PCA

Jason Miller
AMATH 482

February 21, 2020

Abstract

* In this assignment we analyze the motion of a physical system using principal component analysis. We start with video footage of the system from different camera angles, and we use this to extract position data. Then, we use principal component analysis to analyze the system to explore the PCA algorithm.

1 Introduction and Overview

We are given video footage of a bucket attatched to a spring, where the top of the spring is held in place and the bucket is moving in four different cases: an ideal case, a noisy case (where there is a lot of camera shake), a case with horizontal displacement, and a case with horizontal displacement and rotation. The footage is from three different camera angles, in one of which the camera is turned 90°. Given this unwieldy footage, we construct position vectors for the bucket using a Point-Tracker, from MATLAB's computer vision toolbox. Then we perform PCA on a matrix containing the x-coordinates of the bucket, and another matrix containing the y-coordinates of the bucket, where we will note the characteristics of the energies of each mode and examine the low-rank approximations of the system.

2 Theoretical Background

The singular value decomposition (SVD) is an expression of the fact that all linear transformations stretch and rotate vectors (or, that applying a linear transformation to a hypersphere creates an ellipsoid). The formula for the decomposition is:

$$A = U\Sigma V \tag{1}$$

Here, U and V are unitary matrices which represent rotations, and Σ is a diagonal matrix which represents a stretch. Principal component analysis is closely related to the SVD; it is equivelent to taking the SVD of a data matrix, or taking the eigendecomposition of the corresponding covariance matrix for the data matrix. It is used to create low-rank approximations of the data. The energy of a mode is the proportion of the information captured by the mode.

3 Algorithm Implementation and Development

We have four different scripts; one for each scenario. First, we load the .mat files which contain our data. Then, we initialize variables which store our position vectors and the number of frames for each camera, and create a PointTracker object that tracks points on the bucket, and initialize them with the first frame of each movie. The we iterate through each frame, creating the position vectors. We track only one point. For some cases, we compare each frame to a stationary object to remove the camera shake, since our PointTracker otherwise would get lost. Next, since the cameras don't begin filming at the same time, we manually find the first minimum of the y-values in the position vectors, and truncate our data vectors so that they all begin at this minimum and have the same length. This completes our data extraction. Now we put these vectors into a data matrix and compute the SVD. We compute the energies at each point and compute the low-rank approximations.

4 Computational Results

Our position data was succesfully extracted from the video footage. We also successfully computed the PCA for our data matrices. Then, we found the energies for the low-rank approximations, and our results were very interesting. The energy of the rank 1 approximation for scenario 1 was .9073, meaning that most of the information for this scenario was captured in the rank 1 approximation, which makes sense because the motion was ideal and the system had very little noise. For scenario 2, the energy was .8220, which means that the sytem was quite noisy so other rank approximations which represent the noise will be more significant. In either case, all of the energies for the rank 1 approximation were high because there was a high degree of redundancy in our data, and we did not need three camera angles to

capture the behavior of the system in any case.

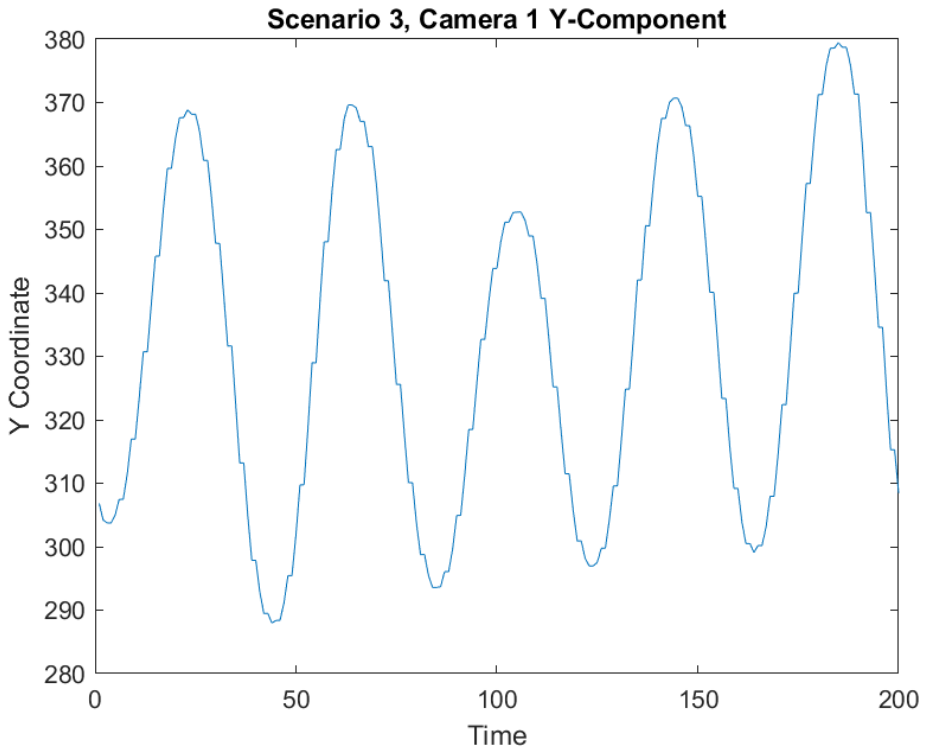

Figure 1: This is an example of the data vectors that we extracted from the movie footage using a PointTracker object.

5 Summary and Conclusions

In this assignment we analyzed four different use cases of PCA, after doing significant work to extract the position data from the movies. The rank 1 approximations for the position data captures most of the information for every system, but the noisier scenarios had lower energies for the rank 1 approximation.

Appendix A: Function Descriptions

- abs: 2-norm
- exp: "e to the"
- fft: Fast Fourier Transform

- fftshift: interchanges first and second halves of the domain
- initialize: initializes PointTracker
- linspace: creates vector with linearly spaced points
- load: loads .mat file
- PointTracker: constructs PointTracker object
- print: saves figure
- release: releases PointTracker
- rgb2gray: changes image to grayscale
- svd: computes the SVD of a matrix
- title: titles plot
- xlabel: labels x-axis on plot
- zeros: creates matrix of zeros

Appendix B: Code

```
clear; clc; close all;

load('cam1_1.mat')
load('cam1_2.mat')
load('cam1_3.mat')
load('cam1_4.mat')
load('cam2_1.mat')
load('cam2_2.mat')
load('cam2_3.mat')
load('cam2_4.mat')
load('cam3_1.mat')
load('cam3_2.mat')
load('cam3_3.mat')
load('cam3_4.mat')

numFrames1 = size(vidFrames1_1, 4);
numFrames2 = size(vidFrames2_1, 4);
numFrames3 = size(vidFrames3_1, 4);

positionTracker = vision.PointTracker;
noiseTracker = vision.PointTracker;

x1 = zeros(1,numFrames1);
```

```
24 y1 = zeros(1,numFrames1);
25 x2 = zeros(1,numFrames2);
26 y2 = zeros(1,numFrames2);
27 x3 = zeros(1,numFrames3);
28 y3 = zeros(1,numFrames3);
29
30 for j = 1:numFrames1
31     gray1(:,:,j) = rgb2gray(vidFrames1_1(:,:,:,j))
           ;
32     if j == 1
33         initialize(positionTracker, [342 292],
               gray1(:,:,1));
34         initialize(noiseTracker, [20 313], gray1
               (:,:,1));
35     end
36     [points, pointValidity] = positionTracker(
           gray1(:,:,j));
37     [perterbation, pertValidity] = noiseTracker(
           gray1(:,:,j));
38     x1(j) = points(1);
39     y1(j) = points(2);
40     %x1(j) = points(1) - (perterbation(1) - 20);
41     %y1(j) = points(2) - (perterbation(2) - 313);
42 end
43
44 for j = 1:numFrames2
45     gray2(:,:,j) = rgb2gray(vidFrames2_1(:,:,:,j))
           ;
46     if j == 1
47         release(positionTracker);
48         release(noiseTracker);
49         initialize(positionTracker, [283 344],
               gray2(:,:,1));
50         initialize(noiseTracker, [58 175], gray2
               (:,:,1));
51     end
52     [points, pointValidity] = positionTracker(
           gray2(:,:,j));
53     [perterbation, pertValidity] = noiseTracker(
           gray2(:,:,j));
54     x2(j) = points(1);
55     y2(j) = points(2);
56     %x2(j) = points(1) - (perterbation(1) - 58);
```

```
57      %y2(j) = points(2) - (perterbation(2) - 175);
58  end
59
60  for j = 1:numFrames3
61      gray3(:,:,j) = rgb2gray(vidFrames3_1(:,:,:,j))
            ;
62      if j == 1
63          release(positionTracker);
64          release(noiseTracker);
65          initialize(positionTracker, [366 288],
                gray3(:,:,1));
66          initialize(noiseTracker, [554 75], gray3
                (:,:,1));
67      end
68      [points, pointValidity] = positionTracker(
            gray3(:,:,j));
69      [perterbation, pertValidity] = noiseTracker(
            gray3(:,:,j));
70      x3(j) = points(1);
71      y3(j) = points(2);
72      %x3(j) = points(1) - (perterbation(1) - 58);
73      %y3(j) = points(2) - (perterbation(2) - 175);
74  end
75  temp = x3;
76  x3 = y3;
77  y3 = temp;
78
79  start1 = 10;
80  x1 = x1(start1 : start1 + 199);
81  y1 = y1(start1 : start1 + 199);
82
83  start2 = 9;
84  x2 = x2(start2 : start2 + 199);
85  y2 = y2(start2 : start2 + 199);
86
87  start3 = 10;
88  x3 = x3(start3 : start3 + 199);
89  y3 = y3(start3 : start3 + 199);
90
91  P = [x1;y1;x2;y2;x3;y3];
92  [U,S,V] = svd(P,'econ');
93  svs = diag(S);
94  e1 = svs(1) / sum(svs)
```

```
clear; clc; close all;

load('cam1_1.mat')
load('cam1_2.mat')
load('cam1_3.mat')
load('cam1_4.mat')
load('cam2_1.mat')
load('cam2_2.mat')
load('cam2_3.mat')
load('cam2_4.mat')
load('cam3_1.mat')
load('cam3_2.mat')
load('cam3_3.mat')
load('cam3_4.mat')

numFrames1 = size(vidFrames1_2, 4);
numFrames2 = size(vidFrames2_2, 4);
numFrames3 = size(vidFrames3_2, 4);

positionTracker = vision.PointTracker;
noiseTracker = vision.PointTracker;

x1 = zeros(1,numFrames1);
y1 = zeros(1,numFrames1);
x2 = zeros(1,numFrames2);
y2 = zeros(1,numFrames2);
x3 = zeros(1,numFrames3);
y3 = zeros(1,numFrames3);

%Initializing these breaks the code for some
    reason
%gray1 = zeros(480,640,numFrames1);
%gray2 = zeros(480,640,numFrames2);
%gray3 = zeros(480,640,numFrames3);

for j = 1:numFrames1
    gray1(:,:,j) = rgb2gray(vidFrames1_2(:,:,:,j))
        ;
    if j == 1
        initialize(positionTracker, [323 358],
            gray1(:,:,1));
        initialize(noiseTracker, [44 71], gray1
            (:,:,1));
```

```
40      end
41      [points, pointValidity] = positionTracker(
            gray1(:,:,j));
42      [perterbation, pertValidity] = noiseTracker(
            gray1(:,:,j));
43      x1(j) = points(1);
44      y1(j) = points(2);
45      %x1(j) = points(1) - (perterbation(1) - 20);
46      %y1(j) = points(2) - (perterbation(2) - 313);
47  end
48
49  for j = 1:numFrames2
50      gray2(:,:,j) = rgb2gray(vidFrames2_2(:,:,:,j))
            ;
51      if j == 1
52          release(positionTracker);
53          release(noiseTracker);
54          initialize(positionTracker, [312 368],
                gray2(:,:,1));
55          initialize(noiseTracker, [58 99], gray2
                (:,:,1));
56      end
57      [points, pointValidity] = positionTracker(
            gray2(:,:,j));
58      [perterbation, pertValidity] = noiseTracker(
            gray2(:,:,j));
59      %x2(j) = points(1);
60      %y2(j) = points(2);
61      x2(j) = points(1) - (perterbation(1) - 58);
62      y2(j) = points(2) - (perterbation(2) - 175);
63  end
64
65  for j = 1:numFrames3
66      gray3(:,:,j) = rgb2gray(vidFrames3_2(:,:,:,j))
            ;
67      if j == 1
68          release(positionTracker);
69          release(noiseTracker);
70          initialize(positionTracker, [383 271],
                gray3(:,:,1));
71          initialize(noiseTracker, [20 20], gray3
                (:,:,1));
72      end
```

```
73      [points, pointValidity] = positionTracker(
            gray3(:,:,j));
74      [perterbation, pertValidity] = noiseTracker(
            gray3(:,:,j));
75      x3(j) = points(1);
76      y3(j) = points(2);
77      %x3(j) = points(1) - (perterbation(1) - 58);
78      %y3(j) = points(2) - (perterbation(2) - 175);
79  end
80  temp = x3;
81  x3 = y3;
82  y3 = temp;
83
84  start1 = 4;
85  x1 = x1(start1 : start1 + 199);
86  y1 = y1(start1 : start1 + 199);
87
88  start2 = 8;
89  x2 = x2(start2 : start2 + 199);
90  y2 = y2(start2 : start2 + 199);
91
92  start3 = 8;
93  x3 = x3(start3 : start3 + 199);
94  y3 = y3(start3 : start3 + 199);
95
96  P = [x1;y1;x2;y2;x3;y3];
97  [U,S,V] = svd(P,'econ');
98  svs = diag(S);
99  e1 = svs(1) / sum(svs)
```

```
1  clear; 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 numFrames1 = size(vidFrames1_3, 4);
17 numFrames2 = size(vidFrames2_3, 4);
18 numFrames3 = size(vidFrames3_3, 4);
19
20 positionTracker = vision.PointTracker;
21 noiseTracker = vision.PointTracker;
22
23 x1 = zeros(1,numFrames1);
24 y1 = zeros(1,numFrames1);
25 x2 = zeros(1,numFrames2);
26 y2 = zeros(1,numFrames2);
27 x3 = zeros(1,numFrames3);
28 y3 = zeros(1,numFrames3);
29
30 %Initializing these breaks the code for some
       reason
31 %gray1 = zeros(480,640,numFrames1);
32 %gray2 = zeros(480,640,numFrames2);
33 %gray3 = zeros(480,640,numFrames3);
34
35 for j = 1:numFrames1
36     gray1(:,:,j) = rgb2gray(vidFrames1_3(:,:,:,j))
           ;
37     if j == 1
38         initialize(positionTracker, [340 340],
               gray1(:,:,1));
39         initialize(noiseTracker, [20 20], gray1
               (:,:,1));
40     end
41     [points, pointValidity] = positionTracker(
           gray1(:,:,j));
42     [perterbation, pertValidity] = noiseTracker(
           gray1(:,:,j));
43     x1(j) = points(1);
44     y1(j) = points(2);
45     %x1(j) = points(1) - (perterbation(1) - 20);
46     %y1(j) = points(2) - (perterbation(2) - 313);
47 end
48
49 for j = 1:numFrames2
50     gray2(:,:,j) = rgb2gray(vidFrames2_3(:,:,:,j))
           ;
```

```
51      if j == 1
52          release(positionTracker);
53          release(noiseTracker);
54          initialize(positionTracker, [254 352],
                gray2(:,:,1));
55          initialize(noiseTracker, [20 20], gray2
                (:,:,1));
56      end
57      [points, pointValidity] = positionTracker(
            gray2(:,:,j));
58      [perterbation, pertValidity] = noiseTracker(
            gray2(:,:,j));
59      x2(j) = points(1);
60      y2(j) = points(2);
61      %x2(j) = points(1) - (perterbation(1) - 58);
62      %y2(j) = points(2) - (perterbation(2) - 175);
63  end
64
65  for j = 1:numFrames3
66      gray3(:,:,j) = rgb2gray(vidFrames3_3(:,:,:,j))
            ;
67      if j == 1
68          release(positionTracker);
69          release(noiseTracker);
70          initialize(positionTracker, [405 242],
                gray3(:,:,1));
71          initialize(noiseTracker, [20 20], gray3
                (:,:,1));
72      end
73      [points, pointValidity] = positionTracker(
            gray3(:,:,j));
74      [perterbation, pertValidity] = noiseTracker(
            gray3(:,:,j));
75      x3(j) = points(1);
76      y3(j) = points(2);
77      %x3(j) = points(1) - (perterbation(1) - 58);
78      %y3(j) = points(2) - (perterbation(2) - 175);
79  end
80  temp = x3;
81  x3 = y3;
82  y3 = temp;
83
84  start1 = 10;
```

```
x1 = x1(start1 : start1 + 199);
y1 = y1(start1 : start1 + 199);

start2 = 9;
x2 = x2(start2 : start2 + 199);
y2 = y2(start2 : start2 + 199);

start3 = 10;
x3 = x3(start3 : start3 + 199);
y3 = y3(start3 : start3 + 199);

P = [x1;y1;x2;y2;x3;y3];
[U,S,V] = svd(P,'econ');
svs = diag(S);
e1 = svs(1) / sum(svs)
```

```
clear; clc; close all;

load('cam1_1.mat')
load('cam1_2.mat')
load('cam1_3.mat')
load('cam1_4.mat')
load('cam2_1.mat')
load('cam2_2.mat')
load('cam2_3.mat')
load('cam2_4.mat')
load('cam3_1.mat')
load('cam3_2.mat')
load('cam3_3.mat')
load('cam3_4.mat')

numFrames1 = size(vidFrames1_4, 4);
numFrames2 = size(vidFrames2_4, 4);
numFrames3 = size(vidFrames3_4, 4);

positionTracker = vision.PointTracker;
noiseTracker = vision.PointTracker;

x1 = zeros(1,numFrames1);
y1 = zeros(1,numFrames1);
x2 = zeros(1,numFrames2);
y2 = zeros(1,numFrames2);
x3 = zeros(1,numFrames3);
y3 = zeros(1,numFrames3);
```

```
29
30 %Initializing these breaks the code for some
       reason
31 %gray1 = zeros(480,640,numFrames1);
32 %gray2 = zeros(480,640,numFrames2);
33 %gray3 = zeros(480,640,numFrames3);
34
35 for j = 1:numFrames1
36     gray1(:,:,j) = rgb2gray(vidFrames1_4(:,:,:,j))
           ;
37     if j == 1
38         initialize(positionTracker, [381 317],
               gray1(:,:,1));
39         initialize(noiseTracker, [20 20], gray1
               (:,:,1));
40     end
41     [points, pointValidity] = positionTracker(
           gray1(:,:,j));
42     [perterbation, pertValidity] = noiseTracker(
           gray1(:,:,j));
43     x1(j) = points(1);
44     y1(j) = points(2);
45     %x1(j) = points(1) - (perterbation(1) - 20);
46     %y1(j) = points(2) - (perterbation(2) - 313);
47 end
48
49 for j = 1:numFrames2
50     gray2(:,:,j) = rgb2gray(vidFrames2_4(:,:,:,j))
           ;
51     if j == 1
52         release(positionTracker);
53         release(noiseTracker);
54         initialize(positionTracker, [256 289],
               gray2(:,:,1));
55         initialize(noiseTracker, [20 20], gray2
               (:,:,1));
56     end
57     [points, pointValidity] = positionTracker(
           gray2(:,:,j));
58     [perterbation, pertValidity] = noiseTracker(
           gray2(:,:,j));
59     x2(j) = points(1);
60     y2(j) = points(2);
```

```
61      %x2(j) = points(1) - (perterbation(1) - 58);
62      %y2(j) = points(2) - (perterbation(2) - 175);
63  end
64
65  for j = 1:numFrames3
66      gray3(:,:,j) = rgb2gray(vidFrames3_4(:,:,:,j))
            ;
67      if j == 1
68          release(positionTracker);
69          release(noiseTracker);
70          initialize(positionTracker, [400 210],
                gray3(:,:,1));
71          initialize(noiseTracker, [20 20], gray3
                (:,:,1));
72      end
73      [points, pointValidity] = positionTracker(
            gray3(:,:,j));
74      [perterbation, pertValidity] = noiseTracker(
            gray3(:,:,j));
75      x3(j) = points(1);
76      y3(j) = points(2);
77      %x3(j) = points(1) - (perterbation(1) - 58);
78      %y3(j) = points(2) - (perterbation(2) - 175);
79  end
80  temp = x3;
81  x3 = y3;
82  y3 = temp;
83
84  start1 = 10;
85  x1 = x1(start1 : start1 + 199);
86  y1 = y1(start1 : start1 + 199);
87
88  start2 = 9;
89  x2 = x2(start2 : start2 + 199);
90  y2 = y2(start2 : start2 + 199);
91
92  start3 = 10;
93  x3 = x3(start3 : start3 + 199);
94  y3 = y3(start3 : start3 + 199);
95
96  P = [x1;y1;x2;y2;x3;y3];
97  [U,S,V] = svd(P,'econ');
98  svs = diag(S);
```

```
 99 e1 = svs(1) / sum(svs)
100
101 plot(y1);
102 ylabel('Y Coordinate');
103 xlabel('Time');
104 title('Scenario 3, Camera 1 Y-Component');
105 print(gcf,'-dpng','3y.png');
```