

AMATH 582 Homework 3

Reed Nomura

February 21, 2020

Abstract

This paper explores the use of principal component analysis in order to evaluate movement within footage. Specifically we will be analyzing four tests of a paint can suspended on a spring. Each test is filmed by three different angles. The paint can will be tracked in each shot and then we will use singular value decomposition to clarify what we are seeing in the data collected.

1 Introduction and Overview

Principal component analysis is a procedure used to transform data into orthogonal dimensions in which our data exists. These dimensions provide important insight into the coordinate system in which our data exists and is best interpreted. In this paper we will be using principal component analysis to understand the movement of a paint can suspended by a spring. We will be examining four different tests. The first test consists of the can simply bouncing up and down on the spring. The second test is similar to the first with some added noise by shaking the cameras as the experiment is being filmed. The first test adds another dimension by swinging the paint can side to side while it is also bouncing up and down. Finally, the fourth test involves having the paint can spin while simultaneously bouncing up and down and swinging side to side. Each of these tests are filmed by three different angles to provide differing points of observation.

2 Theoretical Background

2.1 Singular Value Decomposition (SVD)

A singular value decomposition (SVD) is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications.[1]. SVD decomposition takes on the form

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

with

$$U \in \mathbb{C}^{m \times m} \text{ is unitary} \quad (2)$$

$$V \in \mathbb{C}^{n \times n} \text{ is unitary} \quad (3)$$

$$\Sigma \in \mathbb{R}^{m \times n} \text{ is diagonal} \quad (4)$$

2.2 Eigenvector Decomposition

The most straightforward way to diagonalize the covariance matrix is by making the observation that XX^T is a square, symmetric $m \times m$ matrix, i.e. it is self-adjoint so that the m eigenvalues are real and distinct. [1] Linear algebra provides theorems which state that such a matrix can be rewritten as

$$XX^T = SAS^{-1} = S\Lambda S^T \quad (5)$$

Instead of working with X , we are able to work with

$$Y = S^T X \quad (6)$$

It then follows that

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

$$= \frac{1}{n-1} (S^T X)(S^T X)^T \quad (8)$$

$$= \frac{1}{n-1} (S^T X)(X^T S) \quad (9)$$

$$= \frac{1}{n-1} S^T XX^T S \quad (10)$$

$$= \frac{1}{n-1} S^T \Lambda S^T S \quad (11)$$

Where

$$\Lambda = \begin{bmatrix} \Lambda_1 & & & \\ & \Lambda_2 & & \\ & & \ddots & \\ & & & \Lambda_n \end{bmatrix} \quad (12)$$

Hence

$$C_Y = \frac{1}{n-1} \Lambda \quad (13)$$

In this basis, the principal components are the eigenvectors of XX^T with the interpretation that the j th diagonal value of CY is the variance of X along x_j , the j th column of S [1].

2.3 SVD for Diagonalizing Covariance Matrix

A second method for diagonalizing the co-variance matrix is the SVD method. In this case, the SVD can diagonalize any matrix by working in the appropriate pair of bases U and V as outlined in the first lecture of this section. Thus by defining the transformed variable

$$Y = U^* X \quad (14)$$

where U is the unitary transformation associated with the SVD: $X = U \Sigma V^*$. Just as in the eigenvalue/eigenvector formulation, we then compute the variance in Y :

$$C_Y = \frac{1}{n-1} Y Y^T \quad (15)$$

$$= \frac{1}{n-1} (U^* X)(U^* X)^T \quad (16)$$

$$= \frac{1}{n-1} U^* (X X^T) U \quad (17)$$

$$= \frac{1}{n-1} U^* U \Sigma^2 U U^* \quad (18)$$

$$= \frac{1}{n-1} \Sigma^2 \quad (19)$$

This makes explicit the connection between the SVD and the eigenvalue method, namely that $\Sigma^2 = \Lambda[1]$.

3 Algorithm Implementation and Development

3.1 Outlines

1. Load Footage
2. Standardize footage
3. Track Paint Can
4. Standardize data and Put into Matrix
5. Perform SVD
6. View Data

3.2 Algorithms

Algorithm 1: Analyze footage using SVD

```

Import Footage
Standardize pixel dimensions and frame count
Set Window Width
for  $i = 1 : 3$  do
    Track Paint Can for Camera  $i$ 
    Standardize pixel locations
end for
Store Pixel location vectors in a Matrix
Perform SVD on Matrix
Plot Sigma Values
Plot appropriate modes

```

Algorithm 2: Track Paint Can

```

for  $i = 1 : \text{Number of Frames}$  do
    Convert frame  $i$  to grayscale
    Find the pixel location of the brightest pixel
    Store each  $x$  and  $y$  value in a respective vector
end for

```

4 Computational Results

4.1 Test 1 (Ideal Case)

In this test each of the three clips were trimmed to 200 x 200 pixel square shots in order to try and avoid unnecessary background information interfering with the data. In order to make the clips uniform, each clip was also trimmed to the length of the shortest clip (226 frames). The flashlight on top of the paint can was used as a tracking point as the can was bounced up and down. The graph below shows the movement of the paint can in both the x and y axes.

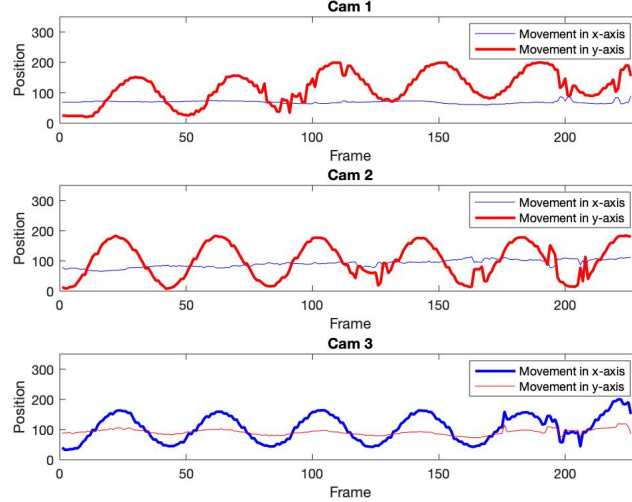


Figure 1: Initial Tracking Plots for Test 1

In this graph we can see the periodic motion of the paint can. For the first two camera angles, this motion is on the y axis. However, for the third camera angle, the can is moving on the x-axis. These camera orientations will be consistent for all four tests. The x and y locations at each frame were stored in vectors. The vectors were then combined to form a 6×226 matrix. I then performed the singular value decomposition on the matrix.

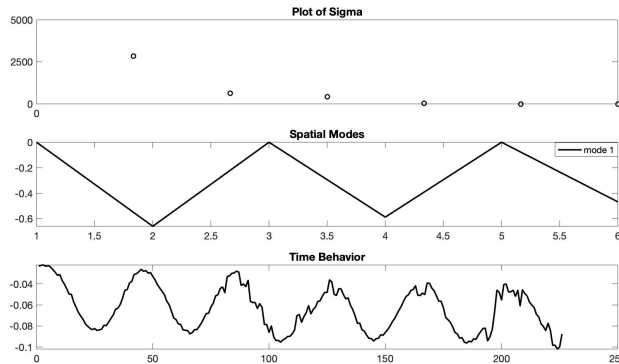


Figure 2: Singular Value Decomposition for Test 1

The first plot in the figure above is all of the sigma values for the svd. The relative size of σ_1 suggests that this is a rank 1 set of data. However, σ_2 and σ_3 are high enough that they are not rounded

off to zero. This is understandable since we could expect some minor correlated movement in the x and z axes in all three shots. Since we know that this is a rank one test, we are going to ignore movement in the other modes. The second plot in figure 2 shows the graph of the values of the first column of U and the third figure is the graph of the first column of V .

4.2 Test 2 (Noisy Case)

This test is very similar to the first. However, there is added noise created by shaking the camera for the duration of the test. Each of the three clips were trimmed to 250 x 250 pixel square shots. These squares needed to be bigger than the first experiment to compensate for the shaking of the cameras. Each clip was also trimmed to the length of the shortest clip (314 frames). Just like the first test, the flashlight was tracked and the graph below shows the movement of the paint can in both the x and y axes.

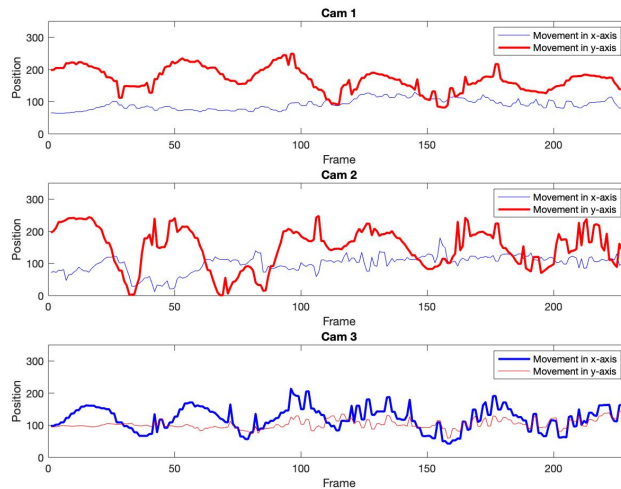


Figure 3: Initial Tracking Plots for Test 2

In this graph we can still partially make out the periodic motion of the paint can. However, the noise has now made it harder to see the movement as clearly as we did in figure 1. Similar to test one we create a matrix of all of the x and y coordinates. This time I made a 6×314 matrix as there are 314 frames to account for. I then performed the singular value decomposition on the matrix.

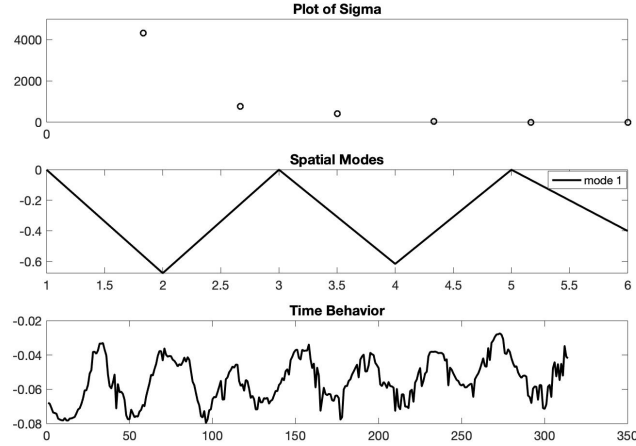


Figure 4: Singular Value Decomposition for Test 2

Once again we see that the relative size of σ_1 suggests that this is a rank 1 set of data. As we expect, the second and third plots look very similar to the ones in figure 4.

4.3 Test 3 (Horizontal Displacement)

In this test there is movement added in the x axis by swinging the object as it is bouncing. Each clip was trimmed to 200 x 200 squares 237 frames long. Just like the first test, the flashlight was tracked and the graph below shows the movement of the paint can in both the x and y axes.

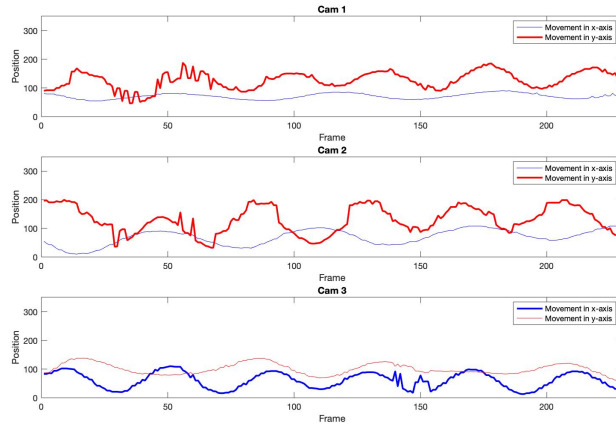


Figure 5: Initial Tracking Plots for Test 3

In this graph we can make out the periodic motion of the paint can bouncing. Additionally, we can see movement in a second axis as well. We create a matrix of all of the x and y coordinates. This time I made a 6×314 matrix as there are 314 frames to account for. I then performed the singular value decomposition on the matrix.

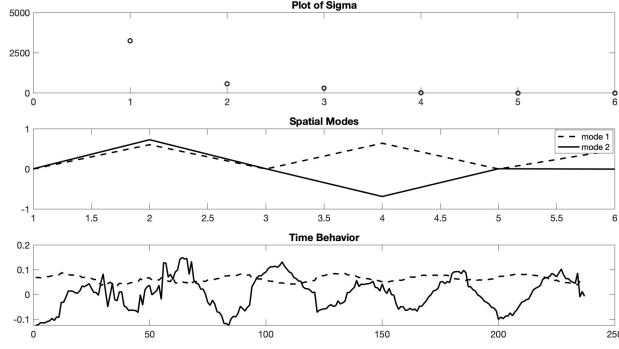


Figure 6: Singular Value Decomposition for Test 3

Due to the size of σ_1 and σ_2 , we can conclude that this is a rank 2 test. We graph the the two modes.

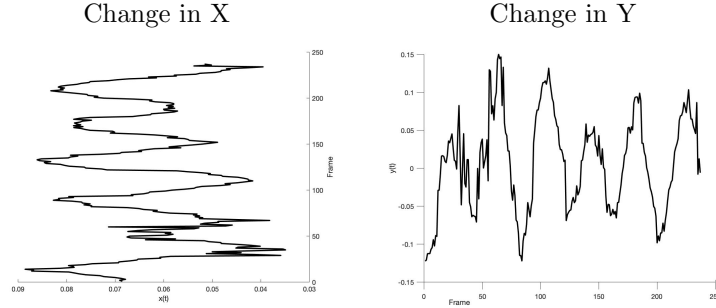


Table 1: Graph of Test 3 change in time

In the figure above we can see the way that the paint can changes on the x axis with respect to time on the right and the y axis on the right.

4.4 Test 4

In this test there is movement added in the z axis by rotating the object as it is bouncing and swinging side-to-side. Each clips was trimmed to 200 x 200 squares 392 frames long. Just like the tests before it, the flashlight was tracked and the graph below shows the movement of the paint can in both the x and y axes. However, there are times in the first clip that the flashlight is turned away from the camera. In order to compensate for that, the pink of the flashlight was turned bright white and tracked as well.

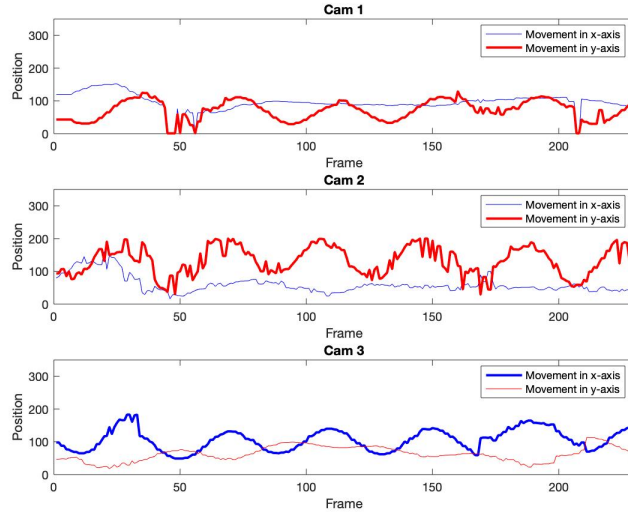


Figure 7: Initial Tracking Plots for Test 4

In this graph we can make out the periodic motion of the paint can bouncing. However, the it has become harder to identify the movement in the other axis. We create a matrix of all of the x and y coordinates. This time I made a 6×392 matrix as there are 392 frames to account for. I then performed the singular value decomposition on the matrix.

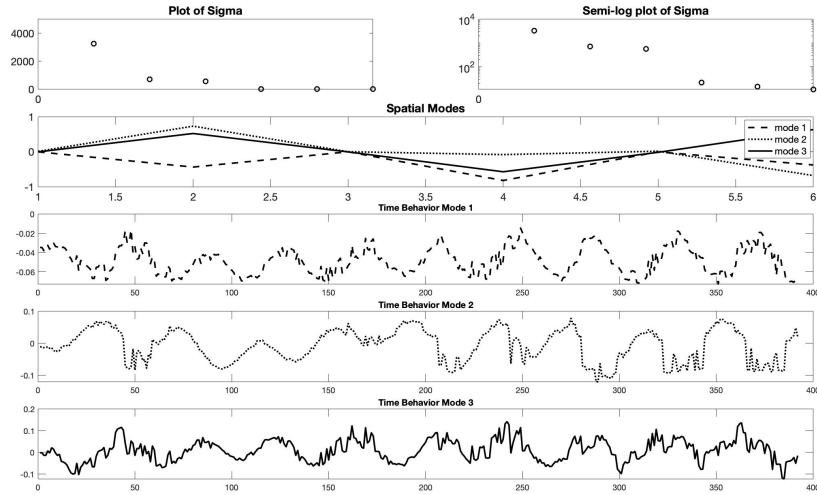


Figure 8: Singular Value Decomposition for Test 4

Due to the size of σ_1 , σ_2 , and σ_3 we can conclude that this is a rank 3 test. We can see the graph the the behavior of the three modes with respect to time in the three figures at the bottom of the figure 8. This follows with our knowledge of the test. Since the can was bouncing, swinging, and spinning, we expected to see three dimensions all engaged.

5 Summary and Conclusions

Each test, although slightly different, essentially included the same basics steps. In each case we resized the images and frame count. We then tracked the can's movement through the frames. After creating a matrix of all of the recorded x and y positions, we performed the svd on the matrix. The plot of the sigma values clued us in to the rank of the test. We were then able to graph the appropriate number of modes in order to see the movement along each orthogonal dimension that the svd discovered.

References

- [1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.
- [2] *MathWorks Website*. URL: <https://www.mathworks.com/help/matlab/index.html>.

Appendix A MATLAB Functions

- **sz = size(A)**: returns a row vector whose elements are the lengths of the corresponding dimensions of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4]. [2]
- **I = rgb2gray(RGB)**: converts the truecolor image RGB to the grayscale image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. [2]
- **[M,I] = max()**: returns the index into the operating dimension that corresponds to the maximum value of A for any of the previous syntaxes. [2]
- **[row,col] = ind2sub(sz,ind)**: returns the arrays row and col containing the equivalent row and column subscripts corresponding to the linear indices ind for a matrix of size sz. Here sz is a vector with two elements, where sz(1) specifies the number of rows and sz(2) specifies the number of columns.[2]
- **A(:)**:reshapes all elements of A into a single column vector. This has no effect if A is already a column vector.
- **size(A)**: returns a row vector whose elements are the lengths of the corresponding dimensions of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4].[2]
- **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 [2].
- **sigma = svd(A)**: returns a vector sigma containing the singular values of a symbolic matrix A. [2]
- **y = linspace(x1,x2,n)**: generates n points. The spacing between the points is $\frac{x2-x1}{n-1}$ [2]
- **D = diag(v)** : returns a square diagonal matrix with the elements of vector v on the main diagonal. [2].

Appendix B MATLAB Code

This code can be found at: <https://github.com/ReedNomura/AMATH-582/blob/master/Homework4.m>

```
1 %AMATH 582 Homework 3
2
3 %% Ideal Case
4 clear all; close all; clc; %Start Fresh
5 load('cam1.1.mat');
6 load('cam2.1.mat');
```



```

7 load('cam3_1.mat');
8
9 %% Initial Dimension Check
10 [A1_1,B1_1,C1_1,D1_1] = size(vidFrames1_1);
11 [A2_1,B2_1,C2_1,D2_1] = size(vidFrames2_1);
12 [A3_1,B3_1,C3_1,D3_1] = size(vidFrames3_1);
13 %% Resizing videos to 200 by 200 and 226 Frames
14 vidFrames1_1 = vidFrames1_1(201:400,251:450,:,:);
15 vidFrames2_1 = vidFrames2_1(101:300,201:400,:,18:243); % This video starts approximately ...
    18 frames earlier than Camera 1 and ends later as well
16 vidFrames3_1 = vidFrames3_1(176:375,251:450,:,7:232); % This video starts approximately 7
17 [A1_1,B1_1,C1_1,D1_1] = size(vidFrames1_1);
18 [A2_1,B2_1,C2_1,D2_1] = size(vidFrames2_1);
19 [A3_1,B3_1,C3_1,D3_1] = size(vidFrames3_1);
20 X1_1 = [];
21 Y1_1 = [];
22 X2_1 = [];
23 Y2_1 = [];
24 X3_1 = [];
25 Y3_1 = [];
26 %% Camera 1
27 for i = 1:D1_1
28     img = rgb2gray(vidFrames1_1(:,:,i)); %Make every frame Black and White
29     [Max, Ind] = max(img(:)); %Reshapes img to vector and find the max value of img and ...
        where it is indexed
30     [y1_1, x1_1] = ind2sub(size(img),Ind);
31     X1_1 = [X1_1, x1_1];
32     Y1_1 = [Y1_1, y1_1];
33 end
34 %% Camera 2
35 for i = 1:D2_1
36     img = rgb2gray(vidFrames2_1(:,:,i));
37     [Max, Ind] = max(img(:));
38     [y2_1, x2_1] = ind2sub(size(img),Ind);
39     X2_1 = [X2_1, x2_1];
40     Y2_1 = [Y2_1, y2_1];
41 end
42 %% Camera 3
43 for i = 1:D3_1
44     img = rgb2gray(vidFrames3_1(:,:,i));
45     [Max, Ind] = max(img(:));
46     [y3_1, x3_1] = ind2sub(size(img),Ind);
47     X3_1 = [X3_1, x3_1];
48     Y3_1 = [Y3_1, y3_1];
49 end
50 %% Plots for Ideal Case
51 figure()
52 subplot(3,1,1)
53 plot(X1_1,'b')
54 hold on;
55 plot(Y1_1, 'r', 'Linewidth',[2])
56 hold off;
57 xlabel('Frame')
58 xlim([0,230])
59 ylabel('Position')
60 ylim([0,350])
61 legend('Movement in x-axis','Movement in y-axis')
62 title('Cam 1')
63
64 subplot(3,1,2)
65 plot(X2_1,'b')
66 hold on;
67 plot(Y2_1, 'r', 'Linewidth',[2])
68 hold off;
69 xlabel('Frame')
70 xlim([0,230])
71 ylabel('Position')
72 ylim([0,350])

```

```

73 legend('Movement in x-axis','Movement in y-axis')
74 title('Cam 2')
75
76 subplot(3,1,3)
77 plot(X3_1,'b','LineWidth',[2])
78 hold on;
79 plot(Y3_1,'r')
80 hold off;
81 xlabel('Frame')
82 xlim([0,230])
83 ylabel('Position')
84 ylim([0,350])
85 legend('Movement in x-axis','Movement in y-axis')
86 title('Cam 3')
87
88 %% Standardize
89 X1_1 = zscore(X1_1);
90 X2_1 = zscore(X2_1);
91 X3_1 = zscore(X3_1);
92 Y1_1 = zscore(Y1_1);
93 Y2_1 = zscore(Y2_1);
94 Y3_1 = zscore(Y3_1);
95
96 %% SVD
97 X1 = [X1_1;Y1_1;X2_1;Y2_1;X3_1;Y3_1];
98 x = linspace(1,6,6);
99 t = linspace(1,226,226);
100 [m,n]=size(X1); % compute data size
101 mn = mean(X1,2); % compute mean for each row
102 X1 = X1-repmat(mn,1,n); % subtract mean
103 [u,s,v]=svd(X1/sqrt(n-1)); % perform the SVD
104 sig = diag(s).^2; % produce diagonal variances
105 Yu = u*X1'; % produce the principal components projection
106 Yv = X1'*v;
107
108 %%
109 figure()
110 sig=diag(s);
111
112 subplot(3,1,1), plot(sig,'ko','LineWidth',[1.5])
113 axis([0 6 0 2000])
114 set(gca,'FontSize',[13],'Ytick',[0 1000 2000],'Xtick',[0 1 2 3 4 5 6])
115 text(20,40,'(a)','FontSize',[13])
116 title('Plot of Sigma')
117
118 % subplot(3,2,2), semilogy(sig,'ko','LineWidth',[1.5])
119 % axis([0 6 0 10000])
120 % set(gca,'FontSize',[13],'Ytick',[0, 10^0,10^2, 10^4],'Xtick',[0 10 15 20 25]);
121 % text(20,40,'(b)','FontSize',[13])
122 % title('Semi-log plot of Sigma')
123
124
125 subplot(3,1,2) % spatial modes
126 plot(x,u(:,1),'k','LineWidth',[2])
127 set(gca,'FontSize',[13])
128 legend('mode 1','Location','NorthWest')
129 title('Spatial Modes')
130
131 subplot(3,1,3) % time behavior
132 plot(t,v(:,1),'k','LineWidth',[2])
133 set(gca,'FontSize',[13])
134 title('Time Behavior')
135 %% Movie
136 figure()
137 for k = 1:D1_1
138     ColorScale = (D1_1 - k)/D1_1;
139     CSM = [ColorScale, ColorScale, ColorScale];
140     Frame = k;

```

```

141     plot(0, v(k,1), '.', 'MarkerSize', [10], 'color', [0 0 0])
142     axis([-0.005 .005 .025 .125])
143     set(gca, 'FontSize', [13], 'Ytick', [], 'Xtick', [])
144     xlabel('Axis 1')
145     ylabel('Axis 2')
146     title('Movement in Relevant Axes')
147
148     text (0.002, .07, 'Frame:' )
149     text (0.003, .07, num2str(k))
150     pause(.05)
151
152 end
153
154
155
156 %% Noisy case
157 clear all; close all; clc; %Start Fresh
158
159 load('cam1.2.mat');
160 load('cam2.2.mat');
161 load('cam3.2.mat');
162
163 %% Initial Dimension Check
164 [A1_2,B1_2,C1_2,D1_2] = size(vidFrames1_2);
165 [A2_2,B2_2,C2_2,D2_2] = size(vidFrames2_2);
166 [A3_2,B3_2,C3_2,D3_2] = size(vidFrames3_2);
167 %% Resizing videos to 250 by 250 and 314 Frames
168 vidFrames1_2 = vidFrames1_2(146:395,251:500,:,:);
169 vidFrames2_2 = vidFrames2_2(76:325,201:450,:,28:341);
170 vidFrames3_2 = vidFrames3_2(151:400,251:500,:,1:314);
171 [A1_2,B1_2,C1_2,D1_2] = size(vidFrames1_2);
172 [A2_2,B2_2,C2_2,D2_2] = size(vidFrames2_2);
173 [A3_2,B3_2,C3_2,D3_2] = size(vidFrames3_2);
174 X1_2 = [];
175 Y1_2 = [];
176 X2_2 = [];
177 Y2_2 = [];
178 X3_2 = [];
179 Y3_2 = [];
180 %% Camera 1
181 for i = 1:D1_2
182     img = rgb2gray(vidFrames1_2(:,:,i)); %Make every frame Black and White
183     [Max, Ind] = max(img(:)); %Reshapes img to vector and findd the max value of img and ...
184         where it is indexed
185     [y1_2, x1_2] = ind2sub(size(img),Ind);
186     X1_2 = [X1_2, x1_2];
187     Y1_2 = [Y1_2, y1_2];
188 end
189 %% Camera 2
190 for i = 1:D2_2
191     img = rgb2gray(vidFrames2_2(:,:,i));
192     [Max, Ind] = max(img(:));
193     [y2_2, x2_2] = ind2sub(size(img),Ind);
194     X2_2 = [X2_2, x2_2];
195     Y2_2 = [Y2_2, y2_2];
196 end
197 %% Camera 3
198 for i = 1:D3_2
199     img = rgb2gray(vidFrames3_2(:,:,i));
200     [Max, Ind] = max(img(:));
201     [y3_2, x3_2] = ind2sub(size(img),Ind);
202     X3_2 = [X3_2, x3_2];
203     Y3_2 = [Y3_2, y3_2];
204 end
205 %% Plots for Noisy Case
206 figure()
207 subplot(3,1,1)
208 plot(X1_2, 'b')

```

```

208 hold on;
209 plot(Y1_2, 'r', 'Linewidth',[2])
210 hold off;
211 xlabel('Frame')
212 xlim([0,230])
213 ylabel('Position')
214 ylim([0,350])
215 legend('Movement in x-axis','Movement in y-axis')
216 title('Cam 1')
217
218 subplot(3,1,2)
219 plot(X2_2,'b')
220 hold on;
221 plot(Y2_2, 'r', 'Linewidth',[2])
222 hold off;
223 xlabel('Frame')
224 xlim([0,230])
225 ylabel('Position')
226 ylim([0,350])
227 legend('Movement in x-axis','Movement in y-axis')
228 title('Cam 2')
229
230 subplot(3,1,3)
231 plot(X3_2,'b', 'Linewidth',[2])
232 hold on;
233 plot(Y3_2, 'r')
234 hold off;
235 xlabel('Frame')
236 xlim([0,230])
237 ylabel('Position')
238 ylim([0,350])
239 legend('Movement in x-axis','Movement in y-axis')
240 title('Cam 3')
241
242 %% Standardize
243 X1_2 = zscore(X1_2);
244 X2_2 = zscore(X2_2);
245 X3_2 = zscore(X3_2);
246 Y1_2 = zscore(Y1_2);
247 Y2_2 = zscore(Y2_2);
248 Y3_2 = zscore(Y3_2);
249
250 %% SVD
251 X2 = [X1_2;Y1_2;X2_2;Y2_2;X3_2;Y3_2];
252 x = linspace(1,6,6);
253 t = linspace(1,314,314);
254 [m,n]=size(X2); % compute data size
255 mn = mean(X2,2); % compute mean for each row
256 X2 = X2-repmat(mn,1,n); % subtract mean
257 [u,s,v]=svd(X2'/sqrt(n-1)); % perform the SVD
258 sig = diag(s).^2; % produce diagonal variances
259 Yu = u*X2'; % produce the principal components projection
260 Yv = X2'*v;
261
262
263 %%
264 figure()
265 sig=diag(s);
266
267 subplot(3,1,1), plot(sig,'ko','Linewidth',[1.5])
268 axis([0 6 0 5000])
269 set(gca,'FontSize',[13],'Xtick',[0 10 15 20 25])
270 title('Plot of Sigma')
271
272 subplot(3,1,2) % spatial modes
273 plot(x,u(:,1),'k','Linewidth',[2])
274 set(gca,'FontSize',[13])
275 legend('mode 1','Location','NorthWest')

```

```

276 title('Spatial Modes')
277
278 subplot(3,1,3) % time behavior
279 plot(t,v(:,1),'k','Linewidth',[2])
280 set(gca,'FontSize',[13])
281 title('Time Behavior')
282 %% Movie
283 figure()
284 for k = 1:D1_2
285     ColorScale = (D1_2 - k)/D1_2;
286     CSM = [ColorScale, ColorScale, ColorScale];
287     Frame = k;
288     plot(0, v(k,1),'.','MarkerSize',[10], 'color',[0 0 0])
289     axis([-0.05 0.05 -0.1 0])
290     set(gca,'FontSize',[13], 'Ytick',[], 'Xtick',[])
291     xlabel('Axis 1')
292     ylabel('Axis 2')
293     title('Movement in Relevant Axes')
294     text (0.017, -0.025, 'Frame:' )
295     text (0.025, -0.025, num2str(k))
296     pause(.05)
297
298 end
299
300
301
302 %% Horizontal Displacement
303 clear all; close all; clc; %Start Fresh
304
305 load('cam1.3.mat');
306 load('cam2.3.mat');
307 load('cam3.3.mat');
308
309 %% Initial Dimension Check
310 [A1_3,B1_3,C1_3,D1_3] = size(vidFrames1_3);
311 [A2_3,B2_3,C2_3,D2_3] = size(vidFrames2_3);
312 [A3_3,B3_3,C3_3,D3_3] = size(vidFrames3_3);
313 %% Resizing videos to 200 by 200 and 237 Frames
314 vidFrames1_3 = vidFrames1_3(201:400,241:440, :, 3:239);
315 vidFrames2_3 = vidFrames2_3(161:360,226:425, :, 39:275); % This video starts approximately ...
    18 frames earlier than Camera 1 and ends later as well
316 vidFrames3_3 = vidFrames3_3(151:350,276:475, :, :); % This video starts approximately 7
317 [A1_3,B1_3,C1_3,D1_3] = size(vidFrames1_3);
318 [A2_3,B2_3,C2_3,D2_3] = size(vidFrames2_3);
319 [A3_3,B3_3,C3_3,D3_3] = size(vidFrames3_3);
320 X1_3 = [];
321 Y1_3 = [];
322 X2_3 = [];
323 Y2_3 = [];
324 X3_3 = [];
325 Y3_3 = [];
326 %% Camera 1
327 for i = 1:D1_3
328     img = rgb2gray(vidFrames1_3(:, :, :, i)); %Make every frame Black and White
329     [Max, Ind] = max(img(:)); %Reshapes img to vector and findd the max value of img and ...
        where it is indexed
330     [y1_3, x1_3] = ind2sub(size(img),Ind);
331     X1_3 = [X1_3, x1_3];
332     Y1_3 = [Y1_3, y1_3];
333 end
334 %% Cam 2 Pink to White
335 for j = 1:D2_3
336     rgbImage = vidFrames2_3(:, :, :, j);
337     %subplot(2, 1, 1);
338     %imshow(rgbImage);
339     %Extract the individual red, green, and blue color channels.
340     redChannel = rgbImage(:, :, 1);
341     greenChannel = rgbImage(:, :, 2);

```

```

342 blueChannel = rgbImage(:, :, 3);
343 %Get the yellow mask
344 pinkMask = 240 < redChannel & 170 < greenChannel & 200 < blueChannel;
345 % Make Pink mask White
346 redChannel(pinkMask) = 255;
347 greenChannel(pinkMask) = 255;
348 blueChannel(pinkMask) = 255;
349 % Recombine separate color channels into a single, true color RGB image.
350 rgbImage2 = cat(3, redChannel, greenChannel, blueChannel);
351 %subplot(2, 1, 2);
352 %imshow(rgbImage2);
353 vidFrames2_3(:, :, :, j) = rgbImage2;
354 end
355
356 %% Camera 2
357 for i = 1:D2_3
358     img = rgb2gray(vidFrames2_3(:, :, :, i));
359     [Max, Ind] = max(img(:));
360     [y2_3, x2_3] = ind2sub(size(img), Ind);
361     X2_3 = [X2_3, x2_3];
362     Y2_3 = [Y2_3, y2_3];
363 end
364
365 %% Camera 3
366 for i = 1:D3_3
367     img = rgb2gray(vidFrames3_3(:, :, :, i));
368     [Max, Ind] = max(img(:));
369     [y3_3, x3_3] = ind2sub(size(img), Ind);
370     X3_3 = [X3_3, x3_3];
371     Y3_3 = [Y3_3, y3_3];
372 end
373 %% Plots for Horizontal Displacement
374 figure()
375 subplot(3,1,1)
376 plot(X1_3, 'b')
377 hold on;
378 plot(Y1_3, 'r', 'Linewidth', [2])
379 hold off;
380 xlabel('Frame')
381 xlim([0,250])
382 ylabel('Position')
383 ylim([0,350])
384 legend('Movement in x-axis', 'Movement in y-axis')
385 title('Cam 1')
386
387 subplot(3,1,2)
388 plot(X2_3, 'b')
389 hold on;
390 plot(Y2_3, 'r', 'Linewidth', [2])
391 hold off;
392 xlabel('Frame')
393 xlim([0,250])
394 ylabel('Position')
395 ylim([0,350])
396 legend('Movement in x-axis', 'Movement in y-axis')
397 title('Cam 2')
398
399 subplot(3,1,3)
400 plot(X3_3, 'b', 'Linewidth', [2])
401 hold on;
402 plot(Y3_3, 'r')
403 hold off;
404 xlabel('Frame')
405 xlim([0,250])
406 ylabel('Position')
407 ylim([0,350])
408 legend('Movement in x-axis', 'Movement in y-axis')
409 title('Cam 3')

```

```

410 %% Standardize
411 X1_3 = X1_3/mean(X1_3);
412 X2_3 = X2_3/mean(X2_3);
413 X3_3 = X3_3/mean(X3_3);
414 Y1_3 = Y1_3/mean(Y1_3);
415 Y2_3 = Y2_3/mean(Y2_3);
416 Y3_3 = Y3_3/mean(Y3_3);
417 %% SVD
418 X3 = [X1_3;Y1_3;X2_3;Y2_3;X3_3;Y3_3];
419 x = linspace(1,6,6);
420 t = linspace(1,237,237);
421 [m,n]=size(X3); % compute data size
422 mn = mean(X3,2); % compute mean for each row
423 X3 = X3-repmat(mn,1,n); % subtract mean
424 [u,s,v]=svd(X3'/sqrt(n-1)); % perform the SVD
425 sig = diag(s).^2; % produce diagonal variances
426 Yu = u*X3'; % produce the principal components projection
427 Yv = X3'*v;
428
429
430 %%
431 figure()
432
433 subplot(3,1,1), plot(sig,'ko','Linewidth',[1.5])
434 axis([0 6 0 5000])
435 set(gca,'FontSize',[13],'Ytick',[0 2500 5000],'Xtick',[0 1 2 3 4 5 6]);
436 title('Plot of Sigma')
437
438 subplot(3,1,2) % spatial modes
439 plot(x,v(:,1),'k:', x, v(:,2),'k', 'Linewidth',[2])
440 set(gca,'FontSize',[13])
441 legend('mode 1','mode 2','Location','NorthWest')
442 title('Spatial Modes')
443
444 subplot(3,1,3) % time behavior
445 plot(t,u(:,1),'k:',t,u(:,2),'k','Linewidth',[2])
446 set(gca,'FontSize',[13])
447 title('Time Behavior')
448
449 % subplot(4,1,4) % time behavior
450 % plot(t,u(:,3),'k','Linewidth',[2])
451 % set(gca,'FontSize',[13])
452 % title('Time Behavior')
453
454 %% 3d Plot of the change in x and y with respect to time
455 figure()
456 plot3(t, v(:,1),v(:,2),'k','Linewidth',[2])
457 xlabel('Frame')
458 ylabel('x(t)')
459 zlabel('y(t)')
460 grid on
461 %% Movie
462 figure()
463 for k = 1:D1_3
464     plot(v(k,1), v(k,2),'.', 'MarkerSize', [10])
465     axis([-0.5 0.5 -0.5 0.5])
466     pause(.05)
467 end
468 %% Movie Vid 3
469 figure()
470 for k = 1:D1_3
471     ColorScale = (D1_3 - k)/D1_3;
472     CSM = [ColorScale, 1-ColorScale, (1+ColorScale)/2];
473     Frame = k;
474     plot3(k, v(k,1),v(k,2),'.', 'MarkerSize', [10], 'color', [0 0 0])
475     axis([1 237 -0.085 -0.15 .15])
476     %set(gca,'FontSize',[13],'Ytick',[],'Xtick',[])
477     xlabel('Axis 1')

```

```

478     ylabel('Axis 2')
479     title('Movement in Relevant Axes')
480     grid on
481     hold on
482     %text (0.1, .1, 'Frame:' )
483     %text (0.112,.1, num2str(k))
484
485     pause(.05)
486
487 end
488
489
490
491
492
493 %% Horizontal Displacement and Rotation:
494 clear all; close all; clc; %Start Fresh
495
496 load('cam1.4.mat');
497 load('cam2.4.mat');
498 load('cam3.4.mat');
499
500 %% Initial Dimension Check
501 [A1_4,B1_4,C1_4,D1_4] = size(vidFrames1_4);
502 [A2_4,B2_4,C2_4,D2_4] = size(vidFrames2_4);
503 [A3_4,B3_4,C3_4,D3_4] = size(vidFrames3_4);
504 %% Resizing videos to 200 by 200 and 392 Frames
505 vidFrames1_4 = vidFrames1_4(226:425,276:475,:,:);
506 vidFrames2_4 = vidFrames2_4(101:300,226:425,:,14:405);
507 vidFrames3_4 = vidFrames3_4(151:350,276:475,:,3:394);
508 [A1_4,B1_4,C1_4,D1_4] = size(vidFrames1_4);
509 [A2_4,B2_4,C2_4,D2_4] = size(vidFrames2_4);
510 [A3_4,B3_4,C3_4,D3_4] = size(vidFrames3_4);
511 X1_4 = [];
512 Y1_4 = [];
513 X2_4 = [];
514 Y2_4 = [];
515 X3_4 = [];
516 Y3_4 = [];
517 %% Cam 1 Remove non pink
518 for j = 1:D1_4
519     rgbImage = vidFrames1_4(:, :, j);
520     %Extract the individual red, green, and blue color channels.
521     redChannel = rgbImage(:, :, 1);
522     greenChannel = rgbImage(:, :, 2);
523     blueChannel = rgbImage(:, :, 3);
524     %Pink Mask
525     pinkMask = 245 < redChannel & 125 < greenChannel & greenChannel < 195 & 140 < blueChannel & ...
        blueChannel < 210;
526     %NonpinkMask
527     nonpinkMask = 240 > redChannel | 130 > greenChannel | greenChannel > 210 | 155 > ...
        blueChannel | blueChannel > 210;
528
529 % Remove Unwanted colors
530 redChannel (nonpinkMask) = 0;
531 greenChannel (nonpinkMask) = 0;
532 blueChannel (nonpinkMask) = 0;
533
534 % Recombine separate color channels into a single, true color RGB image.
535 rgbImage2 = cat(3, redChannel, greenChannel, blueChannel);
536 vidFrames1_4(:, :, :, j) = rgbImage2;
537 end
538
539
540 %% Camera 1
541 for i = 1:D1_4
542     img = rgb2gray(vidFrames1_4(:, :, :, i)); %Make every frame Black and White
543     [Max, Ind] = max(img(:)); %Reshapes img to vector and findd the max value of img and ...

```



```

        where it is indexed
544     [y1_4, x1_4] = ind2sub(size(img),Ind);
545     X1_4 = [X1_4, x1_4];
546     Y1_4 = [Y1_4, y1_4];
547 end
548
549 %% Camera 2
550 for i = 1:D2_4
551     img = rgb2gray(vidFrames2_4(:,:,i));
552     [Max, Ind] = max(img(:));
553     [y2_4, x2_4] = ind2sub(size(img),Ind);
554     X2_4 = [X2_4, x2_4];
555     Y2_4 = [Y2_4, y2_4];
556 end
557
558 %% Camera 3
559 for i = 1:D3_4
560     img = rgb2gray(vidFrames3_4(:,:,i));
561     [Max, Ind] = max(img(:));
562     [y3_4, x3_4] = ind2sub(size(img),Ind);
563     X3_4 = [X3_4, x3_4];
564     Y3_4 = [Y3_4, y3_4];
565 end
566 %% Plots for Horizontal Displacement and Rotation
567 figure()
568 subplot(3,1,1)
569 plot(X1_4, 'b')
570 hold on;
571 plot(Y1_4, 'r', 'Linewidth',[2])
572 hold off;
573 xlabel('Frame')
574 xlim([0,230])
575 ylabel('Position')
576 ylim([0,350])
577 legend('Movement in x-axis','Movement in y-axis')
578 title('Cam 1')
579
580 subplot(3,1,2)
581 plot(X2_4, 'b')
582 hold on;
583 plot(Y2_4, 'r', 'Linewidth',[2])
584 hold off;
585 xlabel('Frame')
586 xlim([0,230])
587 ylabel('Position')
588 ylim([0,350])
589 legend('Movement in x-axis','Movement in y-axis')
590 title('Cam 2')
591
592 subplot(3,1,3)
593 plot(X3_4, 'b', 'Linewidth',[2])
594 hold on;
595 plot(Y3_4, 'r')
596 hold off;
597 xlabel('Frame')
598 xlim([0,230])
599 ylabel('Position')
600 ylim([0,350])
601 legend('Movement in x-axis','Movement in y-axis')
602 title('Cam 3')
603
604
605 %% Standardize
606 m1X=mean(X1_4);
607 m2X=mean(X2_4);
608 m3X=mean(X3_4);
609 m1Y=mean(Y1_4);
610 m2Y=mean(Y2_4);

```

```

611 m3Y=mean(Y3_4);
612 X1_4 = X1_4/m1X;
613 X2_4 = X2_4/m2X;
614 X3_4 = X3_4/m3X;
615 Y1_4 = Y1_4/m1Y;
616 Y2_4 = Y2_4/m2Y;
617 Y3_4 = Y3_4/m3Y;
618 %% Standardize
619 X1_4 = zscore(X1_4);
620 X2_4 = zscore(X2_4);
621 X3_4 = zscore(X3_4);
622 Y1_4 = zscore(Y1_4);
623 Y2_4 = zscore(Y2_4);
624 Y3_4 = zscore(Y3_4);
625 %% SVD
626 X4 = [X1_4;Y1_4;X2_4;Y2_4;X3_4;Y3_4];
627 x = linspace(1,6,6);
628 t = linspace(1,392,392);
629 [m,n]=size(X4); % compute data size
630 mn = mean(X4,2); % compute mean for each row
631 X4 = X4-repmat(mn,1,n); % subtract mean
632 [u,s,v]=svd(X4'/sqrt(n-1)); % perform the SVD
633 sig = diag(s).^2; % produce diagonal variances
634 Yu = u*X4'; % produce the principal components projection
635 Yv = X4'*v;
636 %%
637 figure()
638 subplot(3,1,1)
639 plot(t, Yv(:,1),'k')
640 subplot(3,1,2)
641 plot(t, Yv(:,2),'k')
642 subplot(3,1,3)
643 plot(t, Yv(:,3),'k')
644 %%
645 figure()
646
647 subplot(5,1,1), plot(sig,'ko','Linewidth',[1.5])
648 axis([0 6 0 5000])
649 % set(gca,'FontSize',[13],'Xtick',[0 10 15 20 25]);
650 title('Plot of Sigma')
651
652 % subplot(5,2,2), semilogy(sig,'ko','Linewidth',[1.5])
653 % %axis([0 6 0 100])
654 % %set(gca,'FontSize',[13],'Ytick',[0, 10^1,10^2]);
655 % title('Semi-log plot of Sigma')
656
657
658 subplot(5,1,2) % spatial modes
659 plot(x,v(:,1),'k—', x,v(:,2),'k:', x, v(:,3), 'k', 'Linewidth',[2])
660 set(gca,'FontSize',[13])
661 legend('mode 1','mode 2', 'mode 3', 'Location','NorthWest')
662 title('Spatial Modes')
663
664
665 subplot(5,1,3) % time behavior
666 plot(t,u(:,1),'k—','Linewidth',[2])
667 title('Time Behavior Mode 1')
668 subplot(5,1,4) % time behavior
669 plot(t,u(:,2),'k:', 'Linewidth',[2])
670 title('Time Behavior Mode 2')
671 subplot(5,1,5) % time behavior
672 plot(t,u(:,3),'k', 'Linewidth',[2])
673 title('Time Behavior Mode 3')
674
675 %%
676 figure()
677 for k = 1:D1_4
678     plot(u(k,1),u(k,3),'.', 'MarkerSize', [10], 'color', [0 0 0])

```

```

679     %axis([-0.1 0 -0.15 0.1 -0.1 0.15])
680     set(gca,'FontSize',[13],'Ytick',[])
681     xlabel('Y - Axis')
682     ylabel('Z - Axis')
683     %zlabel('Axis 3')
684     title('Movement in Relevant Axes')
685     grid on
686     hold on
687     %text (0.06, .1,.1, 'Frame:' )
688     %text (0.06,.1,.1, num2str(k))
689     %pause(.03)
690
691 end
692
693 %% Movie Vid 4
694 figure()
695 for k = 1:D1.4
696     ColorScale = (D1.4 - k)/D1.4;
697     CSM = [ColorScale, 1-ColorScale , (1+ColorScale)/2];
698     Frame = k;
699     plot3(u(k,1),u(k,2),u(k,3),'.', 'MarkerSize', [10], 'color', CSM)
700     axis([-0.1 0 -0.15 0.1 -0.1 0.15])
701     %set (gca,'FontSize',[13],'Ytick',[],'Xtick',[])
702     xlabel('Axis 1')
703     ylabel('Axis 2')
704     zlabel('Axis 3')
705     title('Movement in Relevant Axes')
706     grid on
707     hold on
708     %text (0.06, .1,.1, 'Frame:' )
709     %text (0.06,.1,.1, num2str(k))
710     pause(.03)
711
712 end

```