AMATH 582 HW3: Track the paint can

Huasong Zhang
Department of Applied Mathematics
University of Washington
Seattle, WA 98195
huasongz@uw.edu

Abstract

In this assignment, we will be working on a moving paint can and we intend to find the moving trajectories for four different cases: ideal case which is just simple harmonic oscillation with ups and downs, noisy case, horizontal displacement, and horizontal displacement with rotation. Three cameras are set up to take videos of the movement. After extracting the x and y coordinates of the paint can in each camera, We are going to use PCA on the coordinates and extract the principle component of the movement.

1 Introduction and Overview

The majority of the objects in our daily life are moving all the time. How is this object moving and what is the moving trajectories are the things that we care about. However, for different people from different angles observing the movement, they may have different answers toward one movement. In order to figure out the truth, it is the same as solving a case. We need to collect data from various directions and find out the most significant ones, which are the truth we are trying to find.

Therefore, in this assignment, our task is to analyze the movement of the paint can. There are four movements with different cases, which are ideal case, noisy case, horizontal displacement, and horizontal displacement and rotation; and for each movement, there are three cameras taking videos from different directions. What we will do is to find out the moving trajectories by locating the light on the paint can, saving the coordinate, and doing Principle Component Analysis on three set of coordinates in order to find the principle component.

2 Theoretical Background

There are mainly two parts in this assignment. The first part is to find out the coordinates for the movement. The second part is doing PCA on the coordinate matrix we got.

In this assignment, the theoretical part is in the second part, which we need the Principle Component Analysis knowledge. Since we have three sets of coordinates describing the same movement, We want to combine them and find out which components have the most impact on the movement.

We find out the trajectories from three cameras and denote as $x_1, y_1, x_2, y_2, x_3, y_3$. And we put them in a big coordinate matrix:

$$X = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} \tag{1}$$

So the Coord matrix has dimension m by n, where m is the number of measurements and n represents the number of data points that were taken for each measurement. There are redundancies, therefore, we want to compute the covariance

matrix of Coord matrix X. To do that, we use the formula

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

to get an unbiased estimator of covariance with dimension m by m. We want to remove redundancy and identify the components with the maximum variance. Therefore, we want to diagonalize this C_x matrix.

Since XX^T is a square and symmetric matrix, we can apply eigenvalue decomposition on it; and after we do that, we will get:

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

S is the eigenvector matrix, and Λ is a diagonal matrix with eigenvalues pf XX^T on the diagonal and the eigenvalues are sorted in the decreasing order.

Then, we can take

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

to project the original data in X matrix onto the principle component basis.

3 Algorithm Implementation and Development

At first, we need to load the file. For each file loaded, the dimension is $m * n * p * t$. The whole video is composed of hundreds of images. The first two elements m and n is the size for each image. The third element p is equal to 3 which means this is a rbg image which has three layers. The last element t means number of images in this video.

In order to easily deal with each image, I first change each color image into gray-scale by using the command "rgb2gray". And I implemented this command in a for loop to let it automatically loop over all the images in the video.

After we got the gray-scale images, there are many places where the color are very similar. In order to focus on the position of the paint can, I blacked out the background environment, and only leave a small window where the paint can can be fulled tracked. By doing this step, we can clean out the noises in the surroundings and avoid distracting.

There is a light on the paint can, and that is the place where the pixel has the largest magnitude. Therefore, I used "max" and "ind2sub" commands to locate the coordinates of the light. In the "ind2sub", the first output is from m which means the y coordinate and the second output is the x coordinate. After getting the coordinates, I saved them into the X and Y. Therefore, I get the entire X and Y coordinates of the paint can for one video clip.

After doing the same process for the videos taken by three cameras for the ideal case, I realized that the time is different in these three videos. In order to put them together to do PCA later, I find the shortest one and cut the other long clips to keep them the same length.

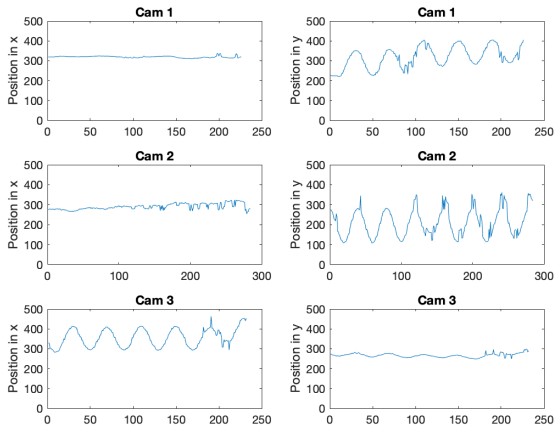
Then I put all the X and Y coordinates together into a big matrix called "Coord" and this matrix has dimension 6 by n where n is the length of the coordinates. Then, we compute the mean for each row and subtract the mean. After that, we use the "Coord" to multiply the transpose of itself and divided by $(1-n)$ to get the covariance matrix, and we denote this matrix as C_x. This matrix now is a n by n square matrix. We now want to diagonalize this matrix so we compute the eigenvectors and eigenvalues of C_x. The eignenvectors are the component of this matrix and the eigenvalues are the variances. Then we use "sort" to sort the eigenvalues with the first one being the biggest and then sort the eignevector matrix in the same order. Now the eigenvector matrix is the principle component matrix. We now multiply the eigenvector matrix with the Coord matrix, we can get the projection of the coordinate matrix into the principle component basis.

The above implementation explains how to process the ideal case. The other cases use the exact same idea and methods.

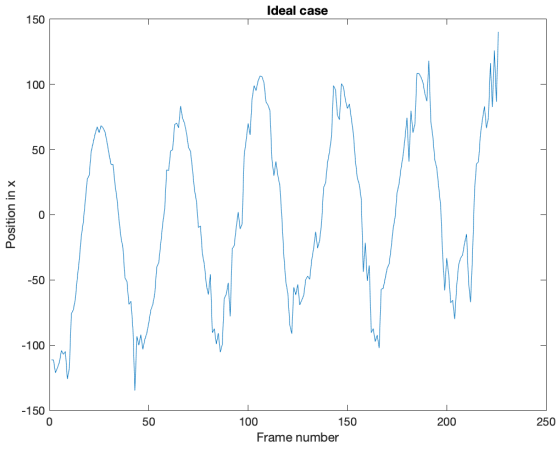
4 Computational Results

Figure 1 is the ideal case of the movement. Figure 1a shows the trajectories in x direction and y direction separately. The left column are the x positions and right column are the y positions. Each row represents one camera. From this figure we can see that the x position is pretty stable; there is almost no movements in the x direction. In y direction, the paint can moves ups and downs. Figure 1b shows what it looks like from the principle component.

Figure 2 shows the movement with noise. As we can see from the figure, on the left column, the x positions are not as stable as in Figure 1a. It oscillates. Also, on the right column, the movement in y direction is not as regular and smooth as in Figure 1a. The amplitude of the oscillations varies. And Figure 2b shows the movement in principle component.

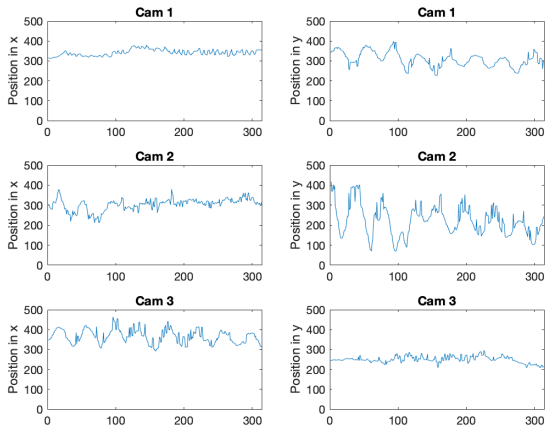


(a) x and y positions from three cameras

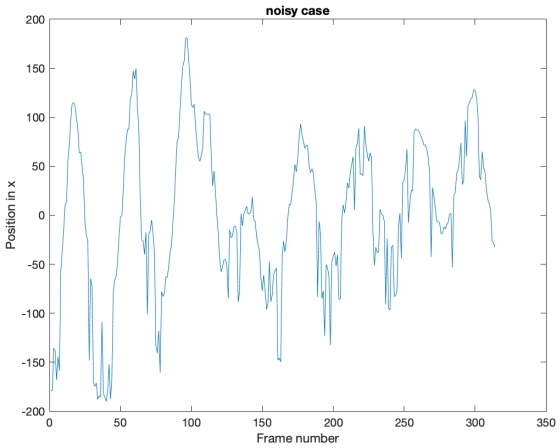


(b) principle component of the movement

Figure 1: Ideal Case



(a) x and y positions from three cameras



(b) principle component of the movement

Figure 2: Noisy Case

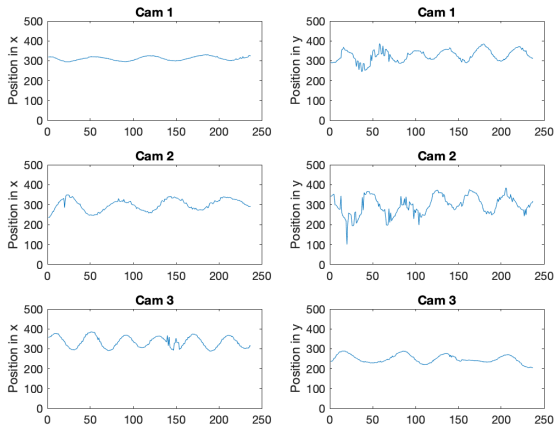
Comparing with Figure 1b, the movement trajectory is not as smooth as shown in Figure 1b. It's obvious that there are noises.

The third case is the horizontal displacement. As we can see from 3a, the movement in x direction becomes stronger and the movement in y direction reduces comparing with the first two cases. And the motion in x direction seems periodic. Therefore, we can know that in this case, the movement is mainly in x direction with horizontal displacement. And Figure 3b shows the movement of principle component.

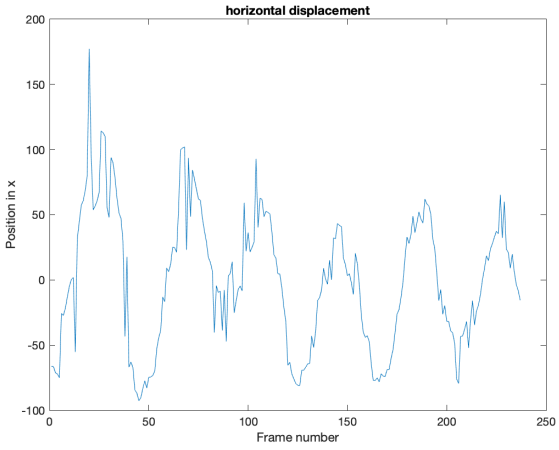
The last case is horizontal displacement and rotation. Since the rotation also happens in x direction, we can see from Figure 4a that the position in x direction are not as regular as that in the third case. We may know that there is rotation get involved. And from Figure 4b we can see that the plot in principle component direction becomes even more complex. So the movement is complicated.

5 Summary and Conclusions

PCA is really useful in finding the most significant components from a several set of data recording the same thing. By finding the covariance matrix, doing eigenvalue decomposition on the covariance matrix, we can get a diagonal matrix

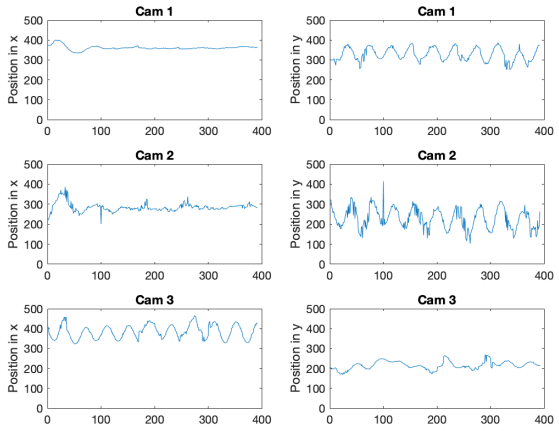


(a) x and y positions from three cameras

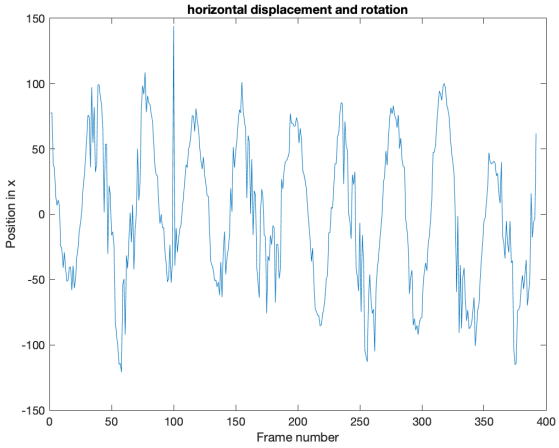


(b) principle component of the movement

Figure 3: Horizontal Displacement



(a) x and y positions from three cameras



(b) principle component of the movement

Figure 4: Ideal Case

with eigenvalues on the diagonal and corresponding eigenvector matrix. The eigenvector matrix will be our principle component basis. Then, we can multiply the eigenvector matrix with the original matrix and get a projection of the original matrix on the principle component basis. Then we will what the original object or movement looks like. Doing PCA is very useful in taking out the most important information and get rid of the redundancy.

Appendix A MATLAB functions used and brief implementation explanation

rgb2gray: change image from rgb to gray-scale with black and white.
max: find out the value and index of the biggest value in a matrix.
ind2sub: locate the index of the maximum value in the matrix, get x and y coordinate.
eig: eigenvalue decomposition, returns eigenvectors and eigenvalues.
sort: sort the numbers in assigned order, returns the order(permutation).

Appendix: MATLAB codes

The following code is just for the first ideal case. The other parts are very similar to this.

```
clear all;clc
% load files
for i = 1:3
    for j = 1:4
        load(['cam' num2str(i) '_' num2str(j) '.mat'])
    end
end

%% Ideal Case
[a11,b11,c11,d11] = size(vidFrames1_1);
X11 = [];
Y11 = [];
for i = 1:d11
    img = rgb2gray(vidFrames1_1(:,:,:,i));
    Cam11(:,:,i) = double(img);
    img(:,1:300) = 0;
    img(:,400:end) = 0;
    img(1:200,:) = 0;
    %imshow(img)
    [Max Ind] = max(img(:));
    [y11 x11] = ind2sub(size(img),Ind);
    X11 = [X11 x11];
    Y11 = [Y11 y11];
end

[a12,b12,c12,d12] = size(vidFrames2_1);
X12 = [];
Y12 = [];
for i = 1:d12
    img = rgb2gray(vidFrames2_1(:,:,:,i));
    Cam21(:,:,i) = double(img);
    img(:,1:230) = 0;
    img(:,350:end) = 0;
    img(1:100,:) = 0;
    img(370:end,:) = 0;
    % imshow(img)
    [Max Ind] = max(img(:));
    [y12 x12] = ind2sub(size(img),Ind);
    X12 = [X12 x12];
    Y12 = [Y12 y12];
end

[a13,b13,c13,d13] = size(vidFrames3_1);
X13 = [];
Y13 = [];
for i = 1:d13
    img = rgb2gray(vidFrames3_1(:,:,:,i));
    Cam31(:,:,i) = double(img);
    img(:,1:250) = 0;
    img(1:200,:) = 0;
    img(350:end,:) = 0;
    % imshow(img)
    [Max Ind] = max(img(:));
    [y13 x13] = ind2sub(size(img),Ind);
    X13 = [X13 x13];
    Y13 = [Y13 y13];
end

figure(1)
```

```
subplot(3,2,1)
plot(X11)
ylabel('Position in x')
ylim([0 500])
title('Cam 1')
subplot(3,2,2)
plot(Y11)
title('Cam 1')
ylabel('Position in y')
ylim([0 500])
subplot(3,2,3)
plot(X12)
ylabel('Position in x')
ylim([0 500])
title('Cam 2')
subplot(3,2,4)
plot(Y12)
title('Cam 2')
ylabel('Position in y')
ylim([0 500])
subplot(3,2,5)
plot(X13)
ylabel('Position in x')
ylim([0 500])
title('Cam 3')
subplot(3,2,6)
plot(Y13)
title('Cam 3')
ylabel('Position in y')
ylim([0 500])
print(gcf,'-dpng','Part1_1.png');

Xmin = min([length(X11) length(X12) length(X13)]);
if length(X11) > Xmin
    X11 = X11(1:Xmin);
    Y11 = Y11(1:Xmin);
end
if length(X12) > Xmin
    X12 = X12(1:Xmin);
    Y12 = Y12(1:Xmin);
end
if length(X13) > Xmin
    X13 = X13(1:Xmin);
    Y13 = Y13(1:Xmin);
end

Coord = [X11;Y11;X12;Y12;X13;Y13];
[m,n]=size(Coord); % compute data size
mn=mean(Coord,2); % compute mean for each row
Coord=Coord-repmat(mn,1,n); % subtract mean

Cx=(1/(n-1))*Coord*Coord'; % covariance
[V,D]=eig(Cx); % eigenvectors(V)/eigenvalues(D)
lambda=diag(D); % get eigenvalues
[dummy,m_arrange]=sort(-1*lambda); % sort in decreasing order
lambda=lambda(m_arrange);
V=V(:,m_arrange);
Y=V'*Coord; % produce the principal components projection
```

```
figure(2)
plot(Y(1,:))
xlabel('Frame number')
ylabel('Position in x')
title('Ideal case')
print(gcf,'-dpng','Part1_2.png');
```