AMATH 582 Homework 3

Qiangqiang Guo

Due February 24, 2021

Abstract

In this assignment, we will explore the principle component analysis (PCA) techniques by analyzing movie files created form three different cameras. These movies show a mass moving in different direction settings. We will find the trajectories of the mass under four different settings, and use PCA to extract the principle component of the movement.

1 Introduction and Overview

In our daily life, we (as well as the sensors) observe objects from different viewpoints. For example, multiple cameras may record the same object from different angles, making it quite different for each individual camera. However, which one is true? What is the fundamental mechanism behind all cameras? In order to find the truth, we can collect data from multiple different angles and find the most significant components, which can be regarded as the true dynamics. In this assignment, we will analyze the movement of a can under four different cases: ideal case, noisy case, horizontal displacement case, and horizontal displacement and rotation case. There are three cameras recording the movement video for each case. We will find the principle components for each case so that we can get the true knowledge of the movement of the can.

In the following of this report, we will first introduce the theoretical background used in this assignment (Section 2). Then, we show how to design and implement the analysis algorithm (Section 3). We show the computation results in Section 4 and conclude this report in Section 5.

2 Theoretical Background

The key task of this assignment is to conduct PCA on the can movement. In order to do this, we have to find the trajectory of the can. Therefore, there are mainly two parts in this assignment: find the trajectory and conduct PCA.

For the first part, observing that there is a light on the top of the can, we can use it as a marker to track the trajectory of the can. It is noted that the light sometimes cannot be observed in some movies, especially for the forth case. Therefore, we design two methods to find the trajectory of the can. For those movies that the light is always observable, we can track the brightest point near the can. For those movies that the light cannot always be observable, we find the can body (nearly white) by setting a threshold to identify white areas, and use the center point as the tracking point. Please check Section 3 for algorithm details.

For the second part, considering the we will have three sets of trajectories (in $(x-y)$ coordination) for each case, we can combine them and find the principle components using PCA. Let $(\mathbf{x}_i, \mathbf{y}_i)$ be the data collected overtime of the position in the $(x-y)$ plane of the camera $i = 1, 2, 3$. $\mathbf{x}_i \in \mathbb{R}^n$ where n is the number of frames. We combine them in a coordinate matrix

$$X = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_1 \\ \mathbf{x}_2 \\ \mathbf{y}_2 \\ \mathbf{x}_3 \\ \mathbf{y}_3 \end{bmatrix} \tag{1}$$

X is a $6 \times n$ matrix which consists of redundancies. Therefore, we consider the covariance between data sets. The covariance matrix is then

$$C_X = \frac{1}{n-1} X X^T \tag{2}$$

C_X is a square symmetric 6×6 matrix whose diagonal represents the variance of particular measurements and the off-diagonal terms are the covariances between measurement types. In order to remove the redundancies and find the component with maximum variance, we can diagonalize the C_X.

Observing that C_X is a square, symmetric matrix, We can use the most straightforward way, i.e., eigenvalue decomposition, to diagonalize the C_X. After this, we have

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

where S is the eigenvector matrix and Λ is a diagonal matrix whose entries correspond to the eigenvalues of $X X^T$, and the eigenvalues are sorted in the decreasing order. Then, we can produce the principal components projection:

$$Y = S^T X \tag{4}$$

3 Algorithm Development and Implementation

3.1 Read the movie data and setup

We firstly load the data files and have an initial understanding of them. Each video consists of hundreds of frames and it is in a $(a \times b \times c \times d)$ dimension, where $a \times b$ is the dimension of a single frame, $c = 3$ indicates this is a rgb frame, d is the number of frames. Considering that for a same case, the number of frames of different videos might be different, we trim each video to the minimum number of frames.

3.2 Find the trajectory of the mass

As discussed in Section 2, there is a light on top of the can. We can track this light to get the trajectory. To make the computation easier, we first convert the rgb frame to gray image. For each video, we manually extract the center point of the light in the first frame. With this starting point, we set those points that far from the starting point as 0 to focus on the can area. This can avoid identifying other places with bright colors as the can. Specifically, let (x_0, y_0) be the starting point, we set a threshold h (50 used in this assignment) so that any points outside the rectangle $[(x_0 - h, y_0 + h), (x_0 + h, y_0 - h)]$ ([upper left point, lower right point]) will be set to 0. Then, we take coordination of the maximum value in the rectangle, (x_{max}, y_{max}) as the current trajectory point. Meanwhile, we update the starting point $(x_0, y_0) = (x_{max}, y_{max})$ and go to process the next frame.

For those movies that the light cannot always be observable, we find the can body (nearly white) by setting a threshold to identify white areas, and use the center point as the tracking point. For each video, we manually extract the center point of the can body (the white part) for the first frame. Start from this point, we use find the white can body in a rectangle area around the center point. To do so, we also set a threshold to judge whether a point belongs to the can body or not. For example, we can simply identify the points whose gray value is larger than 230 to be the while body. Then, we take the coordination of the center point of all the identified points as the current trajectory point. Also, we update the starting point for the next frame to be the current trajectory point. The parameters, e.g., those thresholds, can be found at the MATLAB code appendix and the Github repository.

3.3 PCA analysis

For each case, we put all the trajectories into a coordinate matrix (Equation (1)). Then we sompute the mean for each row and subtract the mean. Then, we calculate the covariance matrix using Equation (2) and conduct eigenvalue decomposition on this matrix. After that, we sort the eigenvalues in decreasing order and sort the eigenvector matrix in the same order. Finally, we multiply the eigenvector matrix with the coordinate matrix to produce the principal components projection. This process is similar to the PCA process in the notes (page 405).

4 Computation results

4.1 Ideal case

Figure 1 shows the trajectories of the can in three cameras and the first three principle components for the ideal case. The left two columns show the position x and y and the different cameras are shown in different lines. The right column shows the first three principle components. From the camera data we can see that the x positions are relatively stable or do not show clear common patterns, while the y position is obviously an harmonic movement. The right figures show that the most significant component of these data is a harmonic movement, which is consistent with the analysis of the camera data. The second principle component is also a harmonic movement. This may be due to the fact that the three videos are not consistent in time. To see this, we trim the trajectory data so that they all begin from the first peak. The result is shown in Figure 2. It can be seen that there is only one harmonic principle component, which corresponds to the y direction movement.

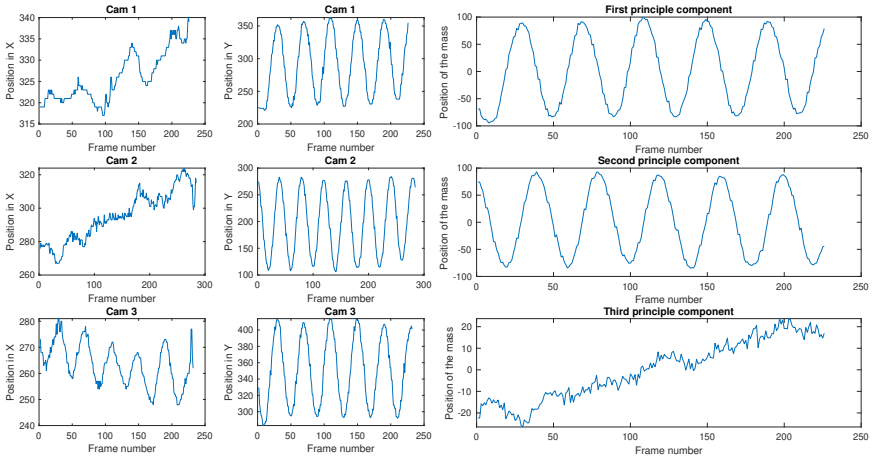

Figure 1: Trajectories of the can in three cameras and the first three principle components: Ideal case

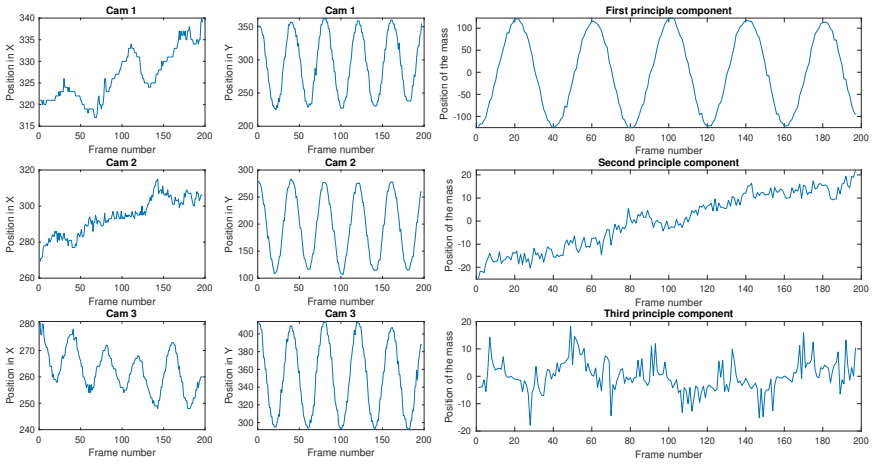

Figure 2: Trajectories of the can in three cameras and the first three principle components: Ideal case with trim

4.2 Noisy case

Figure 3 show the result of noisy case. The structure of this figure is similar to the ideal case. As we can see from the camera data, the trajectories in both x and y directions oscillate a lot. Although their are harmonic patterns in y direction, the patterns are not as clear as the ideal case. This is also reflected by the principle components. As shown in the right figures, we can still identify the harmonic movement from the first principle component. However, the trajectory is no as smooth as the ideal case. In addition, the first principle component is still the only component that we can observe a harmonic movement, indicating that there is only one degree of freedom.

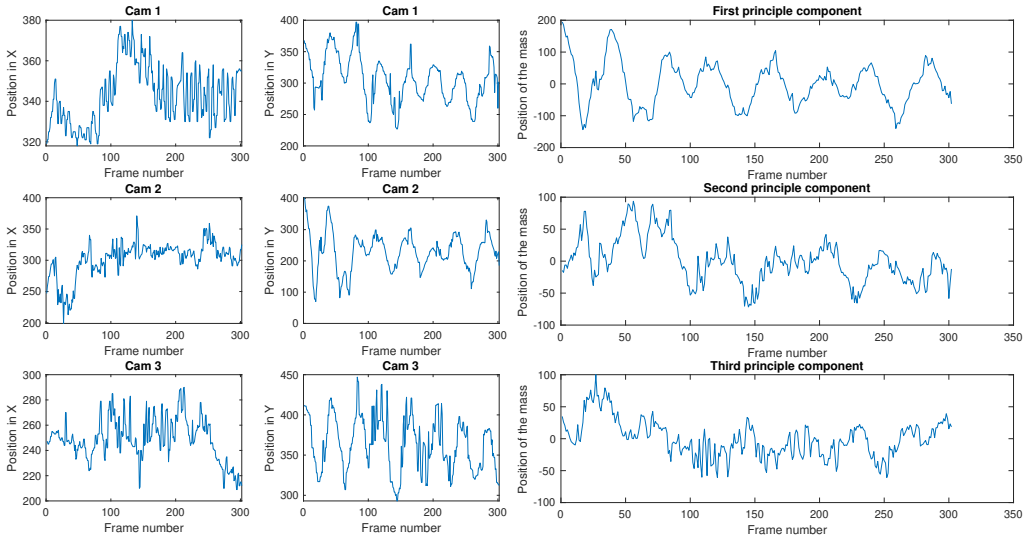

Figure 3: Trajectories of the can in three cameras and the first three principle components: Noisy case with trim

4.3 Horizontal displacement case

Figure 4 shows the result of horizontal displacement case. The structure of this figure is similar to the ideal case. As we can see from the camera data, the trajectories in both x and y directions move in a harmonic way. As shown in the right figures, there are two harmonic principle components. Based on their frequencies, the first principle component corresponds to the y direction movement, while the second principle component corresponds to the x direction movement.

4.4 Horizontal displacement and rotation

Figure 5 shows the result of horizontal displacement case. The structure of this figure is similar to the ideal case. As we can see from the camera data, the trajectories in y direction move in a typical harmonic way. Although the trajectories in x direction show some periodical pattern, the pattern is not very clear. This is due to the fact that the trajectory identification algorithm sometimes can not well distinguish the can from its background due to the changing light conditions. We used a fixed threshold for each case, which may not well capture the can body. As shown in the right figures, there are two harmonic principle components. The first principle component should correspond to the y direction movement, and the second principle component (not as smooth as the first principle component) corresponds to the x direction movement.

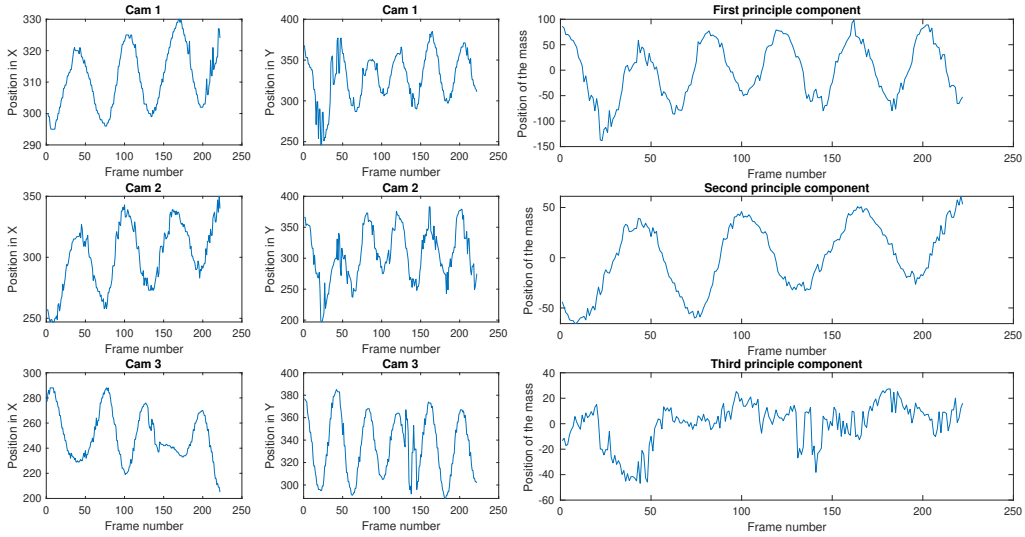

Figure 4: Trajectories of the can in three cameras and the first three principle components: Horizontal displacement case with trim

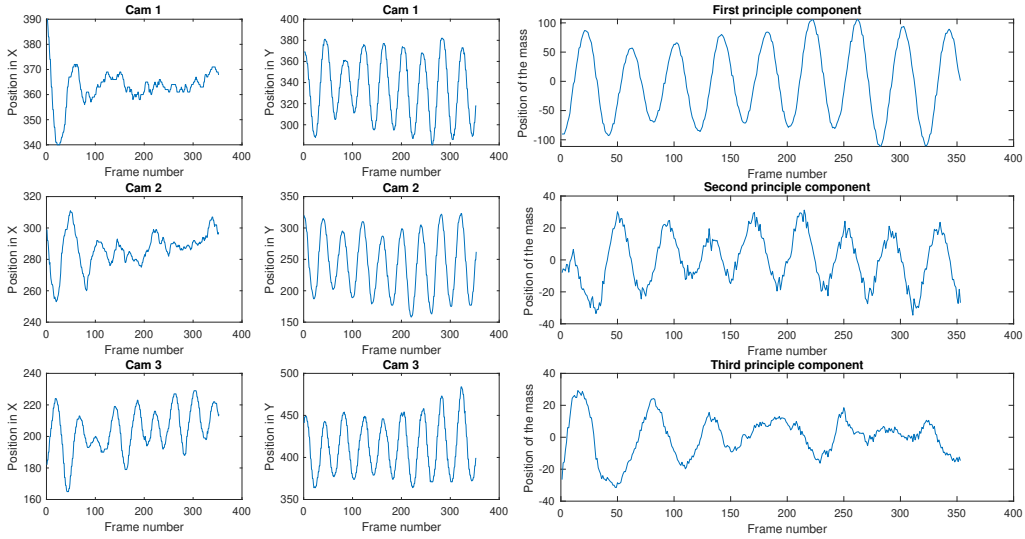

Figure 5: Trajectories of the can in three cameras and the first three principle components: Horizontal displacement and rotation case with trim

5 Summary and Conclusions

In this assignment, we designed algorithms to explore the PCA on four sets of movie data. Each set consists three videos recording a same can movement from different angles. We firstly extracted the trajectory of the can for each video by either tracking the light on the top of the can or identifying the can body. Then,

we used eigenvalue decomposition based principle component analysis to find the principle dynamics behind the data. The computation results showed that the PCA can well eliminate the redundancies and extract the "true" dynamics of the can. During this assignment, we have learned the power of PCA in identifying the principle information from various data.

Appendix A GitHub repository

`https://github.com/Guoqq17/UW-AMATH-582`

Appendix B Key MATLAB Functions

- `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.
- `[y,Fs] = [row,col] = find(X>a)` returns the row and column subscripts of each element in array `X` which is larger than `a`.
- `[row,col] = ind2sub(sz,ind)` 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`. Here `sz` is a vector with two elements, where `sz(1)` specifies the number of rows and `sz(2)` specifies the number of columns.
- `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]`.
- `[V,D] = eig(A)` returns diagonal matrix `D` of eigenvalues and matrix `V` whose columns are the corresponding right eigenvectors, so that `A*V = V*D`.
- `D = diag(v)` returns a square diagonal matrix with the elements of vector `v` on the main diagonal.
- `[B,I] = sort(A)` returns a collection of index vectors for sorting the elements of `A` in ascending order. `I` is the same size as `A` and describes the arrangement of the elements of `A` into `B` along the sorted dimension. For example, if `A` is a vector, then `B = A(I)`.
- `implay(filename)` opens the Video Viewer app, displaying the content of the file specified by `filename`.

Appendix C MATLAB Code

We only include the codes for the ideal case in here. Please refer to the Github repository for the codes of other cases.

```
%% load data
close all;
clear;
clc;

for i=1:1:3
    for j=1:1:4
        load(strcat('cam', num2str(i), '_', num2str(j), '.mat'))
    end
end

%% visulize data
implay(vidFrames1_1)
%%
% video 1_1
[a1, b1, c1, d1] = size(vidFrames1_1);
thresh = 20;                              % the search threshold to find the light spot
```

```
start_point = [320,230];                % start point to search
x1=zeros(1,d1);
y1=zeros(1,d1);
for i=1:1:d1
    img = rgb2gray(vidFrames1_1(:,:,:,i));
    img(:,1:start_point(1)-thresh) = 0;
    img(:,start_point(1)+thresh:end) = 0;
    img(1:start_point(2)-thresh,:) = 0;
    img(start_point(2)+thresh:end,:) = 0;
    [val,ind] = max(img(:));
    [p1,p2]=ind2sub(size(img),ind);
    x1(i)=p2;
    y1(i)=p1;
    start_point=[p2, p1];
end

% video 2_1
[a2, b2, c2, d2] = size(vidFrames2_1);
thresh = 20;
start_point = [276,276];
x2=zeros(1,d2);
y2=zeros(1,d2);
for i=1:1:d2
    img = rgb2gray(vidFrames2_1(:,:,:,i));
    img(:,1:start_point(1)-thresh) = 0;
    img(:,start_point(1)+thresh:end) = 0;
    img(1:start_point(2)-thresh,:) = 0;
    img(start_point(2)+thresh:end,:) = 0;
    [val,ind] = max(img(:));
    [p1,p2]=ind2sub(size(img),ind);
    x2(i)=p2;
    y2(i)=p1;
    start_point=[p2, p1];
end

% video 3_1
[a3, b3, c3, d3] = size(vidFrames3_1);
thresh = 20;
start_point = [323,273];
x3=zeros(1,d3);
y3=zeros(1,d3);
for i=1:1:d3
    img = rgb2gray(vidFrames3_1(:,:,:,i));
    img(:,1:start_point(1)-thresh) = 0;
    img(:,start_point(1)+thresh:end) = 0;
    img(1:start_point(2)-thresh,:) = 0;
    img(start_point(2)+thresh:end,:) = 0;
    [val,ind] = max(img(:));
    [p1,p2]=ind2sub(size(img),ind);
    x3(i)=p1;
    y3(i)=p2;
    start_point=[p2, p1];
end

% trim to same size
n = min(min(d1,d2),d3);
X=[x1(1:n);y1(1:n);x2(1:n);y2(1:n);x3(1:n);y3(1:n)];

% PCA
[m,n]=size(X);
mn=mean(X,2);
X=X-repmat(mn,1,n);
Cx=(1/(n-1))*X*X';
[V,D]=eig(Cx);
lambda=diag(D);
[dummy, m_arrange]=sort(-1*lambda);
V=V(:,m_arrange);
Y=V'*X;
```

```
86
87  figure(1)
88  subplot(3,4,1)
89  plot(x1)
90  xlabel('Frame number')
91  ylabel('Position in X')
92  title('Cam 1')
93  subplot(3,4,2)
94  plot(y1)
95  xlabel('Frame number')
96  ylabel('Position in Y')
97  title('Cam 1')
98  subplot(3,4,5)
99  plot(x2)
100 xlabel('Frame number')
101 ylabel('Position in X')
102 title('Cam 2')
103 subplot(3,4,6)
104 plot(y2)
105 xlabel('Frame number')
106 ylabel('Position in Y')
107 title('Cam 2')
108 subplot(3,4,9)
109 plot(x3)
110 xlabel('Frame number')
111 ylabel('Position in X')
112 title('Cam 3')
113 subplot(3,4,10)
114 plot(y3)
115 xlabel('Frame number')
116 ylabel('Position in Y')
117 title('Cam 3')
118 subplot(3,4,[3,4])
119 plot(Y(1,:))
120 xlabel('Frame number')
121 ylabel('Position of the mass')
122 title('First principle component')
123 subplot(3,4,[7,8])
124 plot(Y(2,:))
125 xlabel('Frame number')
126 ylabel('Position of the mass')
127 title('Second principle component')
128 subplot(3,4,[11,12])
129 plot(Y(3,:))
130 xlabel('Frame number')
131 ylabel('Position of the mass')
132 title('Third principle component')
133
134 %% trim the data so that they are consistent in time, and redo the experiment
135 % trim to same size
136 n = min(min(d1,d2),d3);
137 x1=x1(30:n);
138 y1=y1(30:n);
139 x2=x2(40:n+10);
140 y2=y2(40:n+10);
141 x3=x3(30:n);
142 y3=y3(30:n);
143 X=[x1;y1;x2;y2;x3;y3];
144
145 % PCA
146 [m,n]=size(X);
147 mn=mean(X,2);
148 X=X-repmat(mn,1,n);
149 Cx=(1/(n-1))*X*X';
150 [V,D]=eig(Cx);
151 lambda=diag(D);
152 [dummy, m_arrange]=sort(-1*lambda);
153 V=V(:,m_arrange);
```

```
Y=V'*X;

figure(2)
subplot(3,4,1)
plot(x1)
xlabel('Frame number')
ylabel('Position in X')
title('Cam 1')
subplot(3,4,2)
plot(y1)
xlabel('Frame number')
ylabel('Position in Y')
title('Cam 1')
subplot(3,4,5)
plot(x2)
xlabel('Frame number')
ylabel('Position in X')
title('Cam 2')
subplot(3,4,6)
plot(y2)
xlabel('Frame number')
ylabel('Position in Y')
title('Cam 2')
subplot(3,4,9)
plot(x3)
xlabel('Frame number')
ylabel('Position in X')
title('Cam 3')
subplot(3,4,10)
plot(y3)
xlabel('Frame number')
ylabel('Position in Y')
title('Cam 3')
subplot(3,4,[3,4])
plot(Y(1,:))
xlabel('Frame number')
ylabel('Position of the mass')
title('First principle component')
subplot(3,4,[7,8])
plot(Y(2,:))
xlabel('Frame number')
ylabel('Position of the mass')
title('Second principle component')
subplot(3,4,[11,12])
plot(Y(3,:))
xlabel('Frame number')
ylabel('Position of the mass')
title('Third principle component')
```