AMATH 482 Homework 3

Ariel Luo

Feb 20, 2020

Abstract

We are going to use the Principle Component Analysis on various videos of a spring-mass system. Through the videos taken from different angles, we are going to analyze the movement of the mass.

1 Introduction and Overview

In this homework, we are working with videos of a spring-mass system from 3 different cameras and angles for 4 cases, the ideal case, the noisy case which means that the camera is not as stable as the ideal case, the horizontal displacement case which the mass will be release off-center to create motion in the x-y plane instead of completely vertical motions and the horizontal displacement case with rotation which creates a circular motion on top of the last case. We are going to use Principle Component Analysis to obtain the vertical position change of the mass.

2 Theoretical Background

As we learned from our lectures, the purpose of the Principal Component Analysis(PCA) is to change the basis of given data so that the data will be uncorrelated. By doing this, we can observe the change in our data since there will be no redundancies and we can identify the data with the maximum variance. In order to achieve our goals, we need a covariance matrix that can tells us about the relationship between two variables which was introduced in class as:

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

To find the largest variance, we need to diagonalize our covariance matrix since we need to off-diagonals to be 0. This is when we can use the Singular Value Decomposition(SVD) which is what PCA is based on. We can set $A = \frac{1}{\sqrt{n-1}} X$. So C_x will be AA^T, then we can apply SVD where

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

after simplifying we can get:

$$C_x = U \Sigma^2 U^* \tag{3}$$

Then we want to change the basis of our data by multiplying U^{-1} which is the same as U^T:

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

The new covariance matrix will be:

$$C_y = \Sigma^2 \tag{5}$$

where Σ is the matrix of the singular values of A.

3 Algorithm Implementation and Development

The algorithm can be summarized and separated into two main parts:

For the first part, we find the coordinates(indices) of the moving mass by looking at the white dot located on the mass. For the second part, we apply PCA to plot the data.

Algorithm 1: Finding the indices of the most changed pixel to locate the mass

```
Load data from .mat files
Loop through pixels in different frames:
for $f = 1$ :number of frames do
  for $i = 1$ :x coordinates in frame do
    for $j = 1$ :y coordinates in frame do
      Look for white pixels
      if pixel is white then
        Record change in pixels and indices
      end if
    end for
  end for
  Find the index with the most change
  Store inde
end for
```

Algorithm 2: PCA using SVD

```
Normalize data set
Calculate U, S and V by using SVD
Calculate data set in new basis
```

4 Computational Results

In the ideal case, as we can see from 1, even though it doesn't look as ideal as expected, I was able to get a shape close to $A\cos(\omega t + \phi)$ which is the solution of the position of the mass. Comparing to the ideal case, the noisy case(2) seem to be a little messier, however, we can still see the same pattern. I have to increase my window size what tracks the mass due to the shaky camera, which can include unnecessary data points. And this should be the reason why the graph looks different than the ideal case, because ideally, these two will not be much different. As we can see from 3, comparing to the ideal case, the amplitude is smaller which can be indicating that the displacement in the z direction is smaller than the ideal case. From 4, we can see that the vertical displacement is almost the same as the idea case except that it bounces a little faster.

5 Summary and Conclusions

By applying what we learned in class on PCA and SVD, I was able to detect the position of the mass without applying any physics. As we can see from the results, the benefit of PCA is that it will reduce redundant data while preserving the information we need. The uncorrelated data also helped us find the change in the z direction. The overall results of 4 different cases look very similar to each other since they are all plotting the mass's harmonic movement with the spring.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation.

- `sz = size(A) returns a row vector whose elements are the lengths of the corresponding dimensions of A. For example, if A is a 3-by-4 matrix, then size(A) returns the vector [3 4].`
- `[S = sum(A) returns the sum of the elements of A along the first array dimension whose size does not equal 1.`

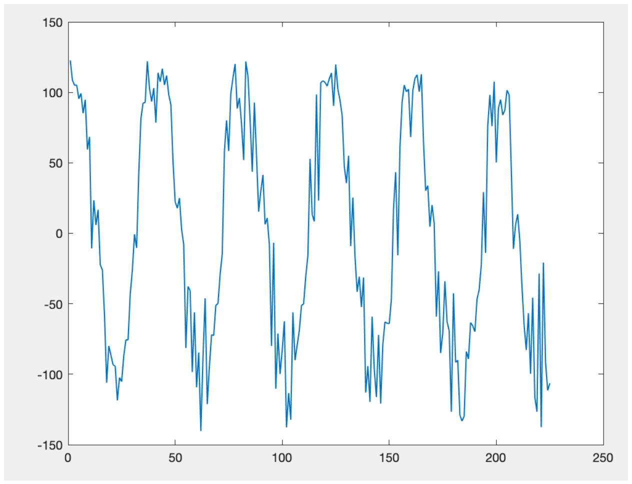

Figure 1: Ideal Case.

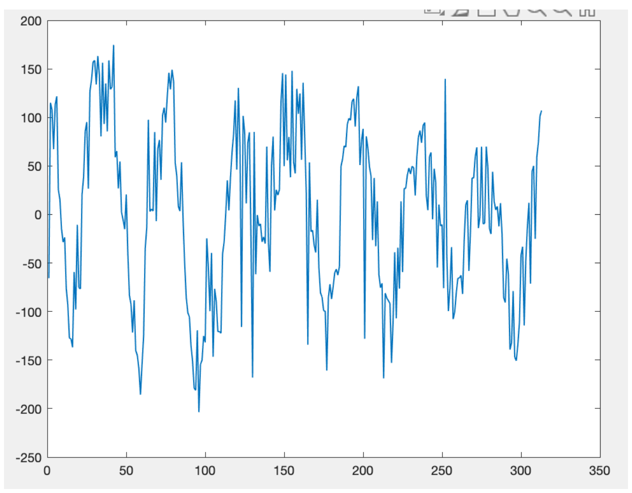

Figure 2: Noisy Case.

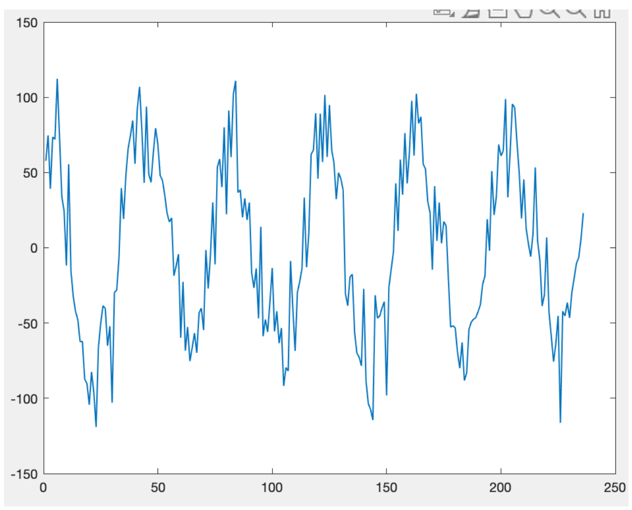

Figure 3: Horizontal Displacement Case.

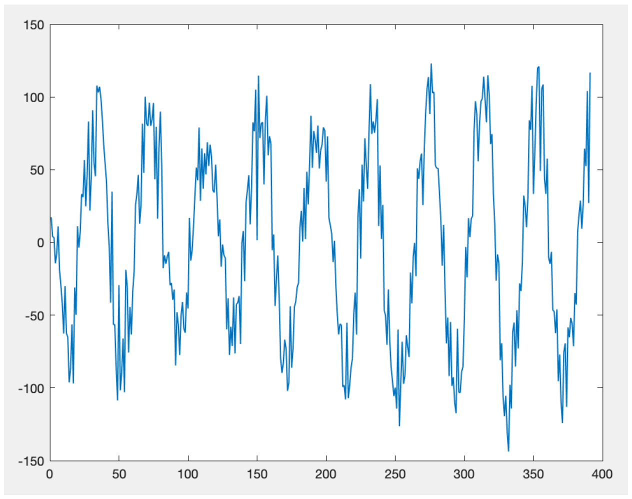

Figure 4: Horizontal Displacement with rotation Case.

- `M = mean(A)` returns the mean of the elements of A along the first array dimension whose size does not equal 1.
- `B = repmat(A,n)` returns an array containing n copies of A in the row and column dimensions. The size of B is size(A)*n when A is a matrix.
- `[U,S,V] = svd(A)` performs a singular value decomposition of matrix A, such that A = U*S*V'.
- `plot(Y)` creates a 2-D line plot of the data in Y versus the index of each value.

Appendix B MATLAB Code

```
%%Same method with all 4 cases with different window sizes only
clear all; close all;clc;
load cam1_1.mat;load cam2_1.mat;load cam3_1.mat;
ind=[];
lastF = vidFrames1_1(:,:,:,1);
[y,x,c,t]=size(vidFrames1_1);
%cam1_1
for f = 2:t
    diff=[];
    index=[];
    for i = 300:400
        for j = 200:400
            cl1 = lastF(j,i,:);
            cl2 = vidFrames1_1(j,i,:,f);
            if(all(cl2(:)>250))
                d = sum(cl2(:)-cl1(:));
                diff=[diff;d];
                index=[index;[i,j]];
            end
        end
    end
    s=find(diff==max(diff(:,1)));
    index=index(s,:);
    ind=[ind;[mean(index(:,1)),mean(index(:,2))]];
    lastF=vidFrames1_1(:,:,:,f);
end
ind=ind';
x1=ind(1,:);
y1=ind(2,:);
 % repeat for cam2_1 and cam3_1
[y,x,c,t]=size(vidFrames2_1);
ind=[];
lastF = vidFrames2_1(:,:,:,1);
for f = 2:t
    diff=[];
    index=[];
    for i = 200:350
        for j = 50:400
            cl1 = lastF(j,i,:);
            cl2 = vidFrames2_1(j,i,:,f);
            if(all(cl2(:)>245))
                d = sum(cl2(:)-cl1(:));
                diff=[diff;d];
                index=[index;[i,j]];
            end
        end
    end
    s=find(diff==max(diff(:,1)));
    index=index(s,:);
    ind=[ind;[mean(index(:,1)),mean(index(:,2))]];
    lastF=vidFrames2_1(:,:,:,f);
end
ind=ind';
x2=ind(1,:);
y2=ind(2,:);
% cam3_1
[y,x,c,t]=size(vidFrames3_1);
lastF = vidFrames3_1(:,:,:,1);
ind=[];
 for f = 2:t
    diff=[];
    index=[];
    for i = 250:450
        for j = 200:350
            cl1 = lastF(j,i,:);
            cl2 = vidFrames3_1(j,i,:,f);
            if(all(cl2(:)>230))
                d = sum(cl2(:)-cl1(:));
                diff=[diff;d];
                index=[index;[i,j]];
            end
        end
    end
    s=find(diff==max(diff(:,1)));
    index=index(s,:);
    ind=[ind;[mean(index(:,1)),mean(index(:,2))]];
    lastF=vidFrames3_1(:,:,:,f);
 end
```

```
ind=ind';
x3=ind(1,:);
y3=ind(2,:);
 minlen=min(length(x1),length(x2));
 minlen=min(minlen,length(x3));
 result =
 ↪  [x1(:,1:minlen);y1(:,1:minlen);x2(:,1:minlen);y2(:,1:minlen);x3(:,1:minlen);y3(:,1:minlen)];
 [m,n]=size(result);
 mn=mean(result,2);
 result=result-repmat(mn,1,n);
 [U,S,V] = svd(result/sqrt(n-1));
 lambda=diag(S).^2;
 Y=U'*result;
xp=1:minlen;
plot(xp, Y(1,:), 'LineWidth', 1);
```

Code for HW3