AMATH 482 Homework 3

David Warne

February 24, 2021

Abstract

Singular Value Decomposition and Principal Component Analysis are two of the most powerful tools that professionals, and everyday people, can use within the scope of data analytics. There is a lot of information that can be discovered about large datasets through the utilization of these techniques, and that is what this assignment will primarily focus on.

1 Introduction and Overview

In this assignment, I am provided with 4 separate scenarios for a mass-spring system. In each of the scenarios, there are 3 cameras filming the same mass-spring system, and I am tasked with extracting the positional data from each of those angles, and then performing SVD/PCA to get a better understanding of what is happening within the system and why.

2 Theoretical Background

In order to better understand the algorithms implemented in this assignment, it is important to talk more about Singular Value Decomposition as well as Principal Component Analysis. From the textbook, [1], we know that the SVD "gives a type of least-square fitting algorithm, allowing us to project a matrix onto low dimensional representations in a formal, algorithmic way". This method of SVD is then utilized to figure out what a system's principal components are, which give insight about the directions in which the data has the most information. In other words, PCA helps us get a better understanding of the variance within a dataset, and allow us to visualize what parts of that data are the most meaningful and relevant.

That being said, I will be implementing SVD and PCA on the data extracted from the videos of the mass-spring system in order to get a better understanding of the variance between each camera angle in each scenario, and then comparing them to one another.

Below is the Gaussian filter that I will be using to isolate the light on top of the can that is part of the mass-spring system:

$$g(x,y) = e^{-a*(Kx^2 - +Ky^2)} \tag{1}$$

Where Kx and Ky represent the frequencies within the image that we are isolating to pinpoint the coordinates of the can to track its trajectory.

Below is the PCA method that I will be implementing:

$$V = \sigma/\sqrt{n-1} * U \tag{2}$$

Where V is the vector generated by the principal component U, σ is the corresponding singular value, and n is the number of data points (frames in this case) in the given system.

3 Algorithm Implementation and Development

There are essentially two algorithms. One that is used to extract the x and y coordinate data from each respective camera, and one that implements SVD and PCA to reveal the information that we want to know about the system. See Algorithm 1 for extracting x and y coordinates, and see Algorithm **??** for implementing SVD and PCA.

Algorithm 1: Extracting the X and Y coordinates for each frame

Load the data from the provided files that contain frame by frame pictures of the mass-spring system
Figure out which of the cameras has the shortest amount of frames and set that as the largest possible frames for each of the cameras to reach so that the data is consistent between matrices.
for j = 1:numFrames **do**
  Parse the camera data to extract a frame
  Convert the image to gray scale and make sure that the data type is set to double.
  Set up and apply a gaussian filter on the data
  Utilize the max command to find the brightest pixel (which corresponds to the light on the can in the mass-spring system.
  Using the index of that max value, use ind2sub to convert those indices to x and y coordinates, and store those coordinates in the empty vectors that are being used to track the coordinates for each iteration.
end for
Repeat for each camera.
Compile the x and y vectors into a matrix that can be analyzed with SVD and PCA.

Algorithm 2: Applying SVD and PCA to the compiled Matrix

Use MATLAB's built in SVD function on the compiled matrix
Once U, S, and V have been calculated by MATLAB, utilize equation 2 to determine the vector that is produced by each principal component
Plot the data to determine patterns and gain insight on the behavior of the system according to the PCA.

4 Computational Results

- Unfortunately, I was unable to successfully figure out how to extract the x and y coordinates from the provided data. I have attached my MATLAB code that shows where my thoughts lied, and the information in the algorithms describes my knowledge of what I believe to be the correct way to approach the rest of the assignment.

5 Summary and Conclusions

In conclusion, SVD and PCA are extremely powerful tools that allow us to get a better idea of systems that have multiple dimensions. Despite the fact that we were given data in 4 dimensions, we were able to apply SVD and PCA to gain insight that other processes could not have provided.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

Here are the main functions that I used in this assignment:

- `[U,S,V] = svd()` allows us to compute singular value decomposition on our matrix of x and y components.

Appendix B MATLAB Code

Here is the MATLAB code that was used to produce the information in this assignment.

```
%% Assignment 3 Script
clc; clear all; close all;

%%
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')

%% vidFrames1_1
numFrames = size(vidFrames1_1,4);

xs = size(vidFrames1_1, 1);
ys = size(vidFrames1_1, 2);
xcoords1 = zeros(1, xs);
ycoords1 = zeros(1, ys);

for j = 1:numFrames

    X = vidFrames1_1(:,:,:,j);
    I = rgb2gray(X);
    I = im2double(I);

    % use gaussian filter
    [Ny,Nx] = size(I);
    kx = [0:Nx/2-1 -Nx/2:-1];
    ky = [0:Ny/2-1 -Ny/2:-1];
    [Kx,Ky] = meshgrid(kx,ky);

    a = 1e-3;
    filter = exp(-a*(Kx.^2 + Ky.^2));

    If = I.*filter;

    [M, I] = max(abs(If),[],'all', 'linear');
    [x_coordinate, y_coordinate] = ind2sub([size(I,1) size(I, 2)], I);
    xcoords1(j) = x_coordinate;
    ycoords1(j) = y_coordinate;
end

%% vidFrames2_1

numFrames = size(vidFrames2_1,4);

xs = size(vidFrames2_1, 1);
ys = size(vidFrames2_1, 2);
xcoords2 = zeros(1, xs);
ycoords2 = zeros(1, ys);

for j = 1:numFrames

    X = vidFrames2_1(:,:,:,j);
    I = rgb2gray(X);
    I = im2double(I);

    % use gaussian filter
    [Ny,Nx] = size(I);
    kx = [0:Nx/2-1 -Nx/2:-1];
    ky = [0:Ny/2-1 -Ny/2:-1];
    [Kx,Ky] = meshgrid(kx,ky);
```