

Principal Component Analysis: Evaluating a Spring-Mass System

Christina Smith

AMATH 482 - Winter 2020

Abstract

In this exercise, we evaluate the motion of a spring-mass system under four test cases using principal component analysis. Our data sets contain video filmed from three different perspectives. We perform singular value decomposition on our data sets and use the results to transform our data to new bases. We then use principal component analysis to extract the most relevant data and evaluate the motion of the spring-mass system under given conditions.

I. Introduction

Our data sets consist of videos of an oscillating paint can suspended from a spring. We have four cases and in each case the can is filmed from three different perspectives. The first case shows near ideal harmonic motion; the can is released perpendicular to the ground with no rotation. In the second case, camera shake is introduced to the video. In the third case the can is released at a horizontal displacement from the connection point of the spring. In the fourth case, there is horizontal displacement and the can is also released with rotation. In order to process this data we must track the motion of a point on the paint can for each video clip. We then combine the data for each test into a single matrix for analysis. Using Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) we transform the data to a new basis and calculate a covariance matrix for the data in the new basis. Using this, we can see which modes contain the highest variance and use this information to create a reduced rank approximation of our data that can be plotted.

II. Theoretical Background

Singular Value Decomposition

Since we want to find out about the variation in our data, we create a new basis that most accurately fits to the data by removing redundancies and orders the

variances of different measurements. For any given matrix \mathbf{A} the singular value decomposition of \mathbf{A} is as follows:

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

where \mathbf{U} and \mathbf{V} are unitary matrices and $\mathbf{\Sigma}$ is a diagonal matrix. Since we are considering that \mathbf{A} is a matrix where each column represents coordinates measured at a point in time, then for our use, \mathbf{U} contains the principle component vectors that form a basis, $\mathbf{\Sigma}$ contains the singular values that scale the semiaxes, and \mathbf{V} is an orthonormal rotation matrix which is then scaled by $\mathbf{\Sigma}$. Note that by convention, the singular values in $\mathbf{\Sigma}$ are ordered from largest to smallest, left to right.

The more variation a data set exhibits along an axis, the better that axis is for representing the data in the system. In PCA, we use this fact to create a low-rank approximation of the data. For our exercise here we have a set of data in six dimensions and we will use PCA to determine how we can best represent the data in less dimensions.

Our first step is to find the principal components of the data by performing SVD. Once we have done that, we can project our data onto its principal component basis as follows:

$$\mathbf{Y} = \mathbf{U}^T \mathbf{A} \quad (2)$$

This is the state where the data best aligns with the basis and can be evaluated using a reduced rank approximation. In order to do this, we find the covariance matrix of this transformed data.

Covariance and Covariance Matrices

Covariance describes the dependence between data sets. A high covariance means that the data are statistically dependent, and a zero covariance means that there is no dependency (i.e., no redundancy) between data sets.

The formula to compute the covariance between two vectors \mathbf{a} and \mathbf{b} in R^n is

$$\sigma_{\mathbf{ab}}^2 = \frac{1}{n-1} \mathbf{ab}^T \quad (3)$$

In a covariance matrix, the diagonal terms represent the variance within one data set, and the off-diagonal terms represent the covariance between two data sets. Since $\sigma_{\mathbf{ab}}^2 = \sigma_{\mathbf{ba}}^2$, any covariance matrix is symmetric.

The formula for the covariance between the vectors in the \mathbf{Y} matrix is

$$\mathbf{C_Y} = \frac{1}{n-1} \mathbf{YY}^T \quad (4)$$

where n is the number of columns and $\frac{1}{n-1}$ is a normalizing factor which provides an unbiased estimator. Using Formulas (1), (2), and (4) we can derive

$$\mathbf{C}_Y = \frac{1}{n-1} \mathbf{\Sigma}^2 \quad (5)$$

This means that the SVD of our matrix \mathbf{A} give us information about how our data relates to our ideal basis. The non-zero elements of $\mathbf{\Sigma}^2$, σ^2 , represent the variance of the data around a given axis in the ideal basis. The higher the variance around an axis, the better it can be used to describe the data.

When we find the axes with the most variance in the basis we can use these to create a reduced rank approximation of the system.

III. Algorithm Implementation and Development

In order to retrieve the data about the paint can's location in each video frame we use Matlab's `ginput()` method which captures screen coordinates upon a mouse click (See Appx. A). The axes for the clicks is oriented by matching them up with the spring and the top of the can. For each test, we capture the data for each camera in a matrix and then 'crop' the matrices so that the appropriate time samples align.

We then combine these data sets into in a matrix with columns representing the data-points gathered at each time-step. The next step is to normalize the rows in the matrix by dividing each row by its mean. This will ensure that the SVD will best represent the variation of our data along a mode rather than the mean of that data.

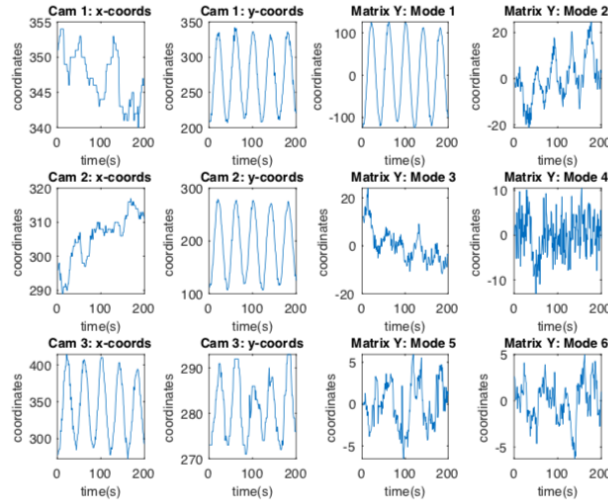


Figure 1: Test 1: Data in original basis and principal component basis

We are now ready to evaluate this data using SVD and PCA. We compute the SVD of \mathbf{A} using Matlab's built in function (See Appx. A) and then use Formula (2) to transform the data to a new basis. We then compute the covariance matrix of the data in the new basis using Formula (5). This gives us the variance around the different axes of the basis. Since the singular values in $\mathbf{\Sigma}$ are ordered from largest to smallest, the variance terms in the covariance matrix are also ordered from largest to smallest.

IV. Computational Results & Supplementary Plots

Test 1: Ideal Signal

Singular values: $\sigma_1 = 1,184$, $\sigma_2 = 134$, $\sigma_3 = 96$, $\sigma_4 = 66$, $\sigma_5 = 38$, $\sigma_6 = 30$

Test 2: Noisy Signal (camera shake)

Singular values: $\sigma_1 = 1,047$, $\sigma_2 = 566$, $\sigma_3 = 392$, $\sigma_4 = 218$, $\sigma_5 = 202$, $\sigma_6 = 145$

Test 3: Horizontal Displacement

Singular values: $\sigma_1 = 693$, $\sigma_2 = 416$, $\sigma_3 = 202$, $\sigma_4 = 142$, $\sigma_5 = 70$, $\sigma_6 = 43$

Test 4: Horizontal Displacement & Rotation

Singular values: $\sigma_1 = 1,403$, $\sigma_2 = 1,046$, $\sigma_3 = 334$, $\sigma_4 = 165$, $\sigma_5 = 117$, $\sigma_6 = 71$

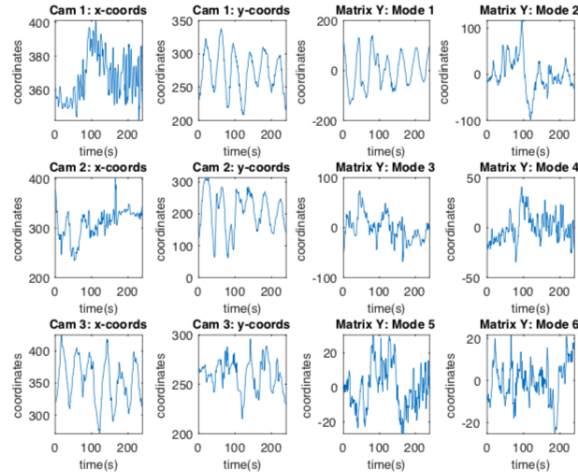


Figure 2: Test 2: Data in original basis and principal component basis

V. Summary and Conclusions

Through the evaluating our data in this exercise by reducing redundancy and we can come to the following conclusions

Test 1 - Ideal Signal (See Fig. 1)

The analysis of this data shows the large majority of the energy in this system can be attributed to one mode and therefore, a rank-1 approximation will provide very accurate data about the behavior of this system. This makes sense because in all three videos, the motion of the can is linear.

Test 2 - Camera Shake (See Fig. 2)

In this case, we see that the the energy is more spread out throughout the modes. Although most of the energy is still contained in one mode, the energy levels in the 2nd and 3rd modes have increased compared to those in the ideal case. When viewing the footage from Cam 2, we can see that some of the camera shake is in the direction of the paint can's oscillation. This would have less of a negative impact on the energy represented by the 1st mode than camera shake that's perpendicular to the line of oscillation.

Test 3 - Horizontal Displacement (See Fig. 3)

In this case a horizontal displacement causes move in an arc perpendicular to the main line of oscillation for the entirety of the video. Of all the cases, the data in this set is the most spread out. Less of the energy is represented by the 1st mode and more represented by the second. This fits with what we know of a spring system. The

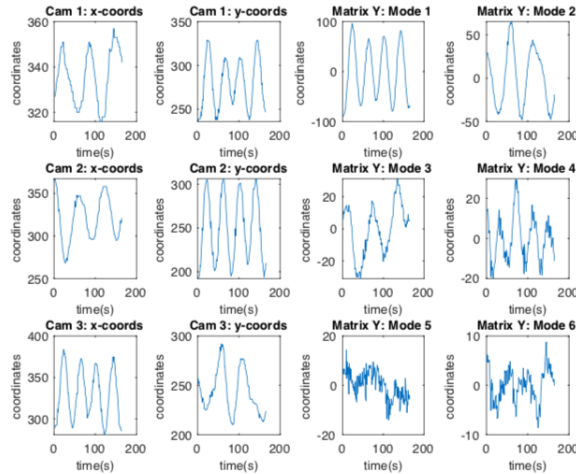


Figure 3: Test 3: Data in original basis and principal component basis

force of the spring increases as it stretches and so stretching away from the original line of oscillation will result in the paint can travelling less distance along that line.

Test 4 - Horizontal Displacement & Rotation (See Fig. 4)

While the energy in this system is more spread out than in cases 1 or 2, more of the energy is contained in the 1st mode here than in case 3. We can see from the video that the effect of the horizontal displacement does not last for the entire video like in case 3. The rotation of the can causes another force to act on it and so the motion becomes near linear again fairly quickly. Again, because we have chosen to track the can at the fixed point where it connects with the spring, this rotation does not have a great effect on our data.

Appendix A: Matlab Functions

implay() / imshow(): These functions takes a video or still image file as an argument and open Matlab's image or video viewer to view the file

ginput(): This function allows us to collect coordinates by clicking on points on a screen. When the function is active a set of axes appear on the screen with the origin at the mouse point

svd(): This function breaks down a matrix in to principal components, singular values and a rotation matrix

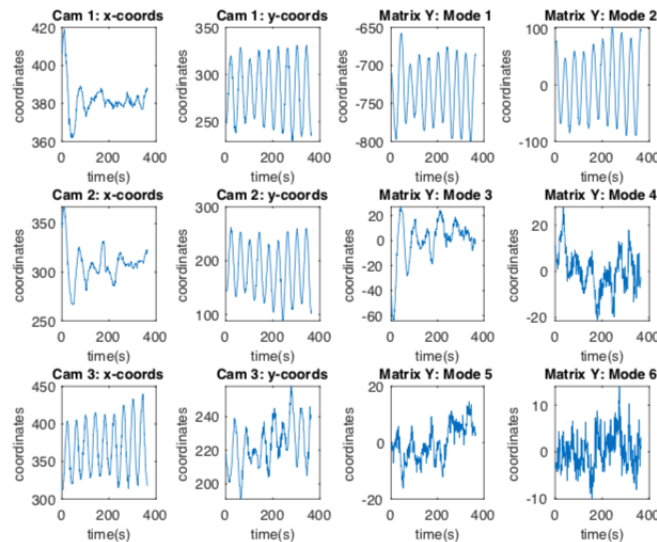


Figure 4: Test 4: Data in original basis and principal component basis

Appendix B: Matlab Code

```
1 %% HW 03 Principal Component Analysis
2
3 clear all; close all; clc
4
5 %% ginput() Camera 1 - Test 1
6
7 % first vertical min at 30 frames
8
9 clear all; close all; clc
10
11 load('cam1_1.mat')
12 implay(vidFrames1_1)
13 numFrames = size(vidFrames1_1,4);
14 cam1_1_coords = zeros(numFrames,2);
15
16 for j = 1:numFrames
17     X = vidFrames1_1(:, :, :, j);
18     imshow(X); drawnow
19     title(num2str(j), 'FontSize', 20)
20     % [x,y] = ginput(1);
21     % cam2_2_coords(j,1) = x;
22     % cam2_2_coords(j,2) = y;
23
24 end
25
26 csvwrite('cam1_1_coords.csv', cam1_1_coords)
27
28
29 %% ginput() Camera 2 - Test 1
30
31 clear all; close all; clc
32
33 load('cam2_1.mat')
34 implay(vidFrames2_1)
35 numFrames = size(vidFrames2_1,4);
36 cam2_1_coords = zeros(numFrames,2);
37
38 for j = 1:numFrames
39     X = vidFrames2_1(:, :, :, j);
40     imshow(X); drawnow
41     title(num2str(j), 'FontSize', 20)
42     % [x,y] = ginput(1);
43     % cam2_2_coords(j,1) = x;
44     % cam2_2_coords(j,2) = y;
45
46 end
47
48 csvwrite('cam2_1_coords.csv', cam2_1_coords)
49
```

```

50 %% ginput() Camera 3 - Test 1
51
52 clear all; close all; clc
53
54 load('cam3_1.mat')
55 implay(vidFrames3_1)
56 numFrames = size(vidFrames3_1,4);
57 cam3_1_coords = zeros(numFrames,2);
58
59 for j = 1:numFrames
60     X = vidFrames3_1(:, :, :, j);
61     imshow(X); drawnow
62     title(num2str(j), 'FontSize', 20)
63     % [x,y] = ginput(1);
64     % cam2_2_coords(j,1) = x;
65     % cam2_2_coords(j,2) = y;
66     pause(1)
67 end
68
69 csvwrite('cam3_1_coords.csv', cam3_1_coords)
70
71 %% ginput() Camera 1 - Test 2
72
73 clear all; close all; clc
74
75 load('cam1_2.mat');
76 implay(vidFrames1_2)
77 numFrames = size(vidFrames1_2,4);
78 cam1_2_coords = zeros(numFrames,2);
79
80 for j = 1:numFrames
81     X = vidFrames1_2(:, :, :, j);
82     imshow(X); drawnow
83     title(num2str(j), 'FontSize', 20)
84     % [x,y] = ginput(1);
85     % cam1_2_coords(j,1) = x;
86     % cam1_2_coords(j,2) = y;
87 end
88
89 csvwrite('cam1_2_coords.csv', cam1_2_coords)
90
91 %% ginput() Camera 2 - Test 2
92
93 clear all; close all; clc
94
95 load('cam2_2.mat');
96 implay(vidFrames2_2)
97 numFrames = size(vidFrames2_2,4);
98 cam2_2_coords = zeros(numFrames,2);
99
100 for j = 1:numFrames

```



```

101     X = vidFrames2_2(:, :, :, j);
102     imshow(X); drawnow
103     title(num2str(j), 'FontSize', 20)
104     %     [x,y] = ginput(1);
105     %     cam2_2_coords(j,1) = x;
106     %     cam2_2_coords(j,2) = y;
107 end
108
109 % csvwrite('cam2_2_coords.csv', cam2_2_coords)
110
111 %% ginput() Camera 3 - Test 2
112
113 clear all; close all; clc
114
115 load('cam3_2.mat')
116 implay(vidFrames3_2)
117 numFrames = size(vidFrames3_2,4);
118 cam3_2_coords = zeros(numFrames,2);
119
120 for j = 1:numFrames
121     X = vidFrames3_2(:, :, :, j);
122     imshow(X); drawnow
123     title(num2str(j), 'FontSize', 20)
124     %     [x,y] = ginput(1);
125     %     cam2_2_coords(j,1) = x;
126     %     cam2_2_coords(j,2) = y;
127
128 end
129
130 csvwrite('cam3_2_coords.csv', cam3_2_coords)
131
132 %% ginput() Camera 1 - Test 3
133
134 clear all; close all; clc
135
136 load('cam1_3.mat');
137 numFrames = size(vidFrames1_3,4);
138 cam1_3_coords = zeros(numFrames,2);
139
140 for j = 1:239
141     X = vidFrames1_3(:, :, :, j);
142     imshow(X); drawnow
143     title(num2str(j), 'FontSize', 20)
144     [x,y] = ginput(1);
145     cam1_3_coords(j,1) = x;
146     cam1_3_coords(j,2) = y;
147     % pause(0.75)
148 end
149
150 csvwrite('cam1_3_coords.csv', cam1_3_coords)
151

```

```

152 %% ginput() Camera 2 - Test 3
153
154 clear all; close all; clc
155
156 load('cam2_3.mat');
157 numFrames = size(vidFrames2_3,4);
158 cam2_3_coords = zeros(numFrames,2);
159
160 for j = 1:numFrames
161     X = vidFrames2_3(:, :, :, j);
162     imshow(X); drawnow
163     title(num2str(j), 'FontSize', 20)
164     [x,y] = ginput(1);
165     cam2_3_coords(j,1) = x;
166     cam2_3_coords(j,2) = y;
167     % pause(0.75)
168 end
169
170 csvwrite('cam2_3_coords.csv', cam2_3_coords)
171
172 %% ginput() Camera 3 - Test 3
173
174 clear all; close all; clc
175
176 load('cam3_3.mat');
177 numFrames = size(vidFrames3_3,4);
178 cam3_3_coords = zeros(numFrames,2);
179
180 for j = 1:numFrames
181     X = vidFrames3_3(:, :, :, j);
182     imshow(X); drawnow
183     title(num2str(j), 'FontSize', 20)
184     [x,y] = ginput(1);
185     cam3_3_coords(j,1) = x;
186     cam3_3_coords(j,2) = y;
187     % pause(0.75)
188 end
189
190 csvwrite('cam3_3_coords.csv', cam3_3_coords)
191
192 %% ginput() Camera 1 - Test 4
193
194 clear all; close all; clc
195
196 load('cam1_4.mat')
197 % implay(vidFrames1_4)
198 numFrames = size(vidFrames1_4,4);
199 % clippedGrey = zeros(380,80,1,226); % to hold the clipped data\
200 cam1_4_coords = zeros(numFrames,2);
201
202 for j = 1:392

```

```

203     X = vidFrames1_4(:, :, :, j);
204     imshow(X); drawnow
205     %newFrame = im2double(X(50:429, 310:389));
206     %clippedGrey(:, :, :, j) = newFrame;
207     %imshow(clippedGrey(:, :, :, j)); drawnow
208     title(num2str(j))
209     [x,y] = ginput(1);
210     cam1_4_coords(j,1) = x;
211     cam1_4_coords(j,2) = y;
212     pause(1)
213 end
214
215 csvwrite('cam1_4_coords.csv', cam1_4_coords)
216
217 %% ginput() Camera 2 - Test 4
218
219 clear all; close all; clc
220
221 load('cam2_4.mat')
222 % implay(vidFrames2_4)
223 numFrames = size(vidFrames2_4,4);
224 cam2_4_coords = zeros(numFrames,2);
225
226 for j = 1:405
227     X = vidFrames2_4(:, :, :, j);
228     imshow(X); drawnow
229     title(num2str(j), 'FontSize', 20)
230
231     objectRegion = [150 150 300 250];
232     if (mod(j,2) ==0)
233         objectImage = insertShape(X, 'Rectangle', objectRegion, ...
234             'Color', 'blue', 'Linewidth', 2);
235         figure(2);
236         imshow(objectImage);
237     else
238         objectImage = insertShape(X, 'Rectangle', objectRegion, ...
239             'Color', 'red', 'Linewidth', 2);
240         figure(2);
241         imshow(objectImage);
242     end
243
244     [x,y] = ginput(1);
245     cam2_4_coords(j,1) = x;
246     cam2_4_coords(j,2) = y;
247     pause(0.75)
248 end
249
250 csvwrite('cam2_4_coords.csv', cam2_4_coords)
251
252
253 %% ginput() Camera 3 - Test 4

```

```

254
255 clear all; close all; clc
256
257 load( 'cam3_4.mat' )
258 % implay(vidFrames3_4)
259 numFrames = size(vidFrames3_4,4);
260 cam3_4_coords = zeros(numFrames,2);
261
262 for j = 1:394
263     X = vidFrames3_4(:, :, :, j);
264     imshow(X); drawnow
265     title(num2str(j), 'Fontsize', 20)
266
267     objectRegion = [150 150 300 250];
268     if (mod(j,2) ==0)
269         objectImage = insertShape(X, 'Rectangle', objectRegion, ...
270             'Color', 'blue', 'Linewidth', 2);
271         figure(2);
272         imshow(objectImage);
273     else
274         objectImage = insertShape(X, 'Rectangle', objectRegion, ...
275             'Color', 'red', 'Linewidth', 2);
276         figure(2);
277         imshow(objectImage);
278     end
279
280     [x,y] = ginput(1);
281     cam3_4_coords(j,1) = x;
282     cam3_4_coords(j,2) = y;
283     % pause(0.75)
284 end
285
286 csvwrite( 'cam3_4_coords.csv', cam3_4_coords)
287
288 %% Build the Matrices
289
290 % build matrix A1
291
292 % get full coordinates from csv file
293 test1_1_coords = table2array(readtable( 'coordinates/cam1_1_coords.csv' ))
294     ;
295 test1_2_coords = table2array(readtable( 'coordinates/cam2_1_coords.csv' ))
296     ;
297 test1_3_coords = table2array(readtable( 'coordinates/cam3_1_coords.csv' ))
298     ;
299
300 % adjusted coordinates
301 subset_1_1 = test1_1_coords(10:209, 1:2);
302 subset_1_2 = test1_2_coords(19:218, 1:2);
303 subset_1_3 = test1_3_coords(08:207, 1:2);
304

```

```

302 % create the matrix the raw data
303 A_1 = zeros(6,length(subset_1_1));
304 n = length(A_1(1,:));
305
306 for j=1:n
307     A_1(1,j) = subset_1_1(j,1);
308     A_1(2,j) = subset_1_1(j,2);
309     A_1(3,j) = subset_1_2(j,1);
310     A_1(4,j) = subset_1_2(j,2);
311     A_1(5,j) = subset_1_3(j,1);
312     A_1(6,j) = subset_1_3(j,2);
313 end
314
315 csvwrite('A_1.csv',A_1)
316
317 % build matrix A2
318
319 % get full coordinates from csv file
320 test2_1_coors = table2array(readtable('coordinates/cam1_2_coors.csv'))
321     ;
322 test2_2_coors = table2array(readtable('coordinates/cam2_2_coors.csv'))
323     ;
324 test2_3_coors = table2array(readtable('coordinates/cam3_2_coors.csv'))
325     ;
326
327 % adjusted coordinates
328 subset_2_1 = test2_1_coors(35:274, 1:2);
329 subset_2_2 = test2_2_coors(18:257, 1:2);
330 subset_2_3 = test2_3_coors(40:279, 1:2);
331
332 % create the matrix for the raw data
333 A_2 = zeros(6,length(subset_2_1));
334 n = length(A_2(1,:));
335
336 for j=1:n
337     A_2(1,j) = subset_2_1(j,1);
338     A_2(2,j) = subset_2_1(j,2);
339     A_2(3,j) = subset_2_2(j,1);
340     A_2(4,j) = subset_2_2(j,2);
341     A_2(5,j) = subset_2_3(j,1);
342     A_2(6,j) = subset_2_3(j,2);
343 end
344
345 csvwrite('A_2.csv',A_2)
346
347 % build matrix A3
348
349 % get full coordinates from csv file
350 test3_1_coors = table2array(readtable('coordinates/cam1_3_coors.csv'))
351     ;
352 test3_2_coors = table2array(readtable('coordinates/cam2_3_coors.csv'))
353     ;

```

```

348 test3_3_coords = table2array(readtable('coordinates/cam3_3_coords.csv'))
349 ;
350 % adjusted coordinates
351 subset_3_1 = test3_1_coords(35:199, 1:2);
352 subset_3_2 = test3_2_coords(25:189, 1:2);
353 subset_3_3 = test3_3_coords(28:192, 1:2);
354
355 % create the matrix for the raw data
356 A_3 = zeros(6,length(subset_3_1));
357 n = length(A_3(1,:));
358
359 for j=1:length(subset_3_1)
360     A_3(1,j) = subset_3_1(j,1);
361     A_3(2,j) = subset_3_1(j,2);
362     A_3(3,j) = subset_3_2(j,1);
363     A_3(4,j) = subset_3_2(j,2);
364     A_3(5,j) = subset_3_3(j,1);
365     A_3(6,j) = subset_3_3(j,2);
366 end
367
368 csvwrite('A_3.csv',A_3)
369
370 % build matrix A4
371
372 % get full coordinates from csv file
373 test4_1_coords = table2array(readtable('coordinates/cam1_4_coords.csv'))
374 ;
375 test4_2_coords = table2array(readtable('coordinates/cam2_4_coords.csv'))
376 ;
377 test4_3_coords = table2array(readtable('coordinates/cam3_4_coords.csv'))
378 ;
379
380 % adjusted coordinates
381 subset_4_1 = test4_1_coords(12:375, 1:2);
382 subset_4_2 = test4_2_coords(18:381, 1:2);
383 subset_4_3 = test4_3_coords(11:374, 1:2);
384
385 % create the matrix for the raw data
386 A_4 = zeros(6,length(subset_4_1));
387 n = length(A_4(1,:));
388
389 for j=1:length(subset_4_1)
390     A_4(1,j) = subset_4_1(j,1);
391     A_4(2,j) = subset_4_1(j,2);
392     A_4(3,j) = subset_4_2(j,1);
393     A_4(4,j) = subset_4_2(j,2);
394     A_4(5,j) = subset_4_3(j,1);
395     A_4(6,j) = subset_4_3(j,2);
396 end
397

```

```

395 csvwrite('A_4.csv',A_4)
396
397 %% Test #1
398
399 % Singular Value Decomposition
400
401 clear all; close all; clc;
402
403 % original data matrix
404 A1 = readtable('A_1.csv');
405 A1 = table2array(A1);
406
407 % number of time samples
408 n = length(A1(1,:));
409
410 % plot the A coordinates in each of the 6 dimension over time
411 figure(1)
412 subplot(3,4,1);
413 plot(A1(1,:))
414 title("Cam 1: x-coords", 'FontSize', 10)
415 xlabel('time(s)', 'FontSize', 10)
416 ylabel('coordinates', 'FontSize', 10)
417 subplot(3,4,2);
418 plot(A1(2,:))
419 title("Cam 1: y-coords", 'FontSize', 10)
420 xlabel('time(s)', 'FontSize', 10)
421 ylabel('coordinates', 'FontSize', 10)
422 subplot(3,4,5);
423 plot(A1(3,:))
424 title("Cam 2: x-coords", 'FontSize', 10)
425 xlabel('time(s)', 'FontSize', 10)
426 ylabel('coordinates', 'FontSize', 10)
427 subplot(3,4,6);
428 plot(A1(4,:))
429 title("Cam 2: y-coords", 'FontSize', 10)
430 xlabel('time(s)', 'FontSize', 10)
431 ylabel('coordinates', 'FontSize', 10)
432 subplot(3,4,9);
433 plot(A1(5,:))
434 title("Cam 3: x-coords", 'FontSize', 10)
435 xlabel('time(s)', 'FontSize', 10)
436 ylabel('coordinates', 'FontSize', 10)
437 subplot(3,4,10);
438 plot(A1(6,:))
439 title("Cam 3: y-coords", 'FontSize', 10)
440 xlabel('time(s)', 'FontSize', 10)
441 ylabel('coordinates', 'FontSize', 10)
442
443
444 % normalize the rows by subtracting the mean
445 for j=1:length(A1(:,1))

```

```

446     A1(j,:) = A1(j,:)-mean(A1(j,:));
447 end
448
449
450 % get the SVD of A
451 [U1,S1,~] = svd(A1);
452
453 % find the data in the new basis
454 Y1 = U1'*A1;
455
456 % plot the Y coordinates in each of the 6 dimension over time
457
458 subplot(3,4,3);
459 plot(Y1(1,:))
460 title("Matrix Y: Mode 1", 'FontSize', 10)
461 xlabel('time(s)', 'FontSize', 10)
462 ylabel('coordinates', 'FontSize', 10)
463 subplot(3,4,4);
464 plot(Y1(2,:))
465 title("Matrix Y: Mode 2", 'FontSize', 10)
466 xlabel('time(s)', 'FontSize', 10)
467 ylabel('coordinates', 'FontSize', 10)
468 subplot(3,4,7);
469 plot(Y1(3,:))
470 title("Matrix Y: Mode 3", 'FontSize', 10)
471 xlabel('time(s)', 'FontSize', 10)
472 ylabel('coordinates', 'FontSize', 10)
473 subplot(3,4,8);
474 plot(Y1(4,:))
475 title("Matrix Y: Mode 4", 'FontSize', 10)
476 xlabel('time(s)', 'FontSize', 10)
477 ylabel('coordinates', 'FontSize', 10)
478 subplot(3,4,11);
479 plot(Y1(5,:))
480 title("Matrix Y: Mode 5", 'FontSize', 10)
481 xlabel('time(s)', 'FontSize', 10)
482 ylabel('coordinates', 'FontSize', 10)
483 subplot(3,4,12);
484 plot(Y1(6,:))
485 title("Matrix Y: Mode 6", 'FontSize', 10)
486 xlabel('time(s)', 'FontSize', 10)
487 ylabel('coordinates', 'FontSize', 10)
488
489
490 % get the diagonalized covariance matrix
491 C1 = (1/(n-1))*S1.^2;
492
493 % calculate energy in each mode
494 energy = S1./sum(sum(S1));
495 energy_squared = S1.^2./sum(sum(S1.^2));
496 sum(sum(energy))

```



```

497 sum(sum(energy_squared))
498
499 % Plot singular values and energy
500 figure(3)
501 subplot(2,2,1)
502 plot(S1, 'ko', 'Linewidth', 2)
503 % axis([0 25 0 50])
504 ylabel('\sigma')
505 set(gca, 'FontSize', 12, 'Xtick', 0:6)
506
507 subplot(2,2,2)
508 semilogy(S1, 'ko', 'Linewidth', 2)
509 % axis([0 25 10^-(18) 10^5])
510 ylabel('\Sigma (log scale)')
511 set(gca, 'FontSize', 12, 'Xtick', 0:6, 'Ytick', logspace(-15, 5, 5))
512
513 subplot(2,2,3)
514 plot(S1.^2/sum(S1.^2), 'ko', 'Linewidth', 2)
515 % axis([0 25 0 1])
516 ylabel('Energy')
517 set(gca, 'FontSize', 12, 'Xtick', 0:6)
518
519 subplot(2,2,4)
520 semilogy(S1.^2/sum(S1.^2), 'ko', 'Linewidth', 2)
521 % axis([0 25 10^-(18) 10^5])
522 ylabel('Energy (log scale)')
523 set(gca, 'FontSize', 12, 'Xtick', 0:6, 'Ytick', logspace(-15, 0, 4))
524 annotation('textbox', [0 0.9 1 0.1], ...
525     'String', 'Test 1 - Ideal Harmonic Motion, Singular Values and
526     Energy', ...
527     'FontSize', 15, ...
528     'Fontweight', 'bold', ...
529     'EdgeColor', 'none', ...
530     'HorizontalAlignment', 'center')
531 %% Test #2
532
533 % Singular Value Decomposition
534
535 clear all; close all; clc;
536
537 % original data matrix
538 A2 = readtable('A_2.csv');
539 A2 = table2array(A2);
540
541 % number of time samples
542 n = length(A2(1,:));
543
544 % plot the A coordinates in each of the 6 dimension over time
545 figure(1)
546 subplot(3,4,1);

```

```

547 plot(A2(1,:))
548 title("Cam 1: x-coords", 'FontSize', 10)
549 xlabel('time(s)', 'FontSize', 10)
550 ylabel('coordinates', 'FontSize', 10)
551 subplot(3,4,2);
552 plot(A2(2,:))
553 title("Cam 1: y-coords", 'FontSize', 10)
554 xlabel('time(s)', 'FontSize', 10)
555 ylabel('coordinates', 'FontSize', 10)
556 subplot(3,4,5);
557 plot(A2(3,:))
558 title("Cam 2: x-coords", 'FontSize', 10)
559 xlabel('time(s)', 'FontSize', 10)
560 ylabel('coordinates', 'FontSize', 10)
561 subplot(3,4,6);
562 plot(A2(4,:))
563 title("Cam 2: y-coords", 'FontSize', 10)
564 xlabel('time(s)', 'FontSize', 10)
565 ylabel('coordinates', 'FontSize', 10)
566 subplot(3,4,9);
567 plot(A2(5,:))
568 title("Cam 3: x-coords", 'FontSize', 10)
569 xlabel('time(s)', 'FontSize', 10)
570 ylabel('coordinates', 'FontSize', 10)
571 subplot(3,4,10);
572 plot(A2(6,:))
573 title("Cam 3: y-coords", 'FontSize', 10)
574 xlabel('time(s)', 'FontSize', 10)
575 ylabel('coordinates', 'FontSize', 10)
576
577 % normalize the rows by subtracting the mean
578 for j=1:length(A2(:,1))
579     A2(j,:) = A2(j,:)-mean(A2(j,:));
580 end
581
582
583 % get the SVD of A
584 [U2,S2,~] = svd(A2);
585
586 % find the data in the new basis
587 Y2 = U2'*A2;
588
589 % plot the Y coordinates in each of the 6 dimension over time
590
591 subplot(3,4,3);
592 plot(Y2(1,:))
593 title("Matrix Y: Mode 1", 'FontSize', 10)
594 xlabel('time(s)', 'FontSize', 10)
595 ylabel('coordinates', 'FontSize', 10)
596 subplot(3,4,4);
597 plot(Y2(2,:))

```

```

598 title("Matrix Y: Mode 2", 'FontSize', 10)
599 xlabel('time(s)', 'FontSize', 10)
600 ylabel('coordinates', 'FontSize', 10)
601 subplot(3,4,7);
602 plot(Y2(3,:))
603 title("Matrix Y: Mode 3", 'FontSize', 10)
604 xlabel('time(s)', 'FontSize', 10)
605 ylabel('coordinates', 'FontSize', 10)
606 subplot(3,4,8);
607 plot(Y2(4,:))
608 title("Matrix Y: Mode 4", 'FontSize', 10)
609 xlabel('time(s)', 'FontSize', 10)
610 ylabel('coordinates', 'FontSize', 10)
611 subplot(3,4,11);
612 plot(Y2(5,:))
613 title("Matrix Y: Mode 5", 'FontSize', 10)
614 xlabel('time(s)', 'FontSize', 10)
615 ylabel('coordinates', 'FontSize', 10)
616 subplot(3,4,12);
617 plot(Y2(6,:))
618 title("Matrix Y: Mode 6", 'FontSize', 10)
619 xlabel('time(s)', 'FontSize', 10)
620 ylabel('coordinates', 'FontSize', 10)
621
622 % get the diagonalized covariance matrix
623 C2 = (1/(n-1))*S2.^2;
624
625 % calculate energy in each mode
626 energy = S2./sum(sum(S2));
627 energy_squared = S2.^2./sum(sum(S2.^2));
628 sum(sum(energy))
629 sum(sum(energy_squared))
630
631 % Plot singular values and energy
632 figure(3)
633 subplot(2,2,1)
634 plot(S2,'ko','Linewidth',2)
635 % axis([0 25 0 50])
636 ylabel('\sigma')
637 set(gca,'FontSize',12,'Xtick',0:6)
638
639 subplot(2,2,2)
640 semilogy(S2,'ko','Linewidth',2)
641 % axis([0 25 10^-(18) 10^5])
642 ylabel('\Sigma (log scale)')
643 set(gca,'FontSize',12,'Xtick',0:6,'Ytick',logspace(-15,5,5))
644
645 subplot(2,2,3)
646 plot(S2.^2/sum(S2.^2),'ko','Linewidth',2)
647 % axis([0 25 0 1])
648 ylabel('Energy')

```

```

649 set(gca,'FontSize',12,'Xtick',0:6)
650
651 subplot(2,2,4)
652 semilogy(S2.^2/sum(S2.^2),'ko','Linewidth',2)
653 % axis([0 25 10^-(18) 10^5])
654 ylabel('Energy (log scale)')
655 set(gca,'FontSize',12,'Xtick',0:6,'Ytick',logspace(-15,0,4))
656 annotation('textbox',[0 0.9 1 0.1], ...
657     'String','Test 2 - Camera Shake, Singular Values and Energy', ...
658     'FontSize', 13,...
659     'Fontweight','bold',...
660     'EdgeColor','none', ...
661     'HorizontalAlignment','center')
662
663 %% Test #3 HORIZONTAL DISPLACEMENT
664
665 % Singular Value Decomposition
666
667 clear all; close all; clc;
668
669 % original data matrix
670 A3 = readtable('A_3.csv');
671 A3 = table2array(A3);
672
673 % number of time samples
674 n = length(A3(1,:));
675
676 % plot the A coordinates in each of the 6 dimension over time
677 figure(1)
678 subplot(3,4,1);
679 plot(A3(1,:))
680 title("Cam 1: x-coords", 'FontSize', 10)
681 xlabel('time(s)', 'FontSize', 10)
682 ylabel('coordinates', 'FontSize', 10)
683 subplot(3,4,2);
684 plot(A3(2,:))
685 title("Cam 1: y-coords", 'FontSize', 10)
686 xlabel('time(s)', 'FontSize', 10)
687 ylabel('coordinates', 'FontSize', 10)
688 subplot(3,4,5);
689 plot(A3(3,:))
690 title("Cam 2: x-coords", 'FontSize', 10)
691 xlabel('time(s)', 'FontSize', 10)
692 ylabel('coordinates', 'FontSize', 10)
693 subplot(3,4,6);
694 plot(A3(4,:))
695 title("Cam 2: y-coords", 'FontSize', 10)
696 xlabel('time(s)', 'FontSize', 10)
697 ylabel('coordinates', 'FontSize', 10)
698 subplot(3,4,9);
699 plot(A3(5,:))

```

```

700 title("Cam 3: x-coords", 'FontSize', 10)
701 xlabel('time(s)', 'FontSize', 10)
702 ylabel('coordinates', 'FontSize', 10)
703 subplot(3,4,10);
704 plot(A3(6,:))
705 title("Cam 3: y-coords", 'FontSize', 10)
706 xlabel('time(s)', 'FontSize', 10)
707 ylabel('coordinates', 'FontSize', 10)
708
709 % normalize the rows by subtracting the mean
710 for j=1:length(A3(:,1))
711     A3(j,:) = A3(j,:)-mean(A3(j,:));
712 end
713
714
715 % get the SVD of A
716 [U3,S3,~] = svd(A3);
717
718 % find the data in the new basis
719 Y3 = U3'*A3;
720
721 % plot the Y coordinates in each of the 6 dimension over time
722 subplot(3,4,3);
723 plot(Y3(1,:))
724 title("Matrix Y: Mode 1", 'FontSize', 10)
725 xlabel('time(s)', 'FontSize', 10)
726 ylabel('coordinates', 'FontSize', 10)
727 subplot(3,4,4);
728 plot(Y3(2,:))
729 title("Matrix Y: Mode 2", 'FontSize', 10)
730 xlabel('time(s)', 'FontSize', 10)
731 ylabel('coordinates', 'FontSize', 10)
732 subplot(3,4,7);
733 plot(Y3(3,:))
734 title("Matrix Y: Mode 3", 'FontSize', 10)
735 xlabel('time(s)', 'FontSize', 10)
736 ylabel('coordinates', 'FontSize', 10)
737 subplot(3,4,8);
738 plot(Y3(4,:))
739 title("Matrix Y: Mode 4", 'FontSize', 10)
740 xlabel('time(s)', 'FontSize', 10)
741 ylabel('coordinates', 'FontSize', 10)
742 subplot(3,4,11);
743 plot(Y3(5,:))
744 title("Matrix Y: Mode 5", 'FontSize', 10)
745 xlabel('time(s)', 'FontSize', 10)
746 ylabel('coordinates', 'FontSize', 10)
747 subplot(3,4,12);
748 plot(Y3(6,:))
749 title("Matrix Y: Mode 6", 'FontSize', 10)
750 xlabel('time(s)', 'FontSize', 10)

```

```

751 ylabel('coordinates', 'FontSize', 10)
752
753 % get the diagonalized covariance matrix
754 C3 = (1/(n-1))*S3.^2;
755
756 % calculate energy in each mode
757 energy = S3./sum(sum(S3));
758 energy_squared = S3.^2./sum(sum(S3.^2));
759 sum(sum(energy))
760 sum(sum(energy_squared))
761
762 % Plot singular values and energy
763 figure(3)
764 subplot(2,2,1)
765 plot(S3,'ko','Linewidth',2)
766 % axis([0 25 0 50])
767 ylabel('\sigma')
768 set(gca,'FontSize',12,'Xtick',0:6)
769
770 subplot(2,2,2)
771 semilogy(S3,'ko','Linewidth',2)
772 % axis([0 25 10^-(18) 10^5])
773 ylabel('\Sigma (log scale)')
774 set(gca,'FontSize',12,'Xtick',0:6,'Ytick',logspace(-15,5,5))
775
776 subplot(2,2,3)
777 plot(S3.^2/sum(S3.^2),'ko','Linewidth',2)
778 % axis([0 25 0 1])
779 ylabel('Energy')
780 set(gca,'FontSize',12,'Xtick',0:6)
781
782 subplot(2,2,4)
783 semilogy(S3.^2/sum(S3.^2),'ko','Linewidth',2)
784 % axis([0 25 10^-(18) 10^5])
785 ylabel('Energy (log scale)')
786 set(gca,'FontSize',12,'Xtick',0:6,'Ytick',logspace(-15,0,4))
787 annotation('textbox', [0 0.9 1 0.1], ...
788     'String', 'Test 3 - Horizontal Displacement, Singular Values and
789     Energy', ...
789     'FontSize', 13,...
790     'Fontweight', 'bold',...
791     'EdgeColor', 'none', ...
792     'HorizontalAlignment', 'center')
793
794 %% Test #4
795
796 close all; clear all; clc;
797
798 % Singular Value Decomposition
799
800 clear all; close all; clc;

```

```

801
802 % original data matrix
803 A4 = readtable('A_4.csv');
804 A4 = table2array(A4);
805
806 % number of time samples
807 n = length(A4(1,:));
808
809 % plot the A coordinates in each of the 6 dimension over time
810 figure(1)
811 subplot(3,4,1);
812 plot(A4(1,:))
813 title("Cam 1: x-coords", 'FontSize', 10)
814 xlabel('time(s)', 'FontSize', 10)
815 ylabel('coordinates', 'FontSize', 10)
816 subplot(3,4,2);
817 plot(A4(2,:))
818 title("Cam 1: y-coords", 'FontSize', 10)
819 xlabel('time(s)', 'FontSize', 10)
820 ylabel('coordinates', 'FontSize', 10)
821 subplot(3,4,5);
822 plot(A4(3,:))
823 title("Cam 2: x-coords", 'FontSize', 10)
824 xlabel('time(s)', 'FontSize', 10)
825 ylabel('coordinates', 'FontSize', 10)
826 subplot(3,4,6);
827 plot(A4(4,:))
828 title("Cam 2: y-coords", 'FontSize', 10)
829 xlabel('time(s)', 'FontSize', 10)
830 ylabel('coordinates', 'FontSize', 10)
831 subplot(3,4,9);
832 plot(A4(5,:))
833 title("Cam 3: x-coords", 'FontSize', 10)
834 xlabel('time(s)', 'FontSize', 10)
835 ylabel('coordinates', 'FontSize', 10)
836 subplot(3,4,10);
837 plot(A4(6,:))
838 title("Cam 3: y-coords", 'FontSize', 10)
839 xlabel('time(s)', 'FontSize', 10)
840 ylabel('coordinates', 'FontSize', 10)
841
842
843 % get the SVD of A
844 [U4,S4,V4_a] = svd(A4, 'econ');
845
846 % find the data in the new basis
847 Y4 = U4'*A4;
848
849 % plot the Y coordinates in each of the 6 dimension over time
850 subplot(3,4,3);
851 plot(Y4(1,:))

```

```

852 title("Matrix Y: Mode 1", 'FontSize', 10)
853 xlabel('time(s)', 'FontSize', 10)
854 ylabel('coordinates', 'FontSize', 10)
855 subplot(3,4,4);
856 plot(Y4(2,:),)
857 title("Matrix Y: Mode 2", 'FontSize', 10)
858 xlabel('time(s)', 'FontSize', 10)
859 ylabel('coordinates', 'FontSize', 10)
860 subplot(3,4,7);
861 plot(Y4(3,:),)
862 title("Matrix Y: Mode 3", 'FontSize', 10)
863 xlabel('time(s)', 'FontSize', 10)
864 ylabel('coordinates', 'FontSize', 10)
865 subplot(3,4,8);
866 plot(Y4(4,:),)
867 title("Matrix Y: Mode 4", 'FontSize', 10)
868 xlabel('time(s)', 'FontSize', 10)
869 ylabel('coordinates', 'FontSize', 10)
870 subplot(3,4,11);
871 plot(Y4(5,:),)
872 title("Matrix Y: Mode 5", 'FontSize', 10)
873 xlabel('time(s)', 'FontSize', 10)
874 ylabel('coordinates', 'FontSize', 10)
875 subplot(3,4,12);
876 plot(Y4(6,:),)
877 title("Matrix Y: Mode 6", 'FontSize', 10)
878 xlabel('time(s)', 'FontSize', 10)
879 ylabel('coordinates', 'FontSize', 10)
880
881 % find the covariance matrix of the data in the new basis
882 C4 = (1/(n-1)).*Y4*Y4';
883
884 % diagonalize the covariance matrix with the SVD
885 [U4_C,S4_C,V4_C] = svd(C4, 'econ');
886
887 % calculate energy in each mode
888
889 energy1b=S4_C(1)/sum(S4_C)
890 energy2b = S4_C(1)^2/sum(S4_C.^2)
891
892 energy_1 = S4_C./sum(sum(S4_C));
893 energy_2 = S4_C.^2/sum(S4_C.^2);
894 energy_3 = S4_C.^2./sum(sum(S4_C.^2));
895
896 % Plot singular values and energy
897 figure(4)
898 subplot(2,2,1)
899 plot(S4_C,'ko','Linewidth',2)
900 % axis([0 25 0 50])
901 ylabel('\sigma')
902 set(gca,'FontSize',12,'Xtick',0:6)

```



```

903
904 subplot(2,2,2)
905 semilogy(S4_C, 'ko', 'Linewidth', 2)
906 % axis([0 25 10^-(18) 10^5])
907 ylabel('\Sigma (log scale)')
908 set(gca, 'FontSize', 12, 'Xtick', 0:6, 'Ytick', logspace(-15, 5, 5))
909
910 subplot(2,2,3)
911 plot(S4_C.^2/sum(S4_C.^2), 'ko', 'Linewidth', 2)
912 % axis([0 25 0 1])
913 ylabel('Energy')
914 set(gca, 'FontSize', 12, 'Xtick', 0:6)
915
916 subplot(2,2,4)
917 semilogy(S4_C.^2/sum(S4_C.^2), 'ko', 'Linewidth', 2)
918 % axis([0 25 10^-(18) 10^5])
919 ylabel('Energy (log scale)')
920 set(gca, 'FontSize', 12, 'Xtick', 0:6, 'Ytick', logspace(-15, 0, 4))
921 annotation('textbox', [0 0.9 1 0.1], ...
922     'String', 'Test 4 - Horizontal Displacement & Rotation', ...
923     'FontSize', 15, ...
924     'Fontweight', 'bold', ...
925     'EdgeColor', 'none', ...
926     'HorizontalAlignment', 'center')

```