AMATH 482 Homework 3

Tommy Dong

February 24, 2021

Abstract

In this project, one is expected to utilize PCA (Principal Component Analysis) to track an object's motion. This time, one is given videos filmed by three cameras from different positions for a spring-mass system. The goal is to use PCA to figure out the motion of the system.

1 Introduction and Overview

Add your introduction and overview here.

1.1 Data Description

There are 12 videos filmed by cameras in total, and they can be put into 4 groups: the ideal group where the motion is simple and there isn't any noise, noisy case where the videos is shaking during filming, case where horizontal movement is added and the case where both horizontal and rotation movements both exist.

1.2 Goal

For the 4 different tests, the goal is to utilize PCA analysis to find the position of the mass along with the time flow in order to tract the motion of the mass.

2 Theoretical Background

2.1 SVD decomposition

Singular Value Decomposition (SVD) decompose some matrix A into unitary matrices U and V and a diagonal matrix Σ.

$$A = U\Sigma V \tag{1}$$

Σ is the singular values of matrix A, U are the left singular vectors of A and V is the right singular vectors of A.

3 Algorithm Implementation and Development

Add your algorithm implementation and development here. See Algorithm **??** for how to include an algorithm in your document. This is how to make an *ordered* list:

1. SVD decomposition SVD decompose matrix A into three part: singular value S, and two singular vectors U and V which make A project to a certain plane.

   In our special case, we have three cameras filming the spring-mass systems, which will give us three different x,y coordinates. Since the mass only moves in one dimension, and we have 6 dimensions in total, we want to use SVD decomposition to decomposes the data from these three cameras, and then make the trajectories for three different views into one common plane. This procedure will yield us the mass motion we pursue and remove the redundancies causing by cameras from different angles. Also, by analyzing the energy of SVD, we can identify if the mass is moving in only one direction or two directions.

4 Computational Results

These four tests' mass movement along with time are demonstrated in graphs below. The blue line represent the horizontal movement, while red line is the vertical movement.

For the ideal case, the mass only move vertically up and down, and this situation is also reflected by SVD decompositions. As Figure 1 shown, the mass is moving harmonically in vertical direction, while almost no horizontal movement is found in idea case.

For the noisy case, besides from the noisy vertical movement due to the shaking cameras, there is also small amplitude of horizontal movement. However, while the noisy vertical movement still has a pattern, the horizontal displacement has no pattern or cycle. This is because the cameras are shaking randomly, without any patterns.

Then for the horizontal displacement test, the horizontal movement displayed this time has certain patterns. Though smaller than the amplitude of vertical movements, it still shows up clearly in the graph with its own wave.

For the last test, the horizontal displacement and rotation one, I expected to see some patterns for horizontal displacement, but the graph told a different story. The horizontal movement for this case is similar to the first ideal case where mass only moves in vertical directions.

4.1 Ideal Case

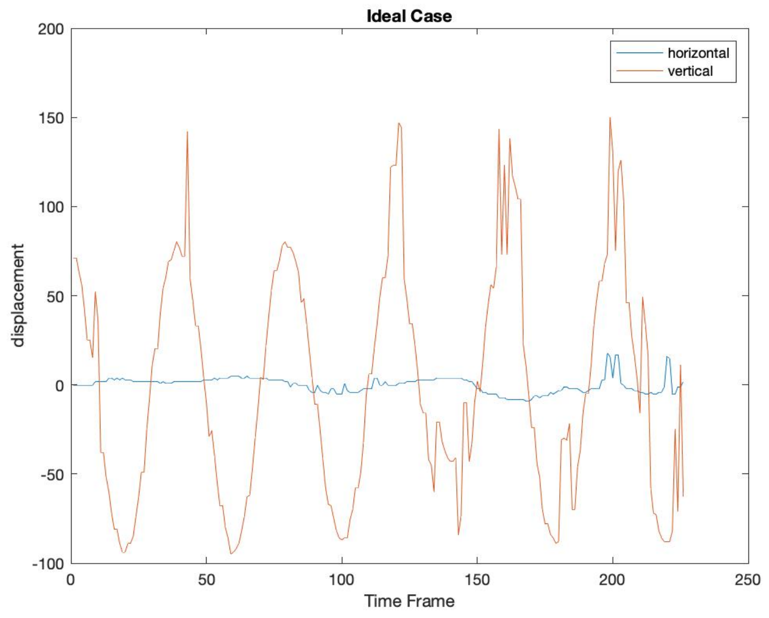

Figure 1: Here is a picture of Dubs [1]. Dubs did not swallow a marble.

4.2 Noisy Case

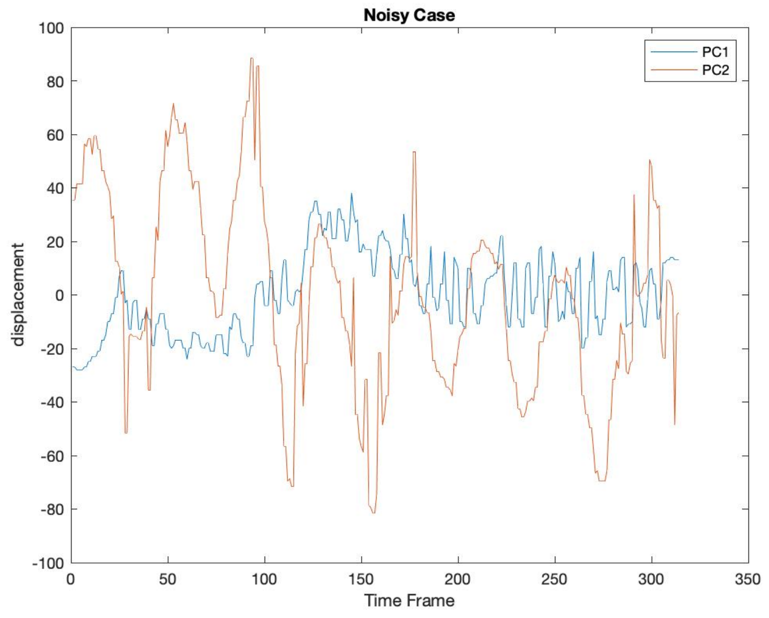

Figure 2: Here is a picture of Dubs [1]. Dubs did not swallow a marble.

4.3 Horizontal Displacement

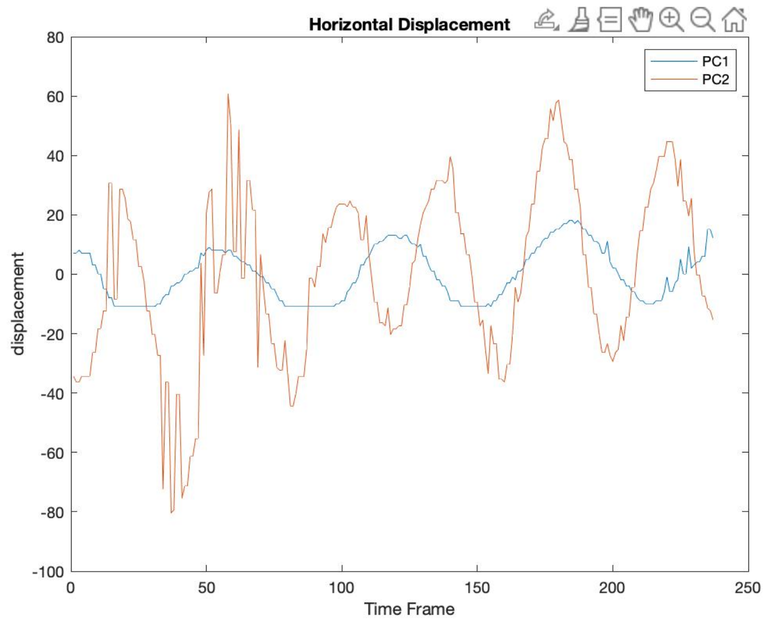

Figure 3: Here is a picture of Dubs [1]. Dubs did not swallow a marble.

4.4 Horizontal Displacement and Rotation

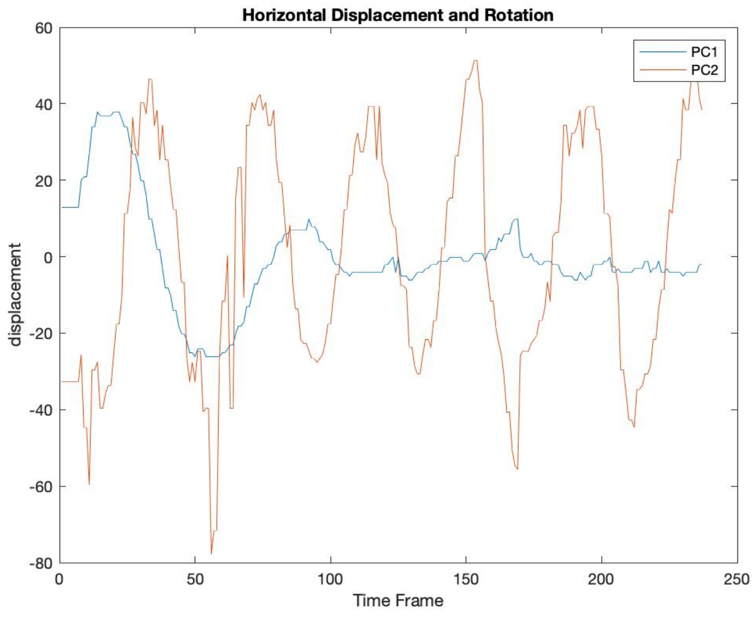

Figure 4: Here is a picture of Dubs [1]. Dubs did not swallow a marble.

5 Summary and Conclusions

As the results shown above, SVD helps us to locate the motion of mass in the spring-mass systems. From this specific project, we can see that SVD is good at ruling out redundancies, for it can track the real location of mass from cameras shooting from different angles. Also, SVD can handle some level of noisy, as it did in Case 2, it still managed to extract the motion of mass in vertical direction under the distraction of noises. IN conclusion, SVD can find the similar properties shared between different matrices, and project them on the same plane. We didn't make use of its other implementation but SVD can also help us decrease the amount of information need to be stored, and still yields the similar results as the orginal information.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- texttt[U,S,V] = svd(A,'econ') produces an economy-size decomposition of m-by-n matrix `A`:

Appendix B MATLAB Code

Add your MATLAB code here. This section will not be included in your page limit of six pages.

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

%%
load('cam1_1.mat')
% implay(vidFrames1_1)
load('cam2_1.mat')
% implay(vidFrames2_1))
load('cam3_1.mat')
% implay(vidFrames3_1)

%%
% 1_1
numFrames1_1 = size(vidFrames1_1,4);
x1_1 = zeros(1,numFrames1_1);
y1_1 = zeros(1,numFrames1_1);
for j = 1:numFrames1_1
    X1_1 = rgb2gray(vidFrames1_1(:,:,:,j));
    X1_1=im2double(X1_1);
    X1_1(:,1:300)=0;
    X1_1(:,400:end)=0;
    X1_1(1:200,:)=0;
    [M,i]=max(X1_1(:));
    [y0,x0]=ind2sub(size(X1_1),i);
    x1_1(j)=x0;
    y1_1(j)=y0;
%     imshow(X1_1);drawnow
end

%2_1
numFrames2_1 = size(vidFrames2_1,4);
x2_1 = zeros(1,numFrames2_1);
y2_1 = zeros(1,numFrames2_1);
for j = 1:numFrames2_1
    X2_1 = rgb2gray(vidFrames2_1(:,:,:,j));
    X2_1=im2double(X2_1);
    X2_1(:,1:250)=0;
    X2_1(:,350:end)=0;
    X2_1(1:80,:)=0;
    X2_1(400:end,:)=0;
    [M,i]=max(X2_1(:));
    [y0,x0]=ind2sub(size(X2_1),i);
    x2_1(j)=x0;
    y2_1(j)=y0;
%     imshow(X2_1);drawnow
end

% 3_1
numFrames3_1 = size(vidFrames3_1,4);
x3_1 = zeros(1,numFrames3_1);
y3_1 = zeros(1,numFrames3_1);
for j = 1:numFrames3_1
    X3_1 = rgb2gray(vidFrames3_1(:,:,:,j));
    X3_1=im2double(X3_1);
    X3_1(1:240,:)=0;
    X3_1(330:end,:)=0;
    X3_1(:,1:250)=0;
    X3_1(:,480:end)=0;
    [M,i]=max(X3_1(:));
```