that were conducted. In doing so, we hope to highlight the strengths and limitations of PCA. Test 1 is the ideal case where the paint can is oscillating up and down in 1 dimension. Test 2 is similar to the ideal case but instead has noise generated by shaking the camera while it is recording. Test 3 includes the addition of horizontal movement to the paint can. Finally, Test 4 has both vertical and horizontal displacement and rotation.

2 Theoretical Background

2.1 Singular Value Decomposition

Every matrix $A \in m \times n$ has a singular value decomposition (SVD) given by Equation (1) [1].

$$A = U\Sigma V^* \quad (1)$$

where $U \in m \times m$ is unitary, $V \in n \times n$ is unitary and $\Sigma \in m \times n$ is diagonal. Furthermore, σ_j , the components of Σ are uniquely determined for all A . The SVD allows us to describe multiplication of a matrix by a vector as a unitary transformation (V^*) followed by a stretch (Σ) and another unitary transformation (U)

2.2 Calculating the SVD

The SVD is easy to compute if we consider the following matrices AA^T and $A^T A$,

$$A^T A = (U\Sigma V^*)^T (U\Sigma V^*) = V\Sigma^2 V^* \quad (2)$$

$$AA^T = (U\Sigma V^*)(U\Sigma V^*)^T = U\Sigma^2 U^* \quad (3)$$

Multiplying (2) and (3) on the right by V and U respectively, we are presented with 2 eigenvalue problems

$$A^T A V = V \Sigma^2 \quad (4)$$

$$A A^T U = U \Sigma^2 \quad (5)$$

So if we find the normalized eigenvectors of AA^T and $A^T A$, we are able to construct U and V and the square-root of the eigenvalues will be the singular values.

2.3 Relationship between SVD and Variance

The covariance matrix of A , C_A , is a square symmetric $m \times m$ where $A \in m \times n$ contains the measured data of the system given by 6 with n measurements and m measuring positions. The diagonal terms of C_A are the variances for particular measurements. We are interested when the variances are large as they are assumed to correspond to the dynamics of interest. The off-diagonal elements are the covariances between measurements. A large covariance represents a high level of redundancy between the two measurements and vice versa. By using the covariance matrix, we can identify the dynamics of interest corresponding to maximal variance and remove redundancy [1].

$$C_A = \frac{1}{n-1} AA^T \quad (6)$$

We can relate the SVD and the covariance matrix in the following way,

$$C_Y = \frac{1}{n-1} AA^T = \frac{1}{n-1} \Sigma^2 \quad (7)$$

where $A = U\Sigma V^*$ and $Y = U^*A$. So by calculating the SVD of A we can determine a relative relationship between the variances of each transformed measurement (ie. which ones are much larger than the others). Furthermore, the columns of U contain the linear proper orthogonal modes (POD) which constitute the orthonormal expansion basis of interest and the columns of V show their evolution in time.

3 Algorithm Implementation and Development

There are 2 main steps that are taken to carry out the analysis. The first is to track the movement of the paint can with each frame in 2 dimensions for each camera and test. The second is to combine the vectors of the paint can's position with each frame into a matrix and to compute its SVD.

A pink flashlight is placed on top of the can and allows us to track the movement in each frame. Depending on the recording and test, we can track either the light emitted by the flashlight (in color or black and white) or its pink body. In order to use our algorithm for tracking the paint can, we will need a few initial parameters. Namely, an initial guess, window width, length of recording in frames, and what color/object to track. The initial guess and the choice of object to track is determined by watching the original footage and the window width is guessed and refined with trial and error. After finding the initial parameters, we can use our algorithm to track the can which is described in detail in Algorithm 1.

Algorithm 1: Tracking the Paint Can

```

Set center of search window to initial guess
Set max intensity of pixel to 0
for  $i = 2$  : (length of recording in frames) do
    Extract  $i$ th frame of recording
    for  $x = (x \text{ center} - \text{width}) : (x \text{ center} + \text{width})$  do
        for  $y = (y \text{ center} - \text{width}) : (y \text{ center} + \text{width})$  do
            if intensity of  $(x, y) > (\text{max intensity})$  then
                Set max intensity to intensity at  $(x, y)$ 
            end if
        end for
    end for
    Set center of search window to  $(x, y)$  of max intensity
    Store location of max intensity
end for

```

3 different functions were defined to track the paint can. They follow the same logic as 1 but look at different intensity values corresponding to white, when the recording is in color or black and white, and pink. After we measure the positions, we plot their positions to center the peaks and troughs of the oscillation and truncate them so that they are of the same length. We construct the measurement matrix $A \in \mathbb{R}^{6 \times n}$ where n is the number of frames and each row corresponds to the X or Y coordinate of the 3 cameras. We will also subtract the mean off each vector to center the data at 0. Finally, we take the SVD of A^T as it is more conventional to solve an over-determined system. Note this flips the interpretation of U and V

4 Computational Results

4.1 Test 1: Ideal Case

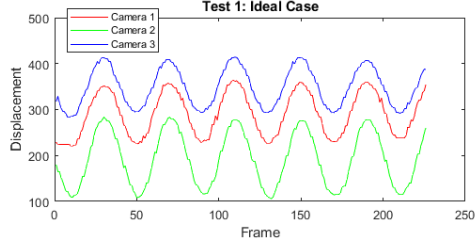


Figure 1: Tracked positions for test 1

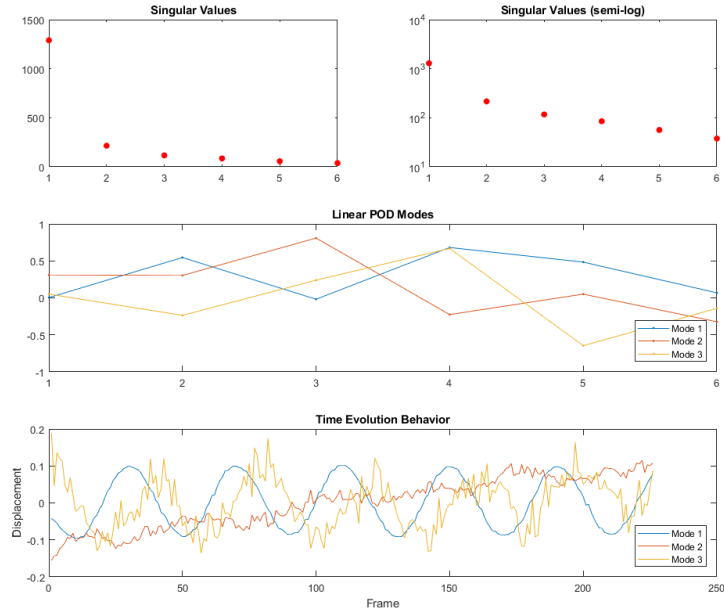


Figure 2: The singular values, linear PODs and the time evolution for Test 1

For the ideal case, we find that our tracking algorithm does a fairly good job. There is 1 large singular value which is what we expect since the paint can is moving in 1-D. The first mode corresponding to the first singular value accounts for 71% of the energy in the system. If we look at Mode 1 in the Linear POD Mode plot, we see that each camera roughly only has one non-zero component which corresponds to movement in 1-D. Finally, looking at the time evolution behavior, we see that mode 1 exhibits simple harmonic motion which is again what we would expect for Test 1.

4.2 Test 2: Noisy Case

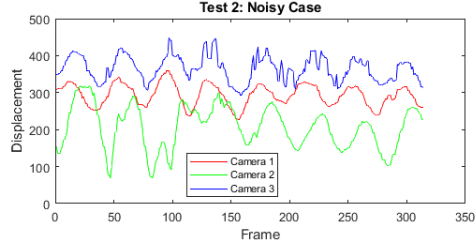


Figure 3: Tracked positions for test 2

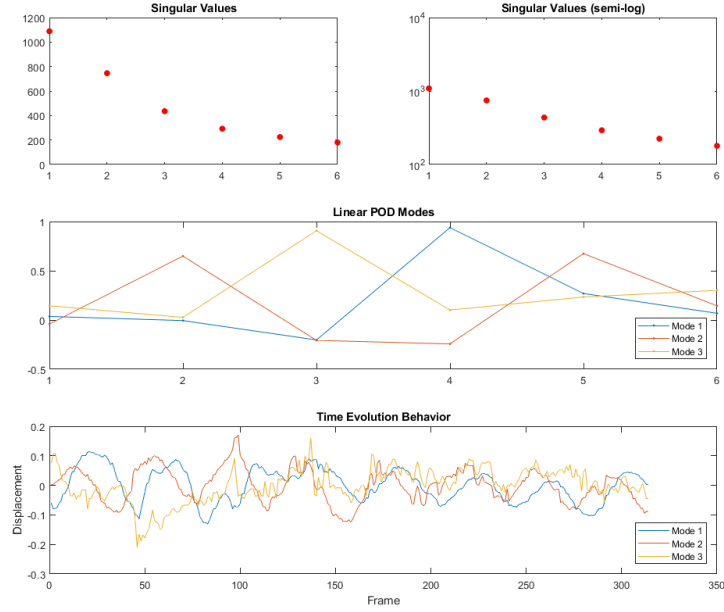


Figure 4: The singular values, linear PODs and the time evolution for Test 2

The tracking algorithm does not perform as well for Test 2 but this is expected as this is the noisy case. Nonetheless, simple harmonic motion can still be made out. The addition of noise has increased the values of the last 5 singular values but there is still 1 large value (37% of energy). For the linear modes and the time evolution behavior, it is no longer apparent that the paint can is only moving in 1-D. The noise obscures the dynamics of the system.

4.3 Test 3: Horizontal Displacement

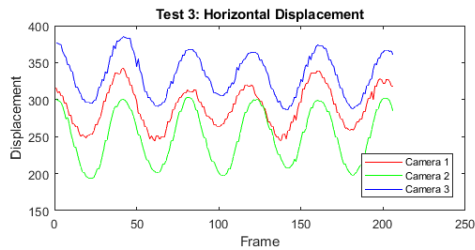


Figure 5: Tracked positions for test 3

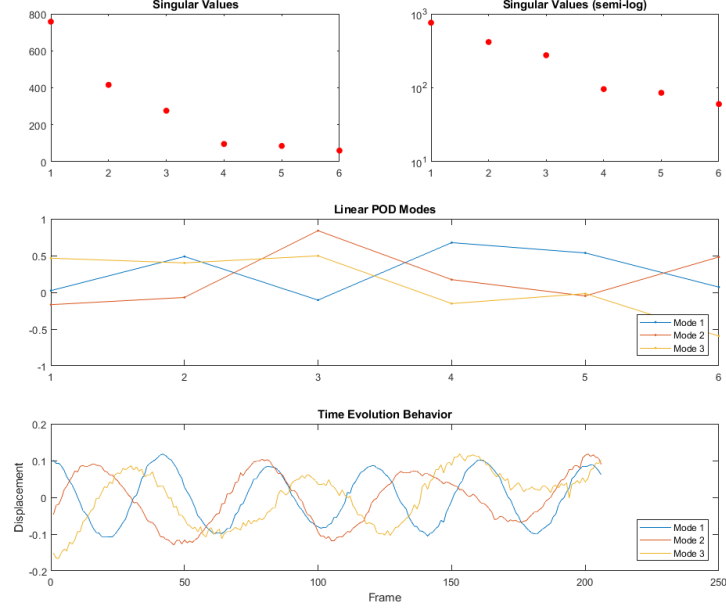


Figure 6: The singular values, linear PODs and the time evolution for Test 3

Our tracking algorithm seems to do well even with the addition of the horizontal displacement. When inspecting the original footage, the addition of horizontal displacement has added displacement in X and Y directions. The camera angle and an imperfect initial push gives us movement in 3-D. This is reflected in the singular values where the largest one corresponds to the harmonic motion in the Z-direction and the next 2 correspond to X and Y directions. The first mode accounts for 45% of the energy while the first 3 modes account for 85%. The singular values tell us that the simple harmonic motion in Z is still the dominant behavior. Looking at the time evolution of the modes, we can now observe oscillatory behavior in modes 2 and 3 in addition to the harmonic motion in mode 1.

4.4 Test 4: Horizontal Displacement and Rotation

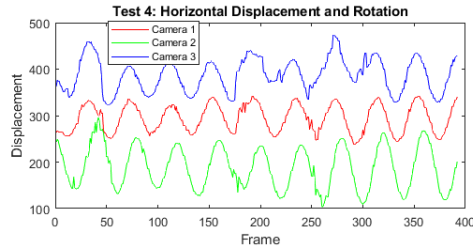


Figure 7: Tracked positions for test 4

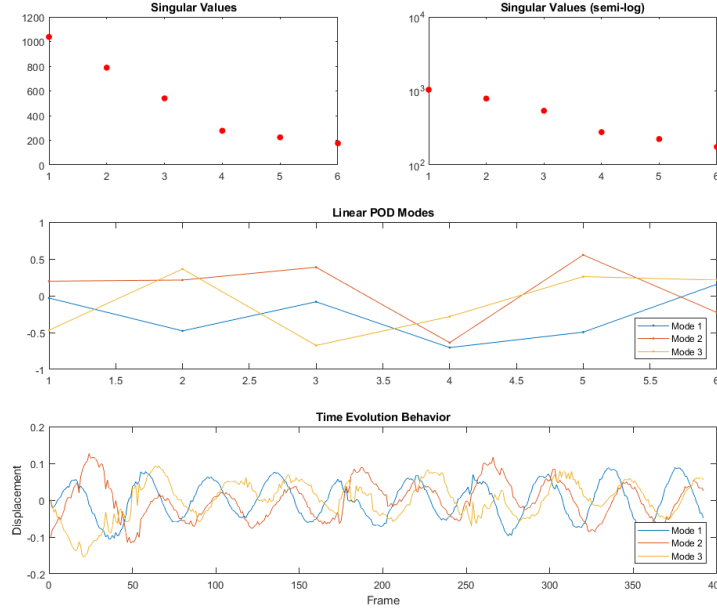


Figure 8: The singular values, linear PODs and the time evolution for Test 4

The tracking algorithm struggles a little more with Test 4 but still produces decent results. Upon inspection of the footage, we observe that the paint can itself is rotating which makes tracking the flashlight more difficult. When the flashlight was not visible, the tracker would instead track the bottom of the paint can. Again, as expected we have 3 large singular values corresponding to displacement in 3-D. The first mode accounts for 34% of the energy while the first 3 modes account for 78%. The harmonic oscillation in the Z-direction is still the dominant behavior of the system. The time evolution also reflects oscillation in 3-D however it is a little 'noisy' due to difficulties in tracking the flashlight.

5 Summary and Conclusions

Through the use of PCA, we have been able to empirically extract the governing equations of motion of the paint can. We were able to confirm our findings by watching the original footage and as such have been able to highlight the strength of PCA to extract meaningful characteristics from data. From analyzing a noisy data set, we were also able to show the limitations of PCA when working with imperfect data.

References

- [1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Below are the key MATLAB functions implemented.

- `imshow(vidFrames)` displays the picture given by the RGB values in a figure
- `[U,S,V]=svd(A,0)` returns the reduced singular value decomposition of A
- `rgb2gray(vidFrames)` converts an RGB image to a grayscale image.
- `diag(S)` returns the diagonal elements of S in a vector

Appendix B MATLAB Code

Github Repo: <https://github.com/chrisnhl/AMATH582/tree/master/Homework3>

```
1 clear all; clc ;close all;
2
3 load( 'cam1_1.mat' )
4 load( 'cam1_2.mat' )
5 load( 'cam1_3.mat' )
6 load( 'cam1_4.mat' )
7 load( 'cam2_1.mat' )
8 load( 'cam2_2.mat' )
9 load( 'cam2_3.mat' )
10 load( 'cam2_4.mat' )
11 load( 'cam3_1.mat' )
12 load( 'cam3_2.mat' )
13 load( 'cam3_3.mat' )
14 load( 'cam3_4.mat' )
15
16 %%
17 % Tracking for camera 1 part 1
18 [x1_1, y1_1] = track_paintcan_gray(vidFrames1_1,[321,228],[20,20,20,20]);
19
20 %%
21 % Playing Camera 1 Part 1
22 for i=1:226
23     figure(1)
24     imshow(vidFrames1_1(:,:,:,i))
25     hold on
26     plot(x1_1(i),y1_1(i),'r*')
27     hold off
28 end
29
30 %%
31 % Tracking for camera 2 part 1
32 [x2_1, y2_1] = track_paintcan_gray(vidFrames2_1,[274,274],[20,20,20,20]);
33
34 %%
35 % Play Camera 2 Part 1
36 for i=1:284
37     figure(2)
38     imshow(uint8(vidFrames2_1(:,:,:,i)))
39     hold on
40     plot(x2_1(i),y2_1(i),'r*')
41     hold off
42 end
43
44 %%
45 % Tracking for camera 3 part 1
46 [x3_1, y3_1] = track_paintcan_gray(vidFrames3_1,[318,271],[15,15,15,15]);
47
48 %%
49 % Play Camera 3 Part 1
50 for i=1:232
51     figure(2)
52     imshow(uint8(vidFrames3_1(:,:,:,i)))
53     hold on
```

```

54     plot(x3_1(i),y3_1(i),'r*')
55     hold off
56 end
57
58 %%
59 %Plot position of all 3 cameras, test 1
60 close all
61
62 figure(3)
63
64 subplot(3,1,1)
65 plot(1:226,y1_1,'r')
66 hold on
67 plot(1:284,y2_1,'g')
68 plot(1:232,x3_1,'b')
69 hold off
70 legend('Camera 1', 'Camera 2', 'Camera 3')
71
72 %truncated
73 subplot(3,1,2)
74 plot(1:226,y1_1,'r')
75 hold on
76 plot(1:226,y2_1(10:235),'g')
77 plot(1:226,x3_1(1:226),'b')
78 hold off
79 legend('Camera 1', 'Camera 2', 'Camera 3')
80 title('Truncated')
81 %%
82 % SVD part 1
83 A1(1,:) = x1_1 - mean(x1_1);
84 A1(2,:) = y1_1 - mean(y1_1);
85 A1(3,:) = x2_1(10:235) - mean(x2_1(10:235));
86 A1(4,:) = y2_1(10:235) - mean(y2_1(10:235));
87 A1(5,:) = x3_1(1:226) - mean(x3_1(1:226));
88 A1(6,:) = y3_1(1:226) - mean(y3_1(1:226));
89 [U1,s1,V1] = svd(A1.',0);
90
91 sig1 = diag(s1);
92 energy1_1 = sig1(1)/sum(sig1);
93 energy2_1 = sum(sig1(1:2))/sum(sig1);
94 energy3_1 = sum(sig1(1:3))/sum(sig1);
95
96 figure(5)
97 subplot(3,2,1)
98 plot(1:length(sig1),sig1,'r.','MarkerSize',20)
99 title('Singular Values')
100 subplot(3,2,2)
101 plot(1:length(sig1),sig1,'r.','MarkerSize',20)
102 set(gca, 'YScale', 'log')
103 title('Singular Values (semi-log)')
104
105 subplot(3,1,2)
106 plot(1:6,V1(:,1),'.-',1:6,V1(:,2),'.-',1:6,V1(:,3),'.-')
107 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
108 xticks([1 2 3 4 5 6])
109 title('Linear POD Modes')

```



```

110
111 subplot(3,1,3)
112 plot(1:226,U1(:,1),1:226,U1(:,2),1:226,U1(:,3))
113 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
114 ylabel('Displacement')
115 xlabel('Frame')
116 title('Time Evolution Behavior')
117 %%
118 % Tracking for camera 1 part 2
119 [x1_2, y1_2] = track_paintcan(vidFrames1_2,[325,308],[17,17*0.9,15,15*0.9]);
120
121 %%
122 % Tracking Camera 1 Part 2
123 for i=1:314
124     figure(2)
125     imshow(uint8(vidFrames1_2(:,:,i)))
126     hold on
127     plot(x1_2(i),y1_2(i),'r*')
128     hold off
129 end
130
131 %%
132 % Tracking for camera 2 part 2
133 [x2_2, y2_2] = track_paintcan(vidFrames2_2,[314,357],[34,34,34,35]);
134
135 %%
136 % Tracking Camera 2 Part 2
137 for i=1:356
138     figure(2)
139     imshow(uint8(vidFrames2_2(:,:,i)))
140     hold on
141     plot(x2_2(i),y2_2(i),'r*')
142     hold off
143 end
144
145 %%
146 % Tracking for camera 3 part 2
147 [x3_2, y3_2] = track_paintcan_gray(vidFrames3_2,[349,245],[44,45,44,44]);
148
149 %%
150 % Tracking Camera 3 Part 2
151 for i=1:327
152     figure(2)
153     imshow(uint8(vidFrames3_2(:,:,i)))
154     hold on
155     plot(x3_2(i),y3_2(i),'r*')
156     hold off
157 end
158
159 %%
160 %Plot position of all 3 cameras part 2
161 close all
162
163 figure(5)
164
165 subplot(2,1,1)

```

```

166 plot(1:314,y1_2,'r')
167 hold on
168 plot(1:356,y2_2,'g')
169 plot(1:327,x3_2,'b')
170 hold off
171 legend('Camera 1','Camera 2','Camera 3')
172
173 %%truncated
174 subplot(2,1,2)
175 plot(1:314,y1_2,'r')
176 hold on
177 plot(1:314,y2_2(15:328),'g')
178 plot(1:314,x3_2(1:314),'b')
179 hold off
180 legend('Camera 1','Camera 2','Camera 3')
181 title('Truncated')
182
183 %%
184 % SVD part 2
185 A2 = zeros(6,314);
186 A2(1,:) = x1_2 - mean(x1_2);
187 A2(2,:) = y1_2 - mean(y1_2);
188 A2(3,:) = x2_2(15:328) - mean(x2_2(15:328));
189 A2(4,:) = y2_2(15:328) - mean(y2_2(15:328));
190 A2(5,:) = x3_2(1:314) - mean(x3_2(1:314));
191 A2(6,:) = y3_2(1:314) - mean(y3_2(1:314));
192 [U2,s2,V2] = svd(A2.',0);
193
194 sig2 = diag(s2);
195 energy1_2 = sig2(1)/sum(sig2);
196 energy2_2 = sum(sig2(1:2))/sum(sig2);
197 energy3_2 = sum(sig2(1:3))/sum(sig2);
198
199 figure(6)
200 subplot(3,2,1)
201 plot(1:length(sig2),sig2,'r.','MarkerSize',20)
202 title('Singular Values')
203 subplot(3,2,2)
204 plot(1:length(sig2),sig2,'r.','MarkerSize',20)
205 set(gca,'YScale','log')
206 title('Singular Values (semi-log)')
207
208 subplot(3,1,2)
209 plot(1:6,V2(:,1),'.-',1:6,V2(:,2),'.-',1:6,V2(:,3),'.-')
210 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
211 xticks([1 2 3 4 5 6])
212 title('Linear POD Modes')
213
214 subplot(3,1,3)
215 plot(1:314,U2(:,1),1:314,U2(:,2),1:314,U2(:,3))
216 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
217 ylabel('Displacement')
218 xlabel('Frame')
219 title('Time Evolution Behavior')
220
221 %%

```

```

222 % Tracking for camera 1 part 3, tracking pink
223
224 [x1_3, y1_3] = track_paintcan_pink(vidFrames1_3,[330,290],[20,20,20,20]);
225 %%
226 % Tracking Camera 1 Part 3
227 for i=1:239
228     figure(2)
229     imshow(uint8(vidFrames1_3(:,:,i)))
230     hold on
231     plot(x1_3(i),y1_3(i),'r*')
232     hold off
233 end
234
235 %%
236 % Tracking for camera 2 part 3, tracking pink
237
238 [x2_3, y2_3] = track_paintcan_pink(vidFrames2_3,[252,294],[20,20,20,20]);
239 %%
240 % Tracking Camera 2 Part 3
241 for i=1:281
242     figure(2)
243     imshow(uint8(vidFrames2_3(:,:,i)))
244     hold on
245     plot(x2_3(i),y2_3(i),'r*')
246     hold off
247 end
248
249 %%
250 % Tracking for camera 3 part 3
251 [x3_3, y3_3] = track_paintcan_gray(vidFrames3_3,[353,229],[17,17,17,17]);
252
253 %%
254 % Tracking Camera 3 Part 3
255 for i=1:237
256     figure(2)
257     imshow(uint8(vidFrames3_3(:,:,i)))
258     hold on
259     plot(x3_3(i),y3_3(i),'r*')
260     hold off
261 end
262
263 %%
264 %Plot position of all 3 cameras part 3
265 close all
266
267 figure(7)
268
269 subplot(2,1,1)
270 plot(1:239,y1_3,'r')
271 hold on
272 plot(1:281,y2_3,'g')
273 plot(1:237,x3_3,'b')
274 hold off
275 legend('Camera 1', 'Camera 2', 'Camera 3')
276
277 %truncated

```

```

278 subplot(2,1,2)
279 plot(1:206,y1_3(20:225),'r')
280 hold on
281 plot(1:206,y2_3(5:210),'g')
282 plot(1:206,x3_3(10:215),'b')
283 hold off
284 legend('Camera 1','Camera 2','Camera 3')
285 title('Truncated')
286
287 %%
288 % SVD part 3
289 A3 = zeros(6,206);
290 A3(1,:) = x1_3(20:225) - mean(x1_3(20:225));
291 A3(2,:) = y1_3(20:225) - mean(y1_3(20:225));
292 A3(3,:) = x2_3(5:210) - mean(x2_3(5:210));
293 A3(4,:) = y2_3(5:210) - mean(y2_3(5:210));
294 A3(5,:) = x3_3(10:215) - mean(x3_3(10:215));
295 A3(6,:) = y3_3(10:215) - mean(y3_3(10:215));
296 [U3,s3,V3] = svd(A3.',0);
297 sig3 = diag(s3);
298
299 energy1_3 = sig3(1)/sum(sig3);
300 energy2_3 = sum(sig3(1:2))/sum(sig3);
301 energy3_3 = sum(sig3(1:3))/sum(sig3);
302
303 figure(8)
304 subplot(3,2,1)
305 plot(1:length(sig3),sig3,'r.','MarkerSize',20)
306 title('Singular Values')
307 subplot(3,2,2)
308 plot(1:length(sig3),sig3,'r.','MarkerSize',20)
309 set(gca,'YScale','log')
310 title('Singular Values (semi-log)')
311
312 subplot(3,1,2)
313 plot(1:6,V3(:,1),'.-',1:6,V3(:,2),'.-',1:6,V3(:,3),'.-')
314 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
315 xticks([1 2 3 4 5 6])
316 title('Linear POD Modes')
317
318 subplot(3,1,3)
319 plot(1:206,U3(:,1),1:206,U3(:,2),1:206,U3(:,3))
320 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
321 ylabel('Displacement')
322 xlabel('Frame')
323 title('Time Evolution Behavior')
324
325
326 %%
327 % Tracking for camera 1 part 4
328 [x1_4, y1_4] = track_paintcan_pink(vidFrames1_4,[402,263],[44,45,44,44]);
329
330 %%
331 % Tracking Camera 1 Part 4
332 for i=1:392
333     figure(2)

```

```

334     imshow(uint8(vidFrames1_4(:,:, :, i)))
335     hold on
336     plot(x1_4(i), y1_4(i), 'r*')
337     hold off
338 end
339
340 %%
341 % Tracking for camera 2 part 4
342 [x2_4, y2_4] = track_paintcan_pink(vidFrames2_4,[244,243],[44,45,44,44]);
343
344 %%
345 % Tracking Camera 2 Part 4
346 for i=1:405
347     figure(2)
348     imshow(uint8(vidFrames2_4(:,:, :, i)))
349     hold on
350     plot(x2_4(i), y2_4(i), 'r*')
351     hold off
352 end
353
354 %%
355 % Tracking for camera 3 part 4
356 [x3_4, y3_4] = track_paintcan(vidFrames3_4,[363,244],[40,40,30,30]);
357
358 %%
359 % Tracking Camera 3 Part 4
360 for i=1:394
361     figure(2)
362     imshow(uint8(vidFrames3_4(:,:, :, i)))
363     hold on
364     plot(x3_4(i), y3_4(i), 'r*')
365     hold off
366 end
367
368 %%
369 %Plot position of all 3 cameras part 4
370 close all
371
372 figure(9)
373
374 subplot(2,1,1)
375 plot(1:392,y1_4, 'r')
376 hold on
377 plot(1:405,y2_4, 'g')
378 plot(1:394,x3_4, 'b')
379 hold off
380 legend('Camera 1', 'Camera 2', 'Camera 3')
381
382 %truncated
383 subplot(2,1,2)
384 plot(1:392,y1_4, 'r')
385 hold on
386 plot(1:392, y2_4(1:392), 'g')
387 plot(1:392, x3_4(1:392), 'b')
388 hold off
389 legend('Camera 1', 'Camera 2', 'Camera 3')

```

```

390 title('Truncated')
391
392 %%
393 % SVD part 4
394 A4 = zeros(6,392);
395 A4(1,:) = x1_4 - mean(x1_4);
396 A4(2,:) = y1_4 - mean(y1_4);
397 A4(3,:) = x2_4(1:392) - mean(x2_4(1:392));
398 A4(4,:) = y2_4(1:392) - mean(y2_4(1:392));
399 A4(5,:) = x3_4(1:392) - mean(x3_4(1:392));
400 A4(6,:) = y3_4(1:392) - mean(y3_4(1:392));
401 [U4,s4,V4] = svd(A4.',0);
402
403 sig4 = diag(s4);
404 energy1_4 = sig4(1)/sum(sig4);
405 energy2_4 = sum(sig4(1:2))/sum(sig4);
406 energy3_4 = sum(sig4(1:3))/sum(sig4);
407
408 figure(10)
409 subplot(3,2,1)
410 plot(1:length(sig4),sig4,'r.','MarkerSize',20)
411 title('Singular Values')
412 subplot(3,2,2)
413 plot(1:length(sig4),sig4,'r.','MarkerSize',20)
414 title('Singular Values (semi-log)')
415 set(gca, 'YScale', 'log')
416
417 subplot(3,1,2)
418 plot(1:6,V4(:,1),'.-',1:6,V4(:,2),'.-',1:6,V4(:,3),'.-')
419 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
420 title('Linear POD Modes')
421
422 subplot(3,1,3)
423 plot(1:392,U4(:,1),1:392,U4(:,2),1:392,U4(:,3))
424 legend('Mode 1','Mode 2','Mode 3','Location','southeast')
425 ylabel('Displacement')
426 xlabel('Frame')
427 title('Time Evolution Behavior')
428
429 %%
430 % Plot all position vectors
431
432 figure(11)
433 plot(1:226,y1_1,'r')
434 hold on
435 plot(1:226,y2_1(10:235),'g')
436 plot(1:226,x3_1(1:226),'b')
437 hold off
438 title('Test 1: Ideal Case')
439 lgd = legend('Camera 1', 'Camera 2', 'Camera 3','Location','southeast');
440 lgd.FontSize =7;
441 ylabel('Displacement')
442 xlabel('Frame')
443
444 figure(12)
445 plot(1:314,y1_2,'r')

```

```

446 hold on
447 plot(1:314, y2_2(15:328), 'g')
448 plot(1:314, x3_2(1:314), 'b')
449 lgd = legend('Camera 1', 'Camera 2', 'Camera 3', 'Location', 'southeast');
450 lgd.FontSize = 7;
451 hold off
452 title('Test 2: Noisy Case')
453 ylabel('Displacement')
454 xlabel('Frame')
455
456
457 figure(13)
458 plot(1:206, y1_3(20:225), 'r')
459 hold on
460 plot(1:206, y2_3(5:210), 'g')
461 plot(1:206, x3_3(10:215), 'b')
462 hold off
463 title('Test 3: Horizontal Displacement')
464 ylabel('Displacement')
465 xlabel('Frame')
466 lgd = legend('Camera 1', 'Camera 2', 'Camera 3', 'Location', 'southeast');
467 lgd.FontSize = 7;
468
469 figure(14)
470 plot(1:392, y1_4, 'r')
471 hold on
472 plot(1:392, y2_4(1:392), 'g')
473 plot(1:392, x3_4(1:392), 'b')
474 hold off
475 title('Test 4: Horizontal Displacement and Rotation')
476 ylabel('Displacement')
477 xlabel('Frame')
478 lgd = legend('Camera 1', 'Camera 2', 'Camera 3', 'Location', 'southeast');
479 lgd.FontSize = 7;

```

B.1 Additional Functions

```

1 %vidFrames is the .dat file corresponding to the recording
2 %guess is the initial guess of the flashlight in [x y] from frame 1
3 %width is the search width of the tracking algorithm
4 %returns vectors xp and yp which are the positions of the paintcan with
5 % each frame
6 function [xp,yp] = track_paintcan(vidFrames, guess, width)
7 dim = size(vidFrames);
8 L = dim(4); %length of video in frames
9 xp = zeros(L,1);
10 yp = zeros(L,1);
11 maxloc = [0,0];
12 maxval = 0;
13
14 init_guess = guess;
15 xp(1) = init_guess(1);
16 yp(1) = init_guess(2);
17
18 %center the first search window to the initial point
19 center = init_guess;

```

```

20
21 for f = 2:L
22     frame = double(vidFrames(:,:, :, f));
23     for x = center(1)-width(1):center(1)+width(2)
24         for y = center(2)-width(3):center(2)+width(4)
25             point = frame(y,x,:);
26             colorsum = point(1,1,1)+point(1,1,2)+point(1,1,3);
27             if colorsum > maxval
28                 maxloc = [x,y];
29                 maxval = colorsum;
30             end
31         end
32     end
33     xp(f) = maxloc(1);
34     yp(f) = maxloc(2);
35     maxloc = [0,0];
36     maxval = 0;
37
38     center(1) = xp(f);
39     center(2) = yp(f);
40 end

1 %vidFrames is the .dat file corresponding to the recording
2 %guess is the initial guess of the flashlight in [x y] from frame 1
3 %width is the search width of the tracking algorithm
4 %returns vectors xp and yp which are the positions of the paintcan with
5 % each frame
6 function [xp,yp] = track_paintcan_gray(vidFrames,guess,width)
7 dim = size(vidFrames);
8 L = dim(4); %length of video in frames
9 xp = zeros(L,1);
10 yp = zeros(L,1);
11 maxloc = [0,0];
12 maxval = 0;
13
14 init_guess = guess;
15 xp(1) = init_guess(1);
16 yp(1) = init_guess(2);
17
18 %center the first search window to the initial point
19 center = init_guess;
20
21 for f = 2:L
22     frame = double(rgb2gray(vidFrames(:,:, :, f)));
23     for x = center(1)-width(1):center(1)+width(2)
24         for y = center(2)-width(3):center(2)+width(4)
25             point = frame(y,x,:);
26             if point > maxval
27                 maxloc = [x,y];
28                 maxval = point;
29             end
30         end
31     end
32     xp(f) = maxloc(1);
33     yp(f) = maxloc(2);
34     maxloc = [0,0];
35     maxval = 0;

```



```

36
37     center(1) = xp(f);
38     center(2) = yp(f);
39 end

1 %vidFrames is the .dat file corresponding to the recording
2 %guess is the initial guess of the flashlight in [x y] from frame 1
3 %width is the search width of the tracking algorithm
4 %returns vectors xp and yp which are the positions of the paintcan with
5 % each frame
6 function [xp,yp] = track_paintcan_pink(vidFrames,guess,width)
7 dim = size(vidFrames);
8 L = dim(4); %length of video in frames
9 xp = zeros(L,1);
10 yp = zeros(L,1);
11 maxloc = [0,0];
12 maxred = 0;
13
14 init_guess = guess;
15 xp(1) = init_guess(1);
16 yp(1) = init_guess(2);
17
18 %center the first search window to the initial point
19 center = init_guess;
20
21 for f = 2:L
22     frame = double(vidFrames(:,:,f));
23     for x = center(1)-width(1):center(1)+width(2)
24         for y = center(2)-width(3):center(2)+width(4)
25             point = frame(y,x,:);
26             if ((point(1,1,1) > maxred)&(point(1,1,2) < 150)&(point(1,1,3) < 150))
27                 maxloc = [x,y];
28                 maxred = point;
29             end
30         end
31     end
32     xp(f) = maxloc(1);
33     yp(f) = maxloc(2);
34     maxloc = [0,0];
35     maxred = 0;
36
37     center(1) = xp(f);
38     center(2) = yp(f);
39 end

```