AMATH 482 Homework 3: PCA Analysis

Leqi Wang

Feb 23, 2020

Abstract

In this assignment, we explore the application of singular value decomposition.

1 Introduction and Overview

We used principle component analysis (PCA or POD) to extract the motion in a spring-mass system filmed by three different probes with different level of stability and angle of releasing. We have an ideal case, a noisy case where camera is shaking, where there is pendulum motion in the system, and where there is also rotation on the mass. And we are going to compare the data we have in these four cases and come up with how PCA will deduct noise and extract primary motions of the system.

2 Theoretical Background

2.1 Spring Mass System

When a mass is released, it undergoes oscillation that satisfy

$$\frac{d^2 f(t)}{dt^2} = -\omega^2 f(t) \tag{1}$$

or

$$f(t) = A\cos\omega t + \omega_0 \tag{2}$$

where $f(t)$ is the function of displacement in the z-direction, ω represents the phase of equation, and ω_0 is the initial state of the spring mass system.

2.2 Singular Value Decomposition(SVD)

Compared to Spectral Decomposition, SVD can be applied to non-square matrix m x n. Matrix A can always be written as:

$$AV = U\Sigma \tag{3}$$

$$A = U\Sigma V^* \tag{4}$$

Where V is a unitary eigen vector matrix of A, U is m x m complex unitary matrix, Σ is m x n rectangular diagonal matrix. To write SVD in a reduced form, where Σ gets rid of zero rows and columns, A can be written as:

$$A = \hat{U}\hat{\Sigma}V^* \tag{5}$$

The rank of should not exceed rank of A, each sigma, it contains the singular values of A and reprents the energy of each singular component. By convention, we arrange these singular values from largest to smallest, along with its corresponding vector basis in U and V. Thus the SVD is just least-square fitting algorithm that projects the original matrix onto lower dimensional basis. The action of A on a vector $\vec{x}$ is now $\hat{U}\hat{\Sigma}V^*\vec{x}$, which represents the action of rotation - stretch - rotation and provides a result of low-rank approximation.

2.2.1 Principal Component Analysis(PCA) and Co-variance Matrix

PCA can be used to reduce redundancy in the spring mass system. If we have the data matrix (mean extracted, or "mean centered data")

$$X = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} \tag{6}$$

where row vectors are measurements from single experiment. The co-variance matrix of X:

$$C_x = X^T X \tag{7}$$

If we write X in terms of SVD, $X = U\Sigma V$, then $T = U\Sigma$ represents the principal components. To sum up, SVD of the mean centered data matrix direcly produces diagonal variances and we can use it to calculate the principal components projection.

2.2.2 Proper Orthogonal Decomposition(POD)

POD uses the same principal with PCA, only PCA works on vector basis, and POD works on function basis. When given a function $f(x,t)$, we can approximately write it into the sum of basis functions

$$f(x,t) \approx \sum_{j=1}^{N} a_j(t)\phi_j(x) \tag{8}$$

where $\phi_{\lambda_j}$are eigen functions of the system and coefficients $a_j(t)$ indicates the weights of the spatial modes.

3 Algorithm Implementation and Development

3.1 Tracking the object:

I first convert the image from color to gray scale, so that the white dot on the mass has the brightest value of 255. I crop the image to only include the rectangle where the light dot is in by a pre-determined prediction coordinates (in "Crop" section in code, see Appendix B). Then in the current matrix of gray scale, for each camera, I track the pixels where the light dot appear and get all of the coordinates by a combination of ind2sub and find function. I then take the mean of the vertical position and horizontal position separately as the center of mass of the object in current frame. I store the position data in a matrix and after finishing calculate the position for one camera, I subtract the mean from the data and devide the data matrix by $\sqrt{numFrames - 1}$. Repeating this process for four cases, and I have the data matrix X.

However, after I plot the scaled position of three different cameras in each case, I found that the data waves are not perfectly aligned and the final result I got is a bit noisy, so I went back to align these videos so that when they start, the object is at the same position, I also make the numFrames to be the smallest value so that the row data has the same length. Note: For case4, when the light dot disappears, I had to manually set a filter making the center of mass position equal to the position of the last frame, when the light dot is still there to eliminate some noise.

3.2 PCA analysis

Then I compute the reduced SVD of data matrix, the diagonal of Σ^2 represents the diagonal variance of each principal component and I also get the principal components of the data matrix, showing the oscillation motion with most energy, now the redundant motion track is reduced. When plotting, I present the variance in terms of energy percentage. For each figure, z-direction represents the vertical direction motion in the camera system, and x-y plane direction is the horizontal motion on the camera screen. When plotting, I also scaled the y axis so that each wave have approximately the same amplitude.

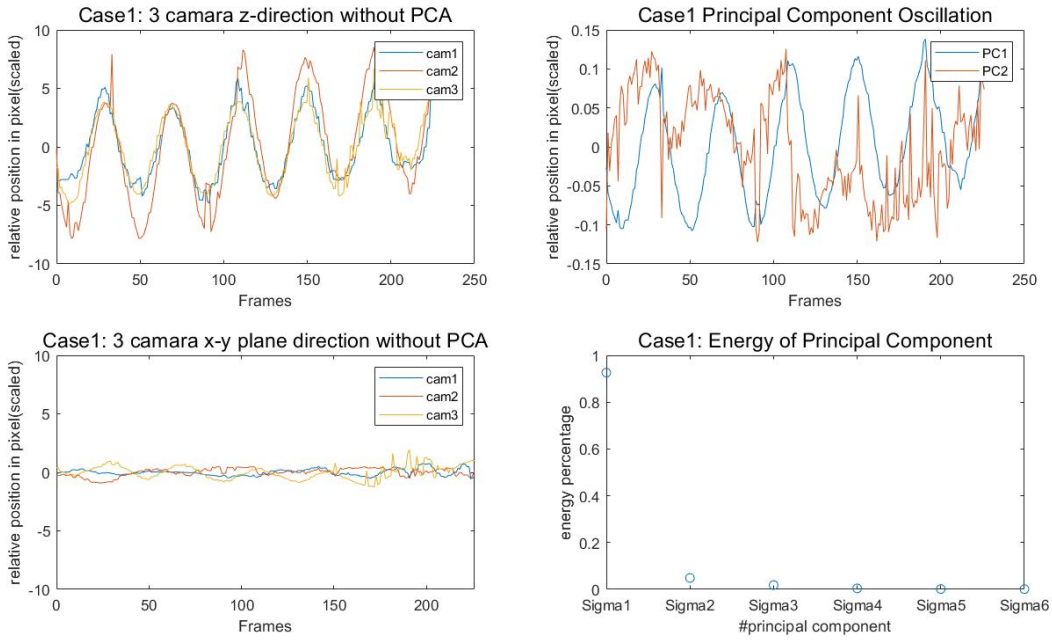

Figure 1: Ideal Case

4 Computational Results

4.1 Case1: ideal

As we can see from Figure1, the ideal case has clear sinusoidal waves for three cameras in the vertical direction and nearly nothing in the horizontal direction(fig1, left column). The first principal component includes about 92 percent of energy (fig1, down-right), it is a wave describing the moving object in vertical axis. The second component dropped vastly, including about 4 percent of energy. It is much noisier than the first principal component, but it also describe a sinusoidal wave. This happens because in the ideal case, the camera is not shaking, and we can expect or catch a clear motion from the object, described PC-1(fig1, up-right).

4.2 Case2: shaking camera

From Figure2, in left two columns which describes the vertical and horizontal movement of three cameras, case two contains more noisy data than case1. Still, the first principal component extract about 70 percent of the motion(fig2, right column), but compare to case1, it is harder to use just one principal component to describe the motion. Since there's not much horizontal displacement according to fig2, lower left graph, so the principal components analysis tells mostly the motion in the vertical direction.

4.3 Case3: horizontal displacement

Case3 involves pendulum motion where there's moth vertical and horizontal displacement. Luckily the camera is not shaking, so I was able to extract clear motion for the horizontal displacement. We can also see that the first principal component has the energy percent around 51 (fig3, lower right), this is because the data is not too well synchronized, and when we include the horizontal motion, the data matrix is more contaminated. However, we can still see that the dominate motion is wave like.

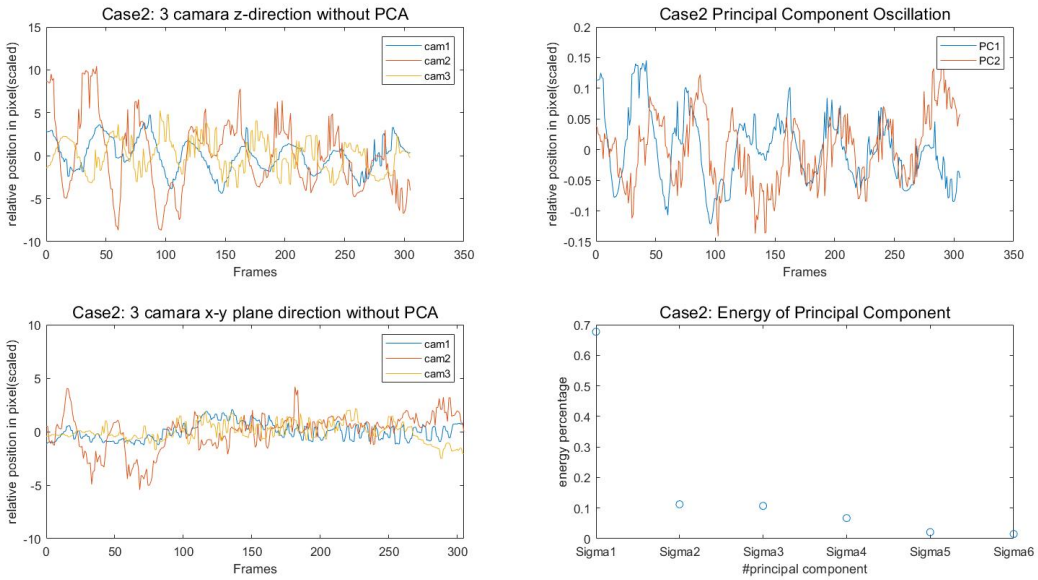

Figure 2: Noisy Case

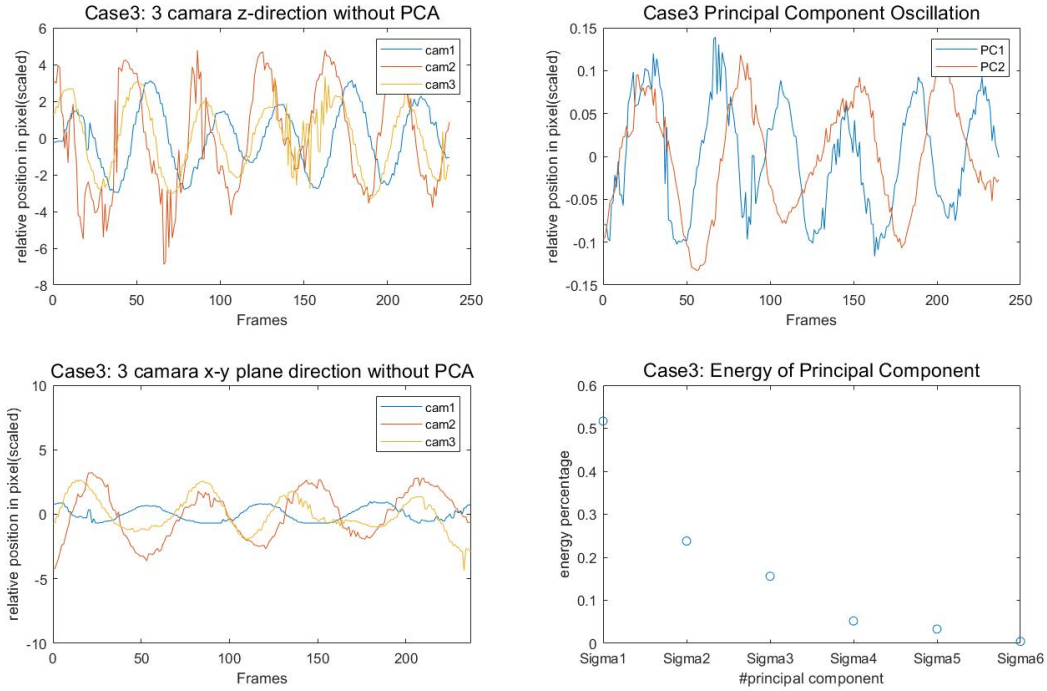

Figure 3: Case with horizontal movement

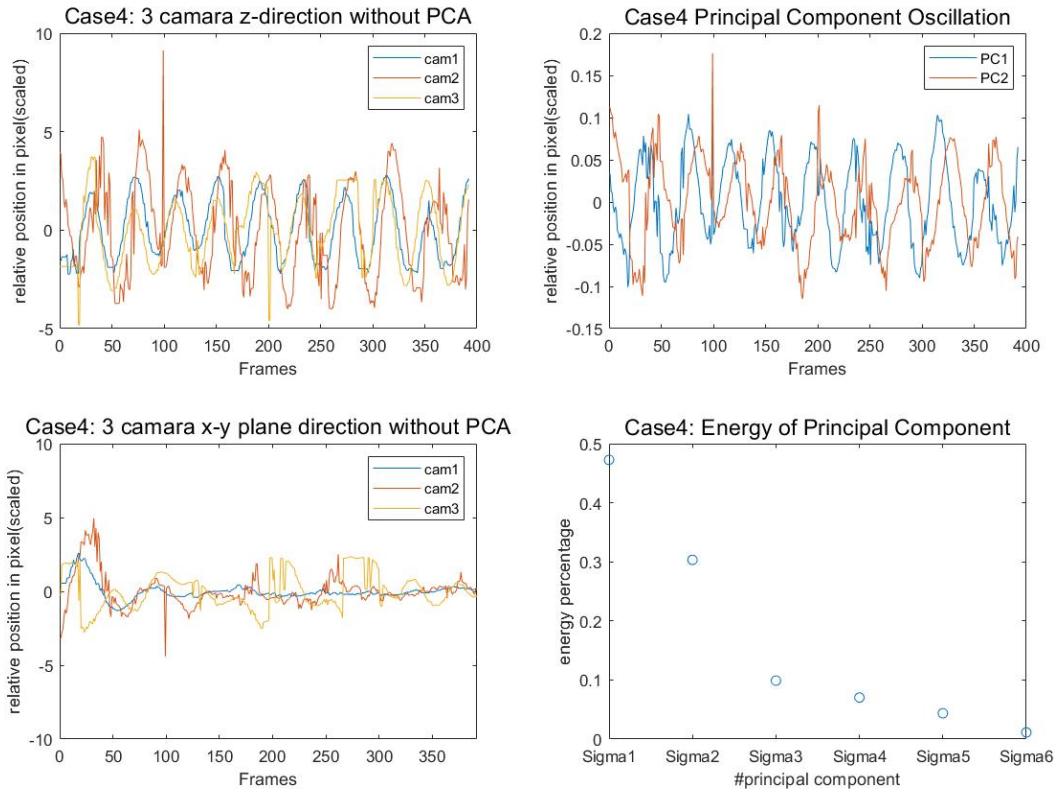

Figure 4: Case with horizontal movement and rotation

4.4 Case4: horizontal displacement with rotation

In case4 it's easy to see that in the vertical direction contains clear sinusoidal motions, but the horizontal motion is not as clear as the vertical one especially after I scaled the position, making both displacement data in the same scale. The first principal components gives most information of the data, first one takes about 48 percent energy and the second one takes about 30 percent, mostly describes the motion in the vertical direction.

5 Summary and Conclusions

PCA can produce the function basis of best fitting data, it is a good tool used for eliminating redundancies and reduce noise in data sets, and we are able to extract the dominant motion in the vertical direction in each case.

Appendix A MATLAB Functions

- Loading section: `load` load data. `implay()` is used to play the vedio.
- Crop the image: `prediction` variables are used to as the boundaries for `imcrop` function, so that there's less noise to be calculated. `zeros()` is used for initialize a variable.
- `rgb2gray` is used to convert the vedio from color to grescale.
- `imrop`: crop the image.

- `find(ind2sub(size(I), find(I == target)))`: used to track the vertical and horizontal position of the light dot.
- `svd(data, 'econ')`: calculate the reduced singular value decomposition of the centered data matrix X.
- Plotting: `linspace()` determine the x-axis in a graph to represent time (frames) dimension. `subplot`: create multiple graphs in one figure. `axix` to have horizontal direction and vertical direction displacement have the same scale when drawing graph.

Appendix B MATLAB Code

```
clear all; close all; clc

% HW3 Leqi Wang
%% load and play vedio
% load('cam1_1.mat') %  case1
% load('cam2_1.mat')
% load('cam3_1.mat')
% load('cam1_2.mat') % Case4
% load('cam2_2.mat')
% load('cam3_2.mat')
% load('cam1_3.mat') % Case4
% load('cam2_3.mat')
% load('cam3_3.mat')
load('cam1_4.mat') % Case4
load('cam2_4.mat')
load('cam3_4.mat')
% vids = {vidFrames1_1,vidFrames2_1(:,:,:,10:end),vidFrames3_1}; % case 1:
    shift graph to make peak aligned
% vids = {vidFrames1_2(:,:,:,10:end),vidFrames2_2,vidFrames3_2}; % Case4
% vids = {vidFrames1_3,vidFrames2_3,vidFrames3_3}; % Case4
vids = {vidFrames1_4,vidFrames2_4,vidFrames3_4}; % case 4
% implay(vids{1})
% implay(vids{2}
% implay(vids{3})

%% Crop
% prediction1 = [308.5 218.5 28 205]; % for cam1_1
% prediction2 = [229.5 66.5 59 295]; % for cam2_1
% prediction3 = [139.5 246.5 258 302]; % for cam3_1
% prediction1 = [305.5 198.5 139 212]; % for cam1_2
% prediction2 = [200.5 62.5 283 362]; % for cam2_2
% prediction3 = [130.5 307.5 228 267]; % for cam3_2
% prediction1 = [305.5 198.5 139 212]; % for cam1_3
% prediction2 = [200.5 62.5 283 362]; % for cam2_3
% prediction3 = [134.5 161.5 240 418]; % for cam3_3
prediction1 = [305.5 198.5 139 212]; % for cam1_4
prediction2 = [200.5 62.5 283 362]; % for cam2_4
prediction3 = [134.5 161.5 240 418]; % for cam3_4

predictions = {prediction1, prediction2, prediction3};
numFrames = min(min(size(vids{1},4), size(vids{2},4)), size(vids{3},4));
data1 = zeros(2, numFrames);
```

```
data2 = zeros(2, numFrames);
data3 = zeros(2, numFrames);
data = {data1, data2, data3};
% [J,rect] = imcrop(I');
%% extract x_1, y_1 ~ x_3, y_3 to construct covariance matrix;
for i = 1:3
    vid = vids{i};
    prediction = predictions{i};
    for t = 1:numFrames
        I = rgb2gray(vid(:,:,:,t));
        if(i == 3)
            I = I';
        end
%         I = double(imcrop(I, prediction)); % crop image
        I = imcrop(I, prediction);
        target = max(max(I)); % grayscale of the white spot, as target
        [vert, hori] = ind2sub(size(I), find(I == target));
        data{i}(1,t) = mean(vert); % center of mass
        if data{i}(1,t) < 90 % the light dot disappears
            data{i}(1,t) = data{i}(1,t - 1);
        end
        data{i}(2,t) = mean(hori);
        % imshow(I); drawnow
    end
    data{i}(1,:) = data{i}(1,:) - mean(data{i}(1,:));
    data{i}(2,:) = data{i}(2,:) - mean(data{i}(2,:));
    data{i}(1,:) = data{i}(1,:) / sqrt(numFrames - 1);
    data{i}(2,:) = data{i}(2,:) / sqrt(numFrames - 1);
    t = linspace(0, numFrames, numFrames);
    figure(1)
%     subplot(2, 2, 1)
    subplot(2, 2, 1)
    title('Case4: 3 camara z-direction without PCA', 'fontsize', 14)
    plot(t, data{i}(1, :))
    xlabel('Frames')
    ylabel('relative position in pixel(scaled)')
    hold on
    legend('cam1', 'cam2', 'cam3')

    subplot(2,2,3)
    title('Case4: 3 camara x-y plane direction without PCA', 'fontsize', 14)
    plot(t, data{i}(2, :))
    xlabel('Frames')
    ylabel('relative position in pixel(scaled)')
    legend('cam1', 'cam2', 'cam3')
    axis([0 numFrames -10 10])
    hold on
end

%% Case PCA analysis
X = [data{1}(1, :); data{1}(2, :); data{2}(1, :);data{2}(2, :); data{3}(1, :);
    data{3}(2, :)];
[u,s,v] = svd(X, 'econ');
lambda = diag(s).^2;
```

```
95  subplot(2, 2, 2) % PCA
96  plot(t,v(:,1:2)')
97  legend('PC1', 'PC2', 'PC3')
98  title('Case4 Principal Component Oscillation', 'fontsize', 14)
99  xlabel('Frames')
100 ylabel('relative position in pixel(scaled)')
101
102 %%
103 subplot(2, 2, 4)
104 plot(1:6, lambda/sum(lambda), 'o')
105 title('Case4: Energy of Principal Component','fontsize', 14)
106 xlabel('#principal component')
107 xticks([1 2 3 4 5 6])
108 xticklabels({'Sigma1','Sigma2','Sigma3','Sigma4','Sigma5','Sigma6'})
109 ylabel('energy percentage')
110
111 hold off;
```