Principal Component Analysis

Yinxi Pan

AMATH482, Homework3, Winter2020

February 22, 2020

Abstract

The report analysis 12 movies that film the movement of a bucket with a flashlight on top of it. The main method used is Principal Component Analysis(PCA). The location of the bucket is found by converting the film into grayscale and then filtering with the Shannon window. Singular Value Decomposition has been applied to the position data matrix. Various aspects of PCA are illustrated and comparisons between different cases are presented.

1 Introduction and Overview

In this report, we show what Principle Component Analysis can tell us and its piratical usefulness. Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components[1]. The main method used for PCA is Singular Value Decomposition(SVD) which is discussed in the later section.

The later sections show the analysis of 12 videos of 4 cases based on PCA. They all show the oscillations of a bucket with a flashlight on the top. The first case is ideal where there's little noise with harmonic oscillations. The second case is the noisy case where the camera is shaking. The third case adds a pendulum motion beside the harmonic oscillations. The last case has a pendulum motion and rotation besides the harmonic oscillations. Each case contains 3 films of the same object under different cameras. The first two cameras are vertically displayed and the last one is horizontally displaced.

2 Theoretical Background

1. **Singular Value Decomposition (SVD)**[2]

[1] https://en.wikipedia.org/wiki/Principal$_{c}omponent_{a}nalysis$

[2] Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

The Singular Value Decomposition is a transformation that stretches/compresses and rotates a given set of vectors. Mathematically, the full SVD can be shown as in equation(1)

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

where $U \in \mathbb{C}^{m\times m}$ is unitary, $V \in \mathbb{C}^{n\times n}$ is unitary and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal. Also, V^* is the conjugate transpose of A. Additionally, it is assumed that the diagonal entries of Λ are nonnegative and ordered from the largest to the smallest such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ where $p = min(m, n)$.Then we also have the relationship between A and A^T as in equation(2)

$$AA^T = U\Sigma V^*(U\Sigma V^*)^T = U\Sigma V^* V\Sigma U^* = U\Sigma^2 U^* \tag{2}$$

2. **Eigen-decomposition**[3]

   The Eigen-decomposition for any square matrix A is defined to be as in equation(3)

$$A = S\Lambda S^{-1} \tag{3}$$

   where S is the square matrix which the columns are the eigenvectors of A and Λ is a square diagonal matrix with the corresponding eigenvalue in S. Then we also have the relationship between A and A^T as in equation(4)

$$AA^T = S\Lambda S^{-1}(S\Lambda S^{-1})^T = S\Lambda^2 S^{-1} \tag{4}$$

3. **Covariance Matrix**[4]

   An easy way to identify redundant data is by considering the covariance between data sets. The covariance for any matrix $X \in \mathbb{R}^{n\times n}$ is define as in equation(5)

$$C_X = \frac{1}{n-1} XX^T \tag{5}$$

4. **Principal Component Analysis (PCA)** [5]

   Principal Component Analysis is an application of SVD. PCA could help to reduce noise and redundancy in large dataset. Based on the SVD and Eigen-decomposition discussed before, we could write the covariance matrix for data matrix X as in equation(6)

$$C_X = \frac{1}{n-1} XX^T = \frac{1}{n-1}\Lambda = \frac{1}{n-1}\Sigma^2 \tag{6}$$

   where Λ is the diagonal matrix that contains the eigenvalues and Σ is the diagonal matrix

[3]Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

[4]Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

[5]Conceptual definitions are all from: Kutz, J. Nathan. Data-Driven Modeling Scientific Computation: Methods for Complex Systems Big Data, Chap13.4-5 OUP Oxford, 2013.

from SVD. The principle components are on the diagonal and they tell us which information are important in the matrix X. Equation(6) also implies that $\Sigma^2 = \Lambda$.

3 Algorithm Implementation and Development

There are two major steps in the algorithm and the algorithm applies to all cases[6].

1. Create the data matrix

   (a) Load the data, and save them into a 4 dimensional matrix.Find the number of frames so that we could convert the film into actual data. For each frame, we transform the frame into gray scale and covert the data type into double.

   (b) To smooth the data, we apply the Shannon filter.

   (c) After applying the filter, we find the values that are larger than 240 which indicates the pink or brighter color. Then we use ind2sub() to convert the index into x,y coordinates. To avoid noise, we average the coordinates and form the x,y coordinate vectors.Note, for different videos, we need to manually adjust the Shannon filter such that it captures the appropriate area.

   (d) Made all coordinate vectors have the same size. Since each film has different frame size, to combine them into the a single data matrix, we need to manually adjust the size. To achieve that, we find the minimal coordinate value which should be the starting point of the bucket and then take all the values after the minimal value. To make all three films have equal length under the same case, we take the minimal length of all the coordinate vectors and truncate the other vectors based on the minimal length.

   (e) Form the data matrix that contains the x,y coordinates under different cameras. We need to subtract the mean value to make sure all cameras have the same scale.

2. Run PCA analysis

   Perform Singular Value Decomposition on our data matrix and plot the results. (We get the principal components from the SVD)

4 Computational Results

1. Test1: Ideal case, illustrated in Figure(1) and Figure(2)

   Figure(1) illustrates the displacement in x,y-direction. As we could see the three cameras are smooth and aligned pretty well. To verify if PCA does a good job, we also plot the first principal components projection and it matches well with the original displacement.

[6]Matlab code check Appendix.B

Figure(2) shows the energy captured by the principal component. As we could see the first component captures approximately 90% energy. This makes sense because we are under the idea case. Only one direction provides important information since the bucket is moving along one direction.

2. Test2: Noisy case, illustrated in Figure(3) and Figure(4)

   Figure(3) illustrates the displacement in x,y-direction. As we could see, the displacements in different cameras are not aligned well but still showing an oscillating pattern. The reason could be we apply the Shannon filter. Since Shannon is a step function, it might cut off some important information if we do not select a perfect area around the bucket. I think the Gaussian filter might perform a better job but I'm not sure how to develop the filter under films. To verify if PCA does a good job, we also plot the first principal components projection and it does not match well with the original displacement because of the noise.

   Figure(4) shows the energy captured by the principal component. As we could see the first two components capture approximately 90% energy. This makes sense because we are in a noisy case. Although the bucket is moving in y-direction we shake the camera and 'add' the displacement in the x-direction.

3. Test3: Horizontal displacement, illustrated in Figure(5) and Figure(6)

   Figure(5) illustrates the displacement in x,y-direction. As we could see the three cameras are smooth and aligned pretty well except the displacement in the x-direction. The reason is under this test, we have the pendulum motion involved. The y-direction aligns well since we don't have noise under this case, but the x-direction performs badly since we choose the minimal y-direction to truncate the data. This might ends with different x coordinates. To verify if PCA does a good job, we also plot the first principal components projection and it matches well with the original displacement expect the projection seems to be the inverted original displacement.

   Figure(6) shows the energy captured by the principal component. As we could see the first two components capture approximately 90% energy. This performs better than the second test since we don't have noise anymore. But the pendulum motion is not redundant data. That's why we need 2 principal components.

4. Test4: Horizontal displacement and rotation, illustrated in Figure(7) and Figure(8)

   Figure(7) illustrates the displacement in x,y-direction. As we could see the three cameras are smooth and aligned pretty well except there's a difference between their magnitude. I'm not sure what causes this difference since ideally, they should be the same. To verify if PCA does a good job, we also plot the first principal components projection and it matches well with the original displacement.

   Figure(8) shows the energy captured by the principal component. As we could see the first two components capture approximately 90% energy. The reason we need two-component is that except the displacement in the y-direction, we still need the rotation around the y-direction, i.e. the displacement in the x. direction.

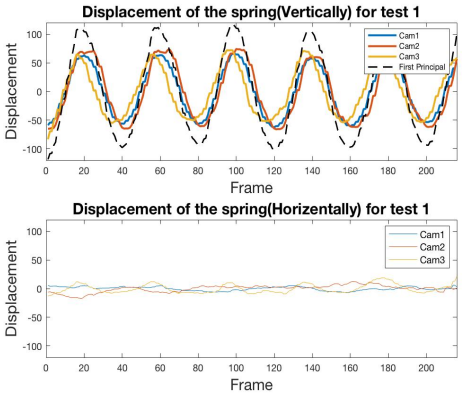

Figure 1: The displacement in x,y direction and also with the first principal component projection. - Test1

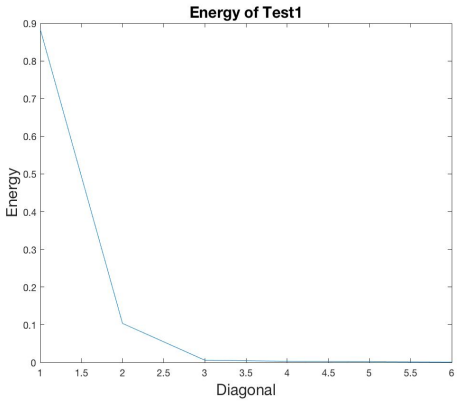

Figure 2: Principal Components - Test1

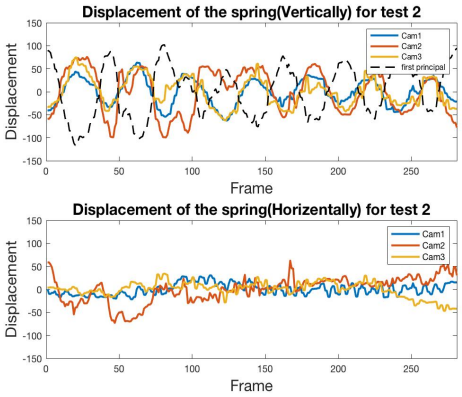

Figure 3: The displacement in x,y direction and also with the first principal component projection. - Test2

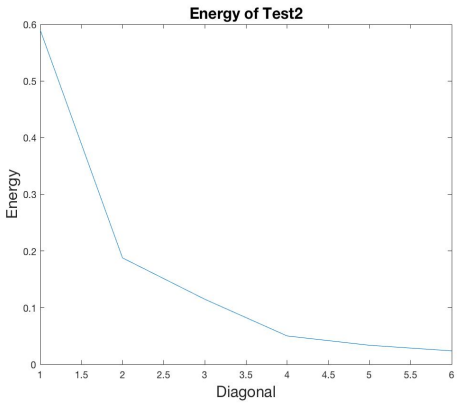

Figure 4: Principal Components - Test2

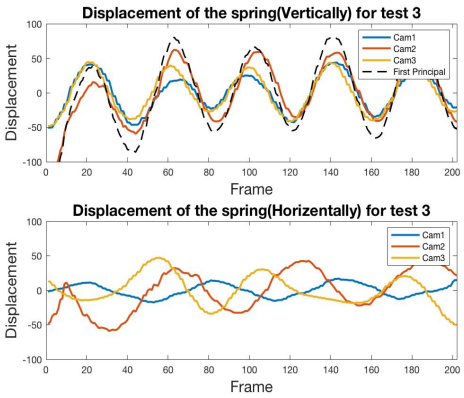

Figure 5: The displacement in x,y direction and also with the first principal component projection. - Test3

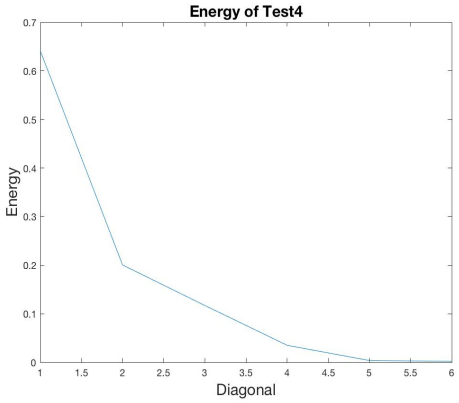

Figure 6: Principal Components - Test3

5 Summary and Conclusions

In this homework, we apply the Principle Component Analysis on 4 videos filmed by 3 different cameras. Although we are not able to get a well-smoothed data, PCA still performs pretty well at capturing the important information. For the first ideal case, since it's only moving in the y-direction, PCA shows one principal component matters the most. For the rest cases, except for the displacement in the y-direction, we have the rotation and pendulum motion that matters in the film. Accordingly, PCA shows two of the principal component captures the most important information. We also plot the first principal components projection. Since the value of the first principal components projection is all greater than 50%, we could see in Figure(1/3/5/7) that the first principal components projection captures most information in the original displacement.

Appendix A.MATLAB functions used[7]

1. implay(): opens the Video Viewer app.
2. frame2im():returns the truecolor (RGB) image from the single movie frame.
3. rgb2gray():converts the truecolor image RGB to the grayscale image I.
4. ind2sub(): 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.
5. repmat(): returns an array containing n copies of A in the row and column dimensions.
6. svd(): performs a singular value decomposition of matrix A, such that A = U*S*V'.

[7]Other functions applied in this report can be found in the first report. The full description of each function below can be foun on : "MATLAB." MATLAB Documentation, www.mathworks.com/help/matlab/.

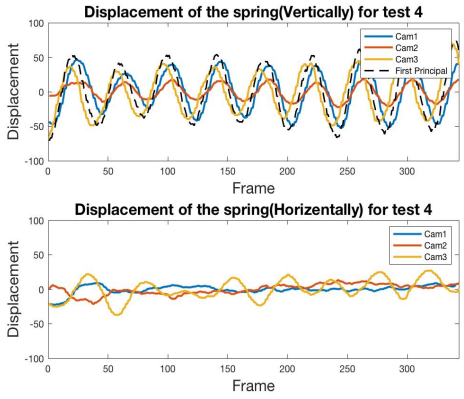

Figure 7: The displacement in x,y direction and also with the first principal component projection. - Test4

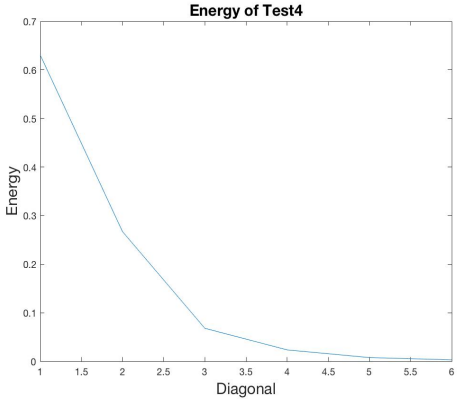

Figure 8: Principal Components - Test4

Appendix B. MATLAB codes

All cases share the same method, but you need to change the width and range of the filer and load different files.

```
1
2  % first video
3  % load data
4  clear; close all; clc;
5
6  load('cam1_1.mat')
7  load('cam2_1.mat')
8  load('cam3_1.mat')
9
10 % play the videos
11 % implay(vidFrames1_1)
12 % implay(vidFrames2_1)
13 % implay(vidFrames3_1)
14
15 v11 = vidFrames1_1;
16 v21 = vidFrames2_1;
17 v31 = vidFrames3_1;
18
19 numFrames11 = size(v11,4);
20 numFrames21 = size(v21,4);
21 numFrames31 = size(v31,4);
22
23
24 for i = 1:numFrames11
25    mov11(i).cdata = v11(:,:,:,i);
26    mov11(i).colormap = [];
27 end
28
29 for i = 1:numFrames21
30    mov21(i).cdata = v21(:,:,:,i);
31    mov21(i).colormap = [];
32 end
33
34 for i = 1:numFrames31
35    mov31(i).cdata = v31(:,:,:,i);
36    mov31(i).colormap = [];
37 end
38
39
40 X1 = [];
41 Y1 = [];
42
43 X2 = [];
44 Y2 = [];
45
46 X3 = [];
47 Y3 = [];
48
49 %%
50 width = 50;
```

```
51
52 sh = zeros(480,640);
53 sh(300 - 2.6*width:1:300+2.6*width, 350-width:1:350+width) = 1;
54
55 for i = 1:numFrames11
56     data = frame2im(mov11(i));
57     data = rgb2gray(data);
58
59     data2 = double(data);
60     dataf = sh .* data2;
61
62     pink = dataf > 240;
63     Index = find(pink);
64     [y, x] = ind2sub(size(pink), Index);
65     X1 = [X1, mean(x)];
66     Y1 = [Y1, mean(y)];
67 end
68
69 sh = zeros(480,640);
70 sh(250 - 1.7*width:1:250+1.7*width, 290-1.3*width:1:290+1.3*width) = 1;
71
72 for i = 1:numFrames21
73     data = frame2im(mov21(i));
74     data = rgb2gray(data);
75
76     data2 = double(data);
77     dataf = sh .* data2;
78
79     pink = dataf > 240;
80     Index = find(pink);
81     [y, x] = ind2sub(size(pink), Index);
82     X2 = [X2, mean(x)];
83     Y2 = [Y2, mean(y)];
84 end
85
86 sh = zeros(480,640);
87 sh(250 - width:1:250+2*width, 360-2.5*width:1:360+2.5*width) = 1;
88
89 for i = 1:numFrames31
90     data = frame2im(mov31(i));
91     data = rgb2gray(data);
92
93     data2 = double(data);
94     dataf = sh .* data2;
95
96     pink = dataf > 240;
97     Index = find(pink);
98     [y, x] = ind2sub(size(pink), Index);
99     X3 = [X3, mean(x)];
100    Y3 = [Y3, mean(y)];
101 end
102
103 %%
104 [Min, I] = min(Y1(1:50));
105 X1 = X1(I:end);
106 Y1 = Y1(I:end);
107
108 [Min, I] = min(Y2(1:50));
```

```
X2 = X2(I:end);
Y2 = Y2(I:end);


[Min, I] = min(X3(1:50));
X3 = X3(I:end);
Y3 = Y3(I:end);

length = [length(X1), length(X2), length(X3)];

minimal = min(length);

X1 = X1(1:minimal);
X2 = X2(1:minimal);
X3 = X3(1:minimal);
Y1 = Y1(1:minimal);
Y2 = Y2(1:minimal);
Y3 = Y3(1:minimal);



X = [X1; Y1; X2; Y2; X3; Y3];

[m, n] = size(X);
mn = mean(X, 2);

X = X - repmat(mn,1,n);


%%
[u,s,v] = svd(((X./sqrt(n-1))));

lambda = diag(s).^2;

Y = u' * X; % principal components projection

s = diag(s);

figure(1)
subplot(2,1,1)
plot(1:216,X(2,:), 1:216,X(4,:), 1:216,X(5,:),'LineWidth',2)
hold on
plot(1:216, Y(1,:), '--k','LineWidth',1.5)
axis([0 216 -120 120 ])
title('Displacement of the spring(Vertically) for test 1','Fontsize',16)
ylabel('Displacement','Fontsize',16)
xlabel('Frame','Fontsize',16)
legend('Cam1', 'Cam2', 'Cam3','First Principal','Fontsize',5)


subplot(2,1,2)
plot(1:216,X(1,:), 1:216,X(3,:), 1:216,X(6,:))
axis([0 216 -120 120 ])
title('Displacement of the spring(Horizentally) for test 1','Fontsize',16)
ylabel('Displacement','Fontsize',16)
xlabel('Frame','Fontsize',16)
legend('Cam1', 'Cam2', 'Cam3','Fontsize',10)

```

```
figure(2)
plot(1:6,lambda/sum(lambda))
title('Energy of Test1','Fontsize',16)
ylabel('Energy','Fontsize',16)
xlabel('Diagonal','Fontsize',16)
```