Principal Component Analysis on Video Data

Htet Aung Myin

February 21, 2020

Abstract

The application of the Principal Component Analysis on different videos of a mass-spring system is performed. A white flash on the hanging mass acts as a tracker for data preparation. Four experiments are performed with three different camera angles for each of them. They include; an ideal test, a noisy test, a horizontally displaced test, and a rotational and horizontally displaced test.

1 Introduction and Overview

The problem given is regarding four tests situations with three camera angles for each situation. In the first test, a mass is moving normally in an oscillatory behavior. The second test introduced camera shake (noise) to the normally oscillating mass. In the third test, the mass is released off center to produce an horizontal displacement. The fourth test introduced a rotational as well as horizontal displacement to the mass. In each experiment, data would be over sampled. As there are three camera angles per experiment, there would be 6 variables corresponding to the X and Y axis of each camera, forming a 6 x N matrix, where N would be the number of frames.

2 Theoretical Background

2.1 Singular Value Decomposition

The Singular Value Decomposition transforms a matrix A into:

$$\mathbf{A} = \mathbf{U\Sigma V}^* \text{ where}$$
$$\mathbf{U} \in \mathbb{C}^{m\times m} \text{ is unitary}$$
$$\mathbf{V} \in \mathbb{C}^{n\times n} \text{ is unitary}$$
$$\mathbf{\Sigma} \in \mathbb{R}^{m\times n} \text{ is diagonal}$$

The SVD can be computed by the following

$$\begin{aligned}\mathbf{A}^T\mathbf{A} &= (\mathbf{U\Sigma V}^*)^T(\mathbf{U\Sigma V}^*)\\ &= \mathbf{V\Sigma U}^*\mathbf{U\Sigma V}^*\\ &= \mathbf{V\Sigma}^2\mathbf{V}^*\end{aligned}$$

$$\begin{aligned}\mathbf{AA}^T &= (\mathbf{U\Sigma V}^*)(\mathbf{U\Sigma V}^*)^T\\ &= \mathbf{U\Sigma V}^*\mathbf{V\Sigma U}^*\\ &= \mathbf{U\Sigma}^2\mathbf{U}^*\end{aligned}$$

$$\begin{aligned}\mathbf{A}^T\mathbf{AV} &= \mathbf{V}\Sigma^2\\ \mathbf{AA}^T\mathbf{U} &= \mathbf{U\Sigma}^2\end{aligned}$$

2.2 Principal Component Analysis

One of the primary applications of the SVD is for Principal Component Analysis (PCA)where it reduces the dimensions of the data set. The data

can be placed a matrix with $X \in \mathbb{R}^{m \times n}$, where m is number of measurements and n is number of data points taken over time. In this matrix, redundancy can be identified and reduced with the covariance matrix and is defined by:

$C_Y = \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T$

3 Algorithm Implementation and Development

The dataset consisted of x-y coordinates of the bucket over time. The procedure was similar for all four cases.

1. The file(s) are loaded in and the number of frames were found using the size command.
2. A filter function is then defined for each individual cases based on human estimations of where the mass could possible be. This narrows down the frame to where the mass is acting on the system. There may have been issues due to this.
3. Over each frame, the colors are gray scaled as the tracking of the white flash would be more covinenet.
4. A threshold is defined for each test case (threshold varies with test as some tests had too much motion.
5. The indices were found using the find command. Each coordinates were averaged and then added into a data array of size 2 x frame number.
6. This process is the repeated over each video for each experiment, giving three matrices.
7. As each video have different numbers of frames, the video with the shortest frame is identified. The other videos are then trimmed and added it to one large 6 x Frame Number matrix.
8. The mean of each row is subtracted from the data to allow the data be on the same scale, and the SVD of the transpose data matrix is taken.
9. The diagonal of S is taken to find the variance and plotted the variance as a function of principal components to see which were most significant.
10. The Principal Component is also plotted.

4 Computational Results

In the first test, a very high energy could be observed for the first principal component. This is ideal as the mass was only moving in one direction.

In the second test, despite the noise, similar to the first test, there could possibly be one principal component as it is still moving in one direction. However, noise may cause issues.

In the third test, A large variance can be observed for 3-4 principal components which could describe the motion of the mass. This is possibly caused by motion in both directions.

In the fourth test, A large variance can be observed for 3-4 principal components. This is possibly caused by motion in both directions along with rotational motion.

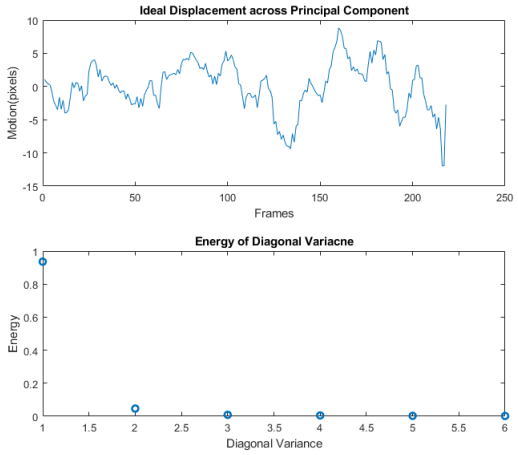

Figure 1: Test 1: Ideal Case

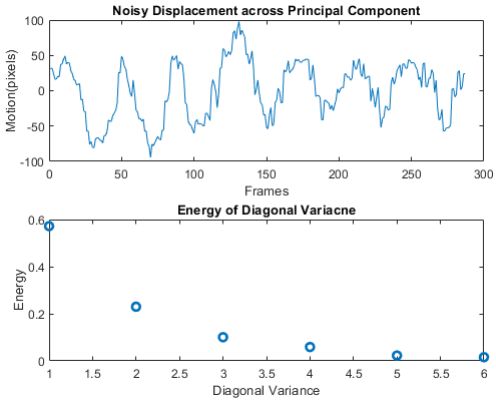

Figure 2: Test 2: Noisy Case

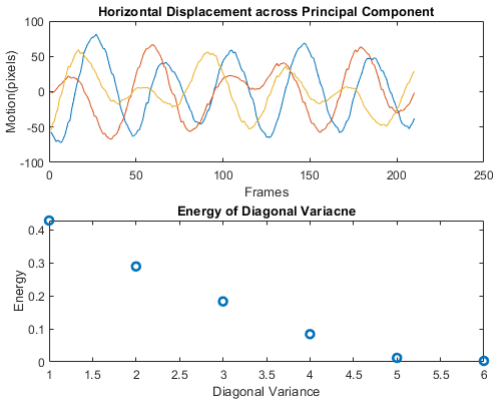

Figure 3: Test 3: Horizontal Displacement

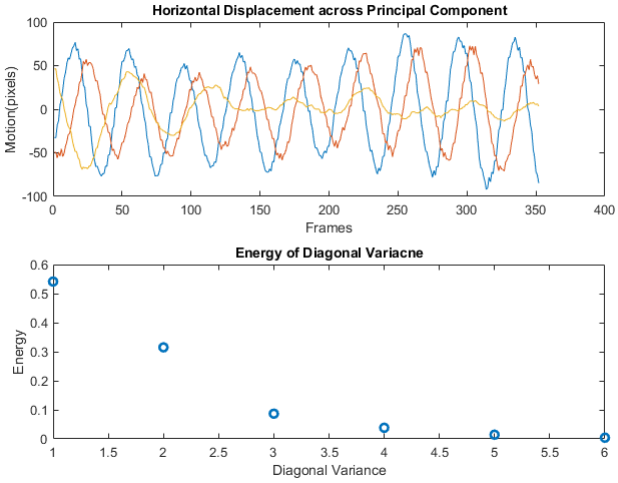

Figure 4: Test 4: Horizontal Displacement and rotation

5 Summary & Conclusion

PCA was successfully performed on each of the four tests as the oscillation was tracked.The principal components are accurate to a certain extent, however, the filter may have over sampling issues due to human error. In test 1 and 2, even with noise, the motion was extracted fairly well. In test 3 and 4, with

6 Appendix A.

- diag(X) = returns diagonal value of matrix X
- find(X) = returns a vector of index in a matrix
- zeros() = fills a matrix with zeros
- svd() = returns a diagonal matrix S along with unitary matrix U and V
- rgb2gray(X) : given an image X, grayscales it
- size(X) : returns dimension of matrix X
- length(X) : returns length of matrix X

7 Appendix B.

```
clear all; close all; clc;

%%
load('cam1_4.mat')
implay(vidFrames1_4)

%% Ideal Case Test 1

clear all; close all; clc;
load('cam1_1.mat')
numFrames1 = size(vidFrames1_1,4);

data1 = [];
filter = zeros (480 ,640) ;
filter (175:425 , 300:400 ) = 1;

%x 480, y 640
% x and y are swapped
for j = 1:numFrames1
    X = vidFrames1_1(:,:,:,j);
    G = double(rgb2gray(X));
    Xf = G .* filter ;
    th = Xf > 250;
    indx = find(th) ;
    [Y ,X] = ind2sub (size(th) , indx) ;
    data1 = [data1; mean(X), mean(Y)];
end


load('cam2_1.mat')
numFrames2 = size(vidFrames2_1,4);
filter = zeros (480 ,640) ;
filter (100:450 , 200:350 ) = 1;
data2 = [];
for j = 1:numFrames2
    X = vidFrames2_1(:,:,:,j);
    G = double(rgb2gray(X));
    Xf = G .* filter ;
```

```
39      th = Xf > 250;
40      indx = find(th) ;
41      [Y ,X] = ind2sub (size(th) , indx ) ;
42      data2 = [data2; mean(X), mean(Y) ];
43 end
44
45
46 load('cam3_1.mat')
47 data3 = [];
48 filter = zeros (480 ,640) ;
49 filter (200:300 , 210:500 ) = 1;
50
51 numFrames3 = size(vidFrames3_1 ,4);
52 for j = 1:numFrames3
53      X = vidFrames3_1(:,:,:,j);
54      G = double(rgb2gray(X));
55      Xf = G .* filter ;
56      th = Xf > 245;
57      indx = find(th) ;
58      [Y ,X] = ind2sub (size(th) , indx) ;
59      data3 = [data3; mean(X), mean(Y) ];
60 end
61
62 %% resizing ideal
63 [M, I] = min(data1(1:30,2));
64 data1 = data1(I:end,:);
65 [M, I] = min(data2(1:30,2));
66 data2 = data2(I:end,:);
67 [M, I] = min(data3(1:30,2));
68 data3 = data3(I:end,:);
69
70 data2 = data2(1:length(data1),:);
71 data3 = data3(1:length(data1),:);
72
73 %% PCA
74 dat_arr = [data1'; data2'; data3'];
75
76 [m,n]=size(dat_arr);
77 mn=mean(dat_arr,2);
78 dat_arr = dat_arr - repmat(mn,1,n);
79 [u,s,v]=svd(dat_arr'/sqrt(n-1));
```

```
lambda=diag(s).^2;
Y= dat_arr' * v;


%% Ideal plot



subplot(2,1,1)
plot(1:218, Y(:,6)), hold on
title("Ideal Displacement across Principal Component")
xlabel("Frames"); ylabel("Motion(pixels)")
subplot(2,1,2)
plot (1:6 , lambda/sum(lambda) ,'o', 'Linewidth', 2);
title("Energy of Diagonal Variacne");
xlabel("Diagonal Variance");
ylabel("Energy")
%% Noisy Case
clear all; close all; clc;
load('cam1_2.mat')
numFrames1 = size(vidFrames1_2,4);

data1 = [];
filter = zeros (480 ,640) ;
filter (175:425 , 300:450 ) = 1;

%x 480, y 640
% x and y are swapped
for j = 1:numFrames1
    X = vidFrames1_2(:,:,:,j);
    G = double(rgb2gray(X));
    Xf = G .* filter ;
    th = Xf > 250;
    indx = find(th) ;
    [Y ,X] = ind2sub (size(th) , indx) ;
    data1 = [data1; mean(X), mean(Y)];
end


load('cam2_2.mat')
numFrames2 = size(vidFrames2_2,4);
```

```
121
122 data2 = [];
123 filter = zeros (480 ,640) ;
124 filter (50:450 , 150:450 ) = 1;
125
126 %x 480, y 640
127 % x and y are swapped
128 for j = 1:numFrames2
129     X = vidFrames2_2(:,:,:,j);
130     G = double(rgb2gray(X));
131     Xf = G .* filter ;
132     th = Xf > 245;
133     indx = find(th) ;
134     [Y ,X] = ind2sub (size(th) , indx) ;
135     data2 = [data2; mean(X), mean(Y)];
136 end
137
138
139 load('cam3_2.mat')
140 numFrames3 = size(vidFrames3_2,4);
141
142 data3 = [];
143 filter = zeros (480 ,640) ;
144 filter (200:400 , 210:500 ) = 1;
145
146 %x 480, y 640
147 % x and y are swapped
148 for j = 1:numFrames1
149     X = vidFrames3_2(:,:,:,j);
150     G = double(rgb2gray(X));
151     Xf = G .* filter ;
152     th = Xf > 245;
153     indx = find(th) ;
154     [Y ,X] = ind2sub (size(th) , indx) ;
155     data3 = [data3; mean(X), mean(Y)];
156 end
157 %% resizing noisy
158 [M, I] = min(data1(1:30,2));
159 data1 = data1(I:end,:);
160 [M, I] = min(data2(1:30,2));
161 data2 = data2(I:end,:);
```

```
[M, I] = min(data3(1:30,2));
data3 = data3(I:end,:);

data2 = data2(1:length(data1),:);
data3 = data3(1:length(data1),:);
%% PCA Noisy
dat_arr = [data1'; data2'; data3'];
[m,n] = size(dat_arr);

mn=mean(dat_arr,2);
dat_arr = dat_arr - repmat(mn,1,n);
[u,s,v]=svd(dat_arr'/sqrt(n-1));
lambda=diag(s).^2;
Y=dat_arr' * v;

%% noisy plots
figure()


subplot(2,1,1)
plot(1:287, Y(:,2))
title("Noisy Displacement across Principal Component")
xlabel("Frames"); ylabel("Motion(pixels)")
subplot(2,1,2)
plot (1:6 , lambda/sum(lambda) ,'o', 'Linewidth', 2);
title("Energy of Diagonal Variacne");
xlabel("Diagonal Variance");
ylabel("Energy")
%% Horizontal Displacment
clear all; close all; clc;
load('cam1_3.mat')
numFrames1 = size(vidFrames1_3,4);

data1 = [];
filter = zeros (480 ,640) ;
filter (200:450 , 300:450 ) = 1;

%x 480, y 640
% x and y are swapped
for j = 1:numFrames1
    X = vidFrames1_3(:,:,:,j);
```

```
203     G = double(rgb2gray(X));
204     Xf = G .* filter ;
205     th = Xf > 250;
206     indx = find(th) ;
207     [Y ,X] = ind2sub (size(th) , indx) ;
208     data1 = [data1; mean(X), mean(Y)];
209 end
210
211 load('cam2_3.mat')
212 numFrames2 = size(vidFrames2_3,4);
213
214 data2 = [];
215 filter = zeros (480 ,640) ;
216 filter (100:450 , 150:425 ) = 1;
217
218 %x 480, y 640
219 % x and y are swapped
220 for j = 1:numFrames2
221     X = vidFrames2_3(:,:,:,j);
222     G = double(rgb2gray(X));
223     Xf = G .* filter ;
224     th = Xf > 245;
225     indx = find(th) ;
226     [Y ,X] = ind2sub (size(th) , indx) ;
227     data2 = [data2; mean(X), mean(Y)];
228 end
229
230
231 load('cam3_3.mat')
232 numFrames3 = size(vidFrames3_3,4);
233
234 data3 = [];
235 filter = zeros (480 ,640) ;
236 filter (150:360 , 210:500 ) = 1;
237
238 %x 480, y 640
239 % x and y are swapped
240 for j = 1:numFrames3
241     X = vidFrames3_3(:,:,:,j);
242     G = double(rgb2gray(X));
243     Xf = G .* filter ;
```

```
244     th = Xf > 245;
245     indx = find(th) ;
246     [Y ,X] = ind2sub (size(th) , indx) ;
247     data3 = [data3; mean(X), mean(Y)];
248 end
249 %% resizing horiz dis
250 [M, I] = min(data1(1:30,2));
251 data1 = data1(I:end,:);
252 [M, I] = min(data2(1:30,2));
253 data2 = data2(I:end,:);
254 [M, I] = min(data3(1:30,2));
255 data3 = data3(I:end,:);
256
257 data2 = data2(1:length(data1),:);
258 data3 = data3(1:length(data1),:);
259
260 %% PCA  horiz
261 dat_arr = [data1'; data2'; data3'];
262 [m,n] = size(dat_arr);
263
264 mn=mean(dat_arr,2);
265 dat_arr = dat_arr - repmat(mn,1,n);
266 [u,s,v]=svd(dat_arr'/sqrt(n-1));
267 lambda=diag(s).^2;
268 Y=dat_arr' * v;
269
270 %% hoirz plots
271 figure()
272
273
274 subplot(2,1,1)
275 plot(1:210, Y(:,1),1:210, Y(:,2),1:210, Y(:,3))
276 title("Horizontal Displacement across Principal
      Component")
277 xlabel("Frames"); ylabel("Motion(pixels)")
278 subplot(2,1,2)
279 plot (1:6 , lambda/sum(lambda) ,'o', 'Linewidth', 2);
280 title("Energy of Diagonal Variacne");
281 xlabel("Diagonal Variance");
282 ylabel("Energy")
283
```

```
284 %% Rotational and Horizotnal
285
286 clear all; close all; clc;
287 load('cam1_4.mat')
288 numFrames1 = size(vidFrames1_4,4);
289
290 data1 = [];
291 filter = zeros (480 ,640) ;
292 filter (230:450 , 280:450 ) = 1;
293
294 %x 480, y 640
295 % x and y are swapped
296 for j = 1:numFrames1
297     X = vidFrames1_4(:,:,:,j);
298     G = double(rgb2gray(X));
299     Xf = G .* filter ;
300     th = Xf > 245;
301     indx = find(th) ;
302     [Y ,X] = ind2sub (size(th) , indx) ;
303     data1 = [data1; mean(X), mean(Y)];
304 end
305
306 load('cam2_4.mat')
307 numFrames2 = size(vidFrames2_4,4);
308
309 data2 = [];
310 filter = zeros (480 ,640) ;
311 filter (100:450 , 150:400 ) = 1;
312
313 %x 480, y 640
314 % x and y are swapped
315 for j = 1:numFrames2
316     X = vidFrames2_4(:,:,:,j);
317     G = double(rgb2gray(X));
318     Xf = G .* filter ;
319     th = Xf > 245;
320     indx = find(th) ;
321     [Y ,X] = ind2sub (size(th) , indx) ;
322     data2 = [data2; mean(X), mean(Y)];
323 end
324
```

```
325
326 load('cam3_4.mat')
327 numFrames3 = size(vidFrames3_4,4);
328
329 data3 = [];
330 filter = zeros (480 ,640) ;
331 filter (100:300 , 260:600 ) = 1;
332
333 %x 480, y 640
334 % x and y are swapped
335 for j = 1:numFrames3
336     X = vidFrames3_4(:,:,:,j);
337     G = double(rgb2gray(X));
338     Xf = G .* filter ;
339     th = Xf > 230;
340     indx = find(th) ;
341     [Y ,X] = ind2sub (size(th) , indx) ;
342     data3 = [data3; mean(X), mean(Y)];
343 end
344 %% resizing r/horiz dis
345 [M, I] = min(data1(1:10,2));
346 data1 = data1(I:end,:);
347 [M, I] = min(data2(1:10,2));
348 data2 = data2(I:end,:);
349 [M, I] = min(data3(1:10,2));
350 data3 = data3(I:end,:);
351
352 data2 = data2(1:length(data1),:);
353 data3 = data3(1:length(data1),:);
354
355 %% PCA  horiz
356 dat_arr = [data1'; data2'; data3'];
357 [m,n] = size(dat_arr);
358
359 mn=mean(dat_arr,2);
360 dat_arr = dat_arr - repmat(mn,1,n);
361 [u,s,v]=svd(dat_arr'/sqrt(n-1));
362 lambda=diag(s).^2;
363 Y=dat_arr' * v;
364 var = diag(s);
365 %% hoirz plots
```

```
366 figure()
367
368
369 subplot(2,1,1)
370 plot(1:384, Y(:,1),1:384, Y(:,2),1:384, Y(:,3));
371 title("Horizontal Displacement across Principal
      Component")
372 xlabel("Frames"); ylabel("Motion(pixels)")
373 subplot(2,1,2)
374 plot (1:6 , lambda/sum(lambda) ,'o', 'Linewidth', 2);
375 title("Energy of Diagonal Variacne");
376 xlabel("Diagonal Variance");
377 ylabel("Energy")
```