A PCA Problem

Chenghao Chen
AMATH 482 HW3

Abstract

In the homework, we have three movie files that are taken in three different angles. In each of the movie files, it has four different scenarios, the ideal case, the noisy case, the horizontal displacement, and the horizontal displacement and rotation case. We want to find out the various aspect of the PCA (Principal Component Analysis) and its practical usefulness. Then, we want to compare different energy level and plot the principal components.

Introduction

As we know, principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observation of possibly correlated variables into a set of values of linearly uncorrelated variables. In this project, we want to use PCA to analyze data in four different scenarios, the ideal case, the noisy case, the horizontal displacement case, and the horizontal displacement with rotation case.

Theoretical Background

Principal Component Analysis

Principal Component Analysis (PCA) highlights one of the key applications of the SVD. It can be used to the data analysis of an unknown, but potentially low-dimensional system. We first need to consider the covariance matrix

$$Cx = \frac{1}{n-1} XX^T$$

where the matrix X contains the experimental data of the system. In particular, X ∈ Cm×n where m are the number of probes or measuring positions, and n is the number of experimental data points taken at each location.

The diagonal terms of CX are the variances for particular measurements. By assumption, large variances correspond to dynamics of interest, whereas low variances are assumed to correspond to non-interesting dynamics. The off-diagonal terms of CX are the covariance between measurements. Indeed, the off-diagonals capture the correlations between all possible pairs of measurements. A large off-diagonal term represents two events that have a high-degree of redundancy, whereas a small off-diagonal coefficient means there is little redundancy in the data.

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. The SVD, much as illustrated in the preceding paragraph, is es- sentially a transformation that stretches/compresses and rotates a given set of vectors.

Diagonalization: The concept of diagonalization is critical for understanding the underpinnings of many physical systems. In this process of diagonalization, the correct co-ordinates, or basis functions, are revealed that reduce the given system to its low-dimensional essence.

Algorithm Implementation

Since we want to use PCA in four different cases with a total of twelve videos, I create two functions in order to reduce the redundancy of my code. The first function is to track the motion of the can. The second function is for PCA.

The first function: `function [x,y] = trackmotion(n,m,a,b,c,d)`. The parameter n is the size of video frames, which is given by `size(vidFrames1_1,4)`. The parameter m is each frame of the videos. The parameters, a, b, c, and d, are time intervals that I want to observe. First, I create a for loop from 1 to the number of frames. Then in the for loop, I use "rgb2gray" to convert the frame to grayscale. After this command, we want to delete time intervals that do not contain the can. By setting those areas to zero, we make them all black. Now, we reduce the size and remove the distraction. Then, we want to find the lightest place, which is the maximum value of the area. We only want areas that are greater or equal than 11/12 of the maximum value and store the data in x and y. By using this function, we can successfully track the can.

The second function `function PCAplot(x1,x2,x3,y1,y2,y3,a,b)`.The parameters ,x1, x2, x3, y1, y2, y3, are vectors that represent motion of the can in three different videos. The parameters, a and b, are used for plotting different figures. First, we want to find the smallest size in three videos and store these vectors in another matrix A row by row. Then, I want to center all the values at zero by subtracting the mean. Then, we put matrix A in SVD form. I plot the diagonal of s matrix over the sum of diagonal, which is the energy level of each mode. At last, I plot v*s, which is the location of principle component multiplied by corresponding energy. By using this function, we can visualize the dominant components and its movement.

In the code, I first load twelve videos and find all the size of videoframes. Then, I store the motion of can in twelve videos by calling the trackmotion function. After I get all the motions, I delete some indexes because I want to make the size of all the motion vectors same. Then, I visualize the dominant components and its movement by calling PCAplot function. Since the third video is rotated, I flip its x and y in plotting.

Computational Result

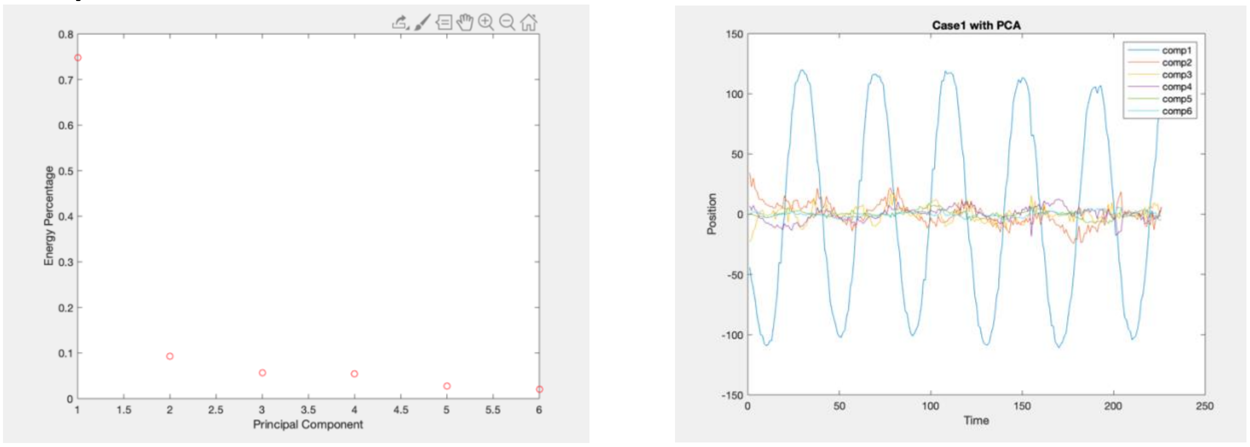

Figure 1: Energy percentage and dynamic captured with PCA in Ideal case

Ideal case:
As we can see, we have only one dominant component in ideal case, which contains about 80% of the total energy. Also, we can clearly see that component 1 is the dominant one in motion from dynamic captured with PCA. Since it is a simple harmonic oscillation, it is true that it only has one dominant component.

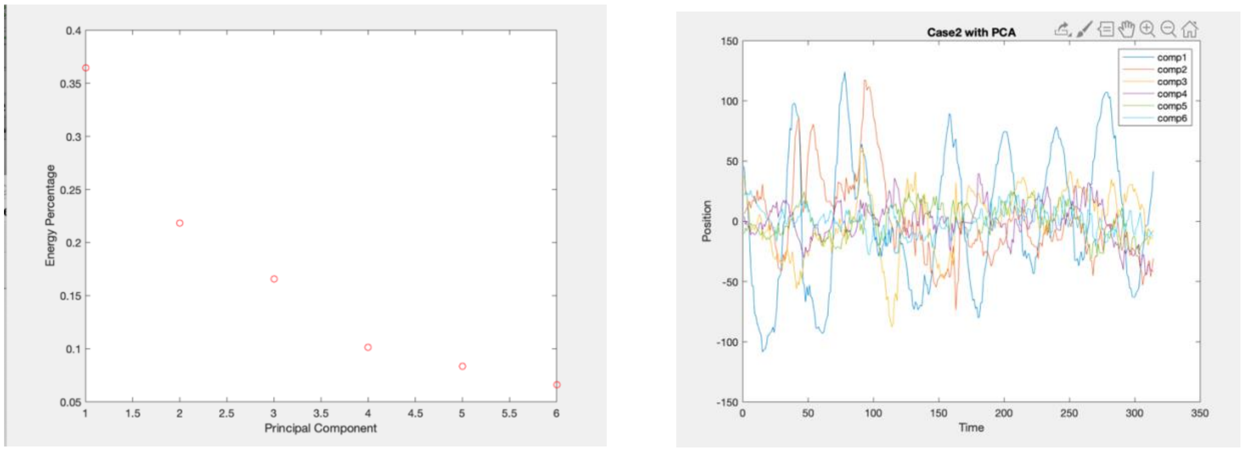

Figure 2: Energy percentage and dynamic captured with PCA in Noisy case

Noisy case:
Ad we can see, we still have one dominant component, which contains about 40% of the total energy. The energy percent reduction is due to the fact that the camera keeps shaking, which creates a lot of noise in the video. Also, we can see that component 2 and 3 also produce movements, even though it is not as significant as the component 1. Therefore, we can conclude that the movement is a oscillation, but not harmonic and smooth.

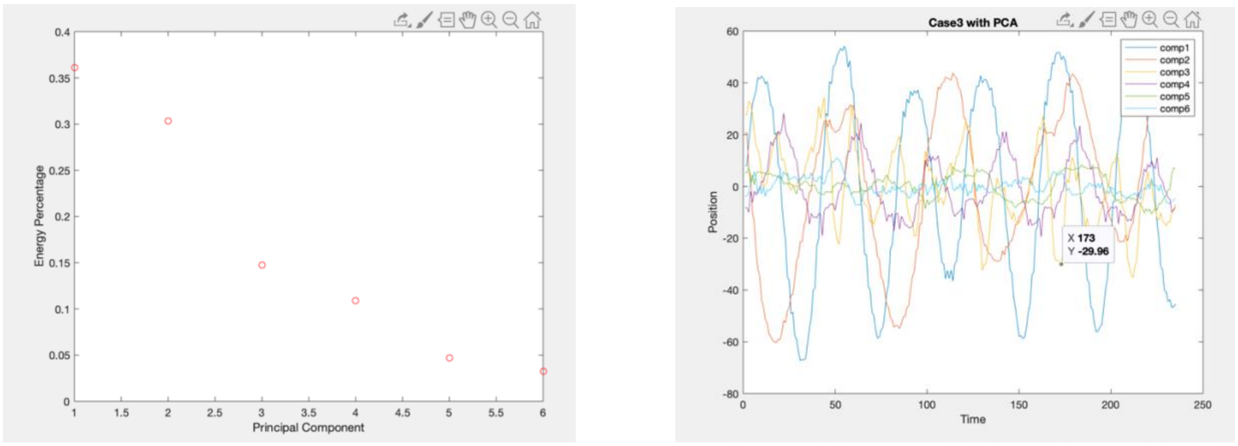

Figure 3: Energy percentage and dynamic captured with PCA in Horizontal displacement case

Horizontal Displacement:
As we can see, we have two dominant components. One is about 37% and another one is about 30%. Since it is a simple harmonic oscillation and a pendulum motion, it is true that it has two dominant components. Also, we can see that component 1 and 2 produce movements, which is much more significant than mother components.

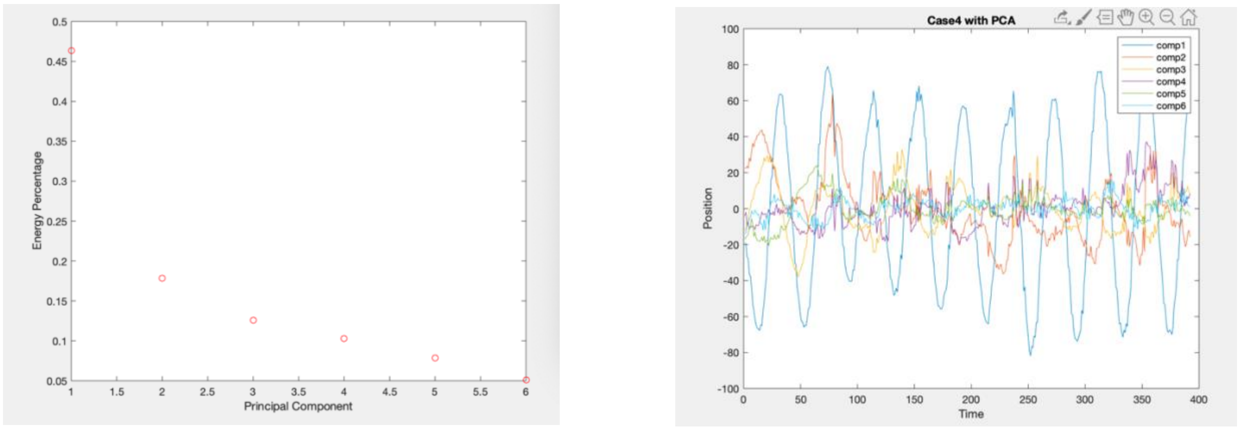

Figure 4: Energy percentage and dynamic captured with PCA in Horizontal displacement and rotation case

Horizontal and rotation:
As we can see, we only have one dominant component, which is about 46%. The energy percent is similar to the noisy case, which is doing oscillation, but not harmonic and smooth, but we have a rotation and horizontal displacement movement. Therefore, PCA does not give us good captured in this case.

Conclusion

After I finish this project, I understand that PCA can effectively find the dominant components and also track the movements of each component. It can also remove the redundancy and decrease distraction. In the four cases, PCA works best in the ideal case, which is only harmonic oscillation. PCA also works well in the horizontal displacement, which are harmonic oscillation and pendulum movement. PCA does not work well in noisy and rotation cases. We can conclude that PCA works well in cases without noisy and rotation.

Citation

[1] J. N. Kutz, Data-driven modeling & scientific computation: methods for complex systems & big data. Oxford: Oxford University Press, 2013.
[2] "Makers of MATLAB and Simulink," MathWorks. [Online]. Available: https://www.mathworks.com/. [Accessed: 25-Jan-2020].

Appendix A

max(N) : This MATLAB function returns the maximum elements of an array. [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]

k = find(X) returns a vector containing the linear indices of each nonzero element in array X.[2]

mean(): This MATLAB function returns the **mean** of the elements of A along the first array dimension whose size does not equal 1. [2]

B = repmat(A,r1,...,rN) specifies a list of scalars, r1,..,rN, that describes how copies of A are arranged in each dimension. When A has N dimensions, the size of B is size(A).*[r1...rN]. [2]

s = svd(A) returns the singular values of matrix A in descending order. [2]

Appendix B

```
clear all; close all; clc;
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')

n11=size(vidFrames1_1,4);
n21=size(vidFrames2_1,4);
n31=size(vidFrames3_1,4);
n12=size(vidFrames1_2,4);
n22=size(vidFrames2_2,4);
n32=size(vidFrames3_2,4);
n13=size(vidFrames1_3,4);
n23=size(vidFrames2_3,4);
n33=size(vidFrames3_3,4);
n14=size(vidFrames1_4,4);
n24=size(vidFrames2_4,4);
n34=size(vidFrames3_4,4);

[x11,y11] = trackmotion(n11,vidFrames1_1,300,400,200,400);
[x21,y21] = trackmotion(n21,vidFrames2_1,200,350,100,350);
[x31,y31] = trackmotion(n31,vidFrames3_1,200,500,200,350);

[x12,y12] = trackmotion(n12,vidFrames1_2,300,400,200,400);
[x22,y22] = trackmotion(n22,vidFrames2_2,200,400,100,350);
[x32,y32] = trackmotion(n32,vidFrames3_2,250,500,200,320);

[x13,y13] = trackmotion(n13,vidFrames1_3,300,400,200,400);
[x23,y23] = trackmotion(n23,vidFrames2_3,200,450,150,350);
[x33,y33] = trackmotion(n33,vidFrames3_3,200,500,200,350);

[x14,y14] = trackmotion(n14,vidFrames1_4,300,400,200,400);
[x24,y24] = trackmotion(n24,vidFrames2_4,200,400,100,350);
[x34,y34] = trackmotion(n34,vidFrames3_4,250,500,150,300);



x21 = x21(10:end);
y21 = y21(10:end);

x22 = x22(20:end);
y22 = y22(20:end);

x13=x13(5:end);
y13=y13(5:end)
```

```
x23=x23(35:end);
y23=y23(35:end);

x24=x24(5:end);
y24=y24(5:end);

PCAplot(x11,x21,x31,y11,y21,y31,1,2);
title('Case1 with PCA');
PCAplot(x12,x22,x32,y12,y22,y32,3,4);
title('Case2 with PCA');
PCAplot(x13,x23,x33,y13,y23,y33,5,6);
title('Case3 with PCA');
PCAplot(x14,x24,x34,y14,y24,y34,7,8);
title('Case4 with PCA');

function [x,y] = trackmotion(n,m,a,b,c,d)
    x=[];y=[];
    for j = 1 : n
        x1 = double(rgb2gray(m(:,:,:,j)));
        x1(:,1:a)=0; x1(:,b:end)=0;
        x1(1:c,:)=0; x1(d:end,:)=0;
        [V,I] = max(x1(:));
        [a1,b1] = find(x1 >= 11 * V /12);
        x(j) = mean(a1);
        y(j) = mean(b1);
    end
end

function PCAplot(x1,x2,x3,y1,y2,y3,a,b)
    l = min([length(x1),length(x2),length(y3)]);
    A = [x1(1:l); y1(1:l); x2(1:l); y2(1:l); x3(1:l); y3(1:l)];
    [m,n] = size(A);
    mn = mean(A,2);
    A = A-repmat(mn,1,n);
    [u,s,v] = svd(A,'econ');
    figure(a)
    plot(diag(s)/sum(diag(s)),'ro')
    xlabel('Principal Component');
    ylabel('Energy Percentage');
    figure(b)
    plot(v*s)
    legend('comp1','comp2','comp3','comp4','comp5','comp6')
    xlabel('Time')
    ylabel('Position')
end
```