AMATH 482 Homework 3

Jiaqi Su

February 24, 2021

Abstract

In real life, one essential way to study the motion of a system in the space is to analyze the dimension of the motion. A practical method would be collecting data in several unknown dimensions and figure out the real dimensions where the position of the objects vary significantly compared to other dimensions using the Singular Value Decomposition and the Principal Component Analysis. In our experiment of a string-mass system, our goal is to study the motions of the system in 4 different cases based on data of 6 dimensions from 3 cameras surrounding the system. Through comparing the principal component analysis results, we can have idea about the dimensions of motion in each case.

1 Introduction and Overview

The expectation of this problem is to study the motion of a string-mass system in 4 cases: ideal case, noisy case, horizontal displacement, and horizontal and rotation case. To understand the motion in each case, we are interested about how the position of object change in each dimension. One main challenge in this project is to locate and track the position of the mass based on given videos. After we extract spatial information of the object from each of three camera's video, we can aggregate data and use the principal component analysis to figure out the which dimension contribute most to approximate the motion.

2 Theoretical Background

2.1 Singular Value Decomposition

Applying a matrix A to a vector x from the left can be seen as rotating and stretching a vector geometrically, and turning circles into ellipses. Therefore every matrix A has a Singular Value Decomposition such that

$$AV = U\Sigma \tag{1}$$

where V is a unitary matrix with v_1 and v_2 unit vectors as columns, U is a unitary matrix with u_1 and u_2 the unit vectors on the semi-axes of the resulting ellipses, and Σ is a diagonal matrix with entries representing the lengths of the semi-axes of the ellipses. Since V is unitary, A can be isolated that

$$A = U\Sigma V^* \tag{2}$$

For non-square $m \times n$ matrix A, assuming $m > n$ we can have the reduced singular value decomposition that

$$A = \hat{U}\hat{\Sigma}V^* \tag{3}$$

where $\hat{U}$ is in $m \times n$ format.
More generally, columns of 0 will be padded to $\hat{U}$ to have a $m \times m$ matrix U. So the standard singular value decomposition is

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

where U $in\mathbb{R}^{m\times m}$, V $in\mathbb{R}^{n\times n}$, and $\Sigma in\mathbb{R}^{m\times n}$ is diagonal.

2.2 Principal Component Analysis

One essential application of SVD is the Principal Component Analysis. Given a matrix X that has several vectors as its row:

$$X = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{5}$$

The co-variance of the row vectors is defined as

$$C_X = \frac{1}{n-1} X X^T \tag{6}$$

where entry $C_{X\,i,j} = \sigma_{i,j} = cov(i,j)$. And if we let matrix $A = \frac{1}{\sqrt{n-1}} X$, $C_X = \frac{1}{n-1} X X^T = A A^T$. Combined with our study about SVD, $C_X = A A^T = U \Sigma^2 U^T$, and the columns of U provide us the basis of principal components of original matrix X, which gives us an idea about the direction that matters most in space of X.

3 Algorithm Implementation and Development

3.1 Locating and extracting position of mass

Given the video, we first divide it into frames at each time. After that the vital task is to locate and extract the position of the mass at each frame so that we can use then to analyze the whole motion. For all four of the cases, the basic algorithm is the same. To implement this algorithm, we need following developments:

Algorithm 1: Locating and extracting position of mass

```
Load videos from three cameras
camN_1.mat, camN_2.mat, camN_3.matalignandtrimthevideostothesamelengthnum
for j = 1 : 3 do
    estimate the smaller rectangular frame of the system in the video
    for j = 1 : num do
        transform the image matrix from rgb to gray
        crop the matrix using the rectangular range we estimated
        find the grayscale value of the bright spot on the bass
        find all indices of pixels matching the grayscale value of the bright spot and average to get the mass
        center position
    end for
    scale the position to center at 0
end for
```

1. If there are other white or bright part in the vedio, the value of position will be influenced when we locate the bright spot on the bass. Therefore, using an estimated frame to crop the video can help us better focus on our system.

2. The singular value along the diagonal of Σ relates to the variance of the position value, so scale the position to center at 0 by subtracting the mean can simplify our computation. And factor by $\frac{1}{\sqrt{n-1}}$ is needed for later SVD calculations.

3.2 SVD and Principal Component Analysis

After we have the position data that includes 6 dimensions from 3 cameras, we can simply combine them into a matrix X and apply Principal Component Analysis.

4 Computational Results

4.1 Ideal case

4.2 Noisy case

4.3 Case3 with horizontal displacement

5 Summary and Conclusions

Appendix A MATLAB Functions

MATLAB functions used in the implementations of Algorithms.

- `M = max(X,[],all)` returns maximum entry value `M` in matrix `X`.

Appendix B MATLAB Code

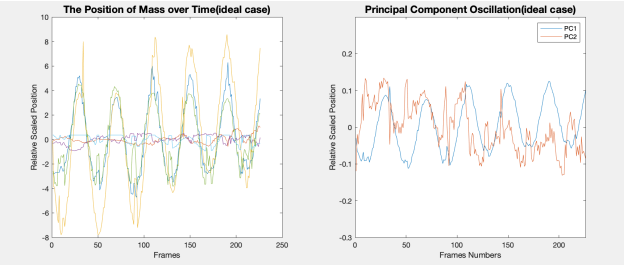

Figure 1: Here is the graph of position over time and pca analysis of ideal case

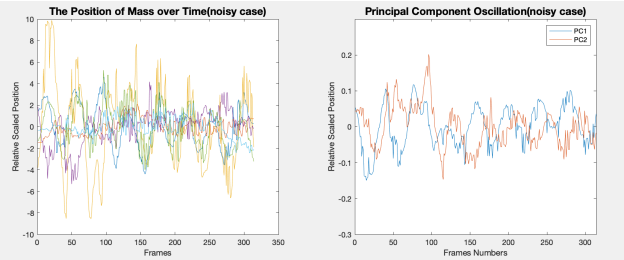

Figure 2: Here is the graph of position over time and pca analysis of noisa case

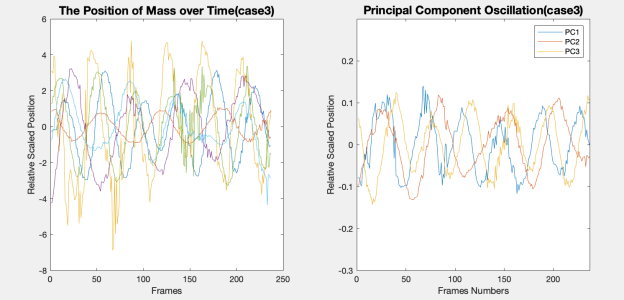

Figure 3: Here is the graph of position over time and pca analysis of ideal case

```
% Clean workspace
clear all; close all; clc

% -- Load Movie
load cam1_3.mat;
[h1,w1,rgb1,num1] = size(vidFrames1_3);
load cam2_3.mat;
[h2,w2,rgb2,num2] = size(vidFrames2_3);
load cam3_3.mat;
[h3,w3,rgb3,num3] = size(vidFrames3_3);
num = min([num1,num2,num3]);
t = 1:num;

% trim movie clips
vidF1 = vidFrames1_3;%case2
vidF2 = vidFrames2_3(:,:,:,1:num);
vidF3 = vidFrames3_3(:,:,:,1:num);

% range1 = [300 200 50 250]; %case1
% range2 = [220 60 70 320];
% range3 = [120 300 150 200];
% range1 = [300 200 100 250]; %case2
% range2 = [160 50 250 350];
% range3 = [120 300 180 220];
range1 = [300 200 150 220]; %case2
range2 = [160 50 300 350];
range3 = [120 150 250 420];

vids = {vidF1,vidF2,vidF3};
ranges = {range1,range2,range3};
X = zeros(6,num);

%%
maxval = zeros(1,num);
for i=1:3
    for j=1:num
        vid = vids{i};
        M = rgb2gray(vid(:,:,:,j));
        if i==3
            M = M';
        end
        M = imcrop(M,ranges{i});
        light = max(M,[],'all');
        [lightx, lighty] = ind2sub(size(M), find(M == light));
        X(2*i-1,j) = mean(lightx);
        X(2*i,j) = mean(lighty);
    end

    X(2*i-1,:) = (X(2*i-1,:)-mean(X(2*i-1,:)))/sqrt(num-1);
    X(2*i,:) = (X(2*i,:)-mean(X(2*i,:)))/sqrt(num-1);
    figure(1)

    subplot(1,2,1)
    title('The Position of Mass over Time(case3)', 'fontsize', 14)
    plot(t, X(2*i-1,:))
    hold on
    plot(t, X(2*i,:))
    xlabel('Frames')
    ylabel('Relative Scaled Position')
end
```