AMATH 482 Homework 3

Erik Huang

February 24, 2020

Abstract

Experiment Principal Component Analysis (PCA) on different videos showing the motion of a spring-mass system. Analyze its practical usefulness in each case, and the effects of noise on the PCA algorithms.

1 Introduction and Overview

This assignment is about the usage of Principal Component Analysis (PCA). We are provided with videos of a paint can in the spring-mass system, which involves a mass hanging from a spring oscillating up and down. There are a total of 4 cases: The ideal case, Noisy case, Off-centre case, Off-centre rotation case. These cases allow us to test PCA on different scenarios.

For each case, there are 3 video files shot at different angles. The reason is that we don't know the dimension of the motion in spring-mass system, so we would need more data to analyze and confirm our presumption. We would expect the PCA algorithm to show the redundancy of the data and outputs one relatively large singular value which represents the 1-dimensional motion in the system.

2 Theoretical Background

2.1 Singular Value Decomposition(SVD)

The concept of Singular Value Decomposition (SVD) is the foundation of our analysis. The SVD is a factorization of a matrix in linear algebra, and one of the most powerful and versatile tool in data analysis. The singular value decomposition of a $m \times n$ matrix A is:

$$\mathbf{A} = \mathbf{U\Sigma V}^*$$

$\mathbf{U}$ is a $m \times n$ unitary matrix with orthonormal columns. $\mathbf{\Sigma}$ is a $m \times n$ rectangular diagonal matrix with non-negative real numbers on the diagonal. $\mathbf{V}$ is a $m \times n$ complex unitary matrix. $\mathbf{U}$ and $\mathbf{V}$ contain information on rotation and $\mathbf{\Sigma}$ describes the stretch in each direction. In application, however, we can't always expect to perform full SVD on data. To make this technique more flexible, we can use the reduced SVD for the general cases:

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

Typically, we can construct $\mathbf{U}$ by adding additional $m - n$ columns that are orthogonal to the existing $\hat{\mathbf{U}}$. We can also add additional $m - n$ rows of zeros to $\hat{\mathbf{\Sigma}}$ to get $\mathbf{\Sigma}$.

In addition, the SVD has the following properties:

- Every matrix $\mathbf{A} \in \mathbb{C}^{m \times n}$ has an SVD
- The singular values are uniquely determined and are always non-negative real numbers.
- The singular values are ordered from largest to smallest, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$, along diagonal of Σ
- The number of nonzero singular values is the rank of $\mathbf{A}$.

- Letting $r = \text{rank}(\mathbf{A})$ we have that $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$ is a basis for the range of $\mathbf{A}$ and $\{\mathbf{v}_r + 1, \ldots, \mathbf{v}_n\}$ is a basis for the null space of A.
- MATLAB actually calculates the rank and a basis for the null space using the SVD when you type in the rank() or null() commands.

2.2 Low-Dimensional Approximations

We also need to know the theorem regarding Low-Dimensional Approximations to understand why SVD is so useful to the analysis.

Theorem 1 *If* $\mathbf{A}$ *is a matrix of rank* r*, then* $\mathbf{A}$ *is the sum of* r *rank 1 matrices:*

$$\mathbf{A} = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^*$$

Theorem 2 *For any* N *so that* $0 < N < r$*, we can define the partial sum*

$$\mathbf{A}_N = \sum_{j=1}^{N} \sigma_j \mathbf{u}_j \mathbf{v}_j^*$$

The above two theorems infer that out of all the matrices that are rank N or less, the best approximation of $\mathbf{A}$ (relative to the two different norms) is given by $\mathbf{A_N}$. When applied to this assignment, our data from the video can be represented as value of each pixels in a gray scale frame. This concept would mean that we can represent the information from each frame without having to store each pixel. The technique of SVD do this task and preserve the trends or correlations within the data.

2.3 Principal Component Analysis(PCA)

The Principal Component Analysis(PCA) is an application of SVD. The results from SVD would indicate the dimensions and directions of principal components from matrix $\mathbf{A}$. By calculating the variance on the data of principal components:

$$\text{variance } = \sigma^2 = \frac{1}{n} \sum_{k=1}^{n} (a_k - \mu)^2$$

we can obtain the total variance. Using the following formula of energy, we can analyze how much of the total variance is explained by certain groups of principal components:

$$\text{energy } = \frac{\sigma_1^2 + \cdots + \sigma_n^2}{\sigma_1^2 + \cdots + \sigma_r^2}$$

3 Algorithm Implementation and Development

In this assignment, there are a total of 4 cases that we need to analyze, but the process on each case is almost identical. Therefore the following method is the algorithm applied on all cases. The tracing of the painting generates all the positioning data. Thanks to the flashlight on the top of the can, we can convert each frame into a gray scale image and locate the brightest spot, which will be the location of the painting can.

1. Load data files and store in a 4-dimensional matrix.
2. Make a filter that crops out desired area from the video. So that we can eliminate the effect of unwanted reflections.
3. Use an iteration to go through each frame. Convert into a gray scale image and change value type to double.

4. During the iteration, apply filter to each frame and find the indices of the brightest spots. Use ind2sub() to convert the indices into XY coordinates. To avoid noise, we then take the average value of these coordinates.

5. After obtaining the data from all 3 cam angles, we then need to synchronize the frames by setting the starting frame as the lowest point the painting can reaches in the beginning of the video. After synchronization, we also need to trim all 3 vectors into the same size in order to combine them into a single matrix.

6. Since videos are shot at different angles, we need to subtract the matrix by the mean value to ensure all 3 camera files have the same scale.

Finally, running SVD on the data matrix gives us the principal components. We can then perform PCA Analysis and plot the result.

4 Computational Results

Figure 1, 2, 3, 4 demonstrates the displacements of the painting can in Z direction and XY-plane. With the presumption that motion in Z direction is dominating, I plotted the first principal component to compare with the motion in Z direction from all 3 camera angles for all cases.

4.1 Case 1 - Ideal Case

The calculation of energies shows that 93.56% of the data is captured by one dimension. This is exactly what we expect, only up-down motion in the ideal spring-mass system.

4.2 Case 2 - Noisy Case

The calculation of energies shows that 58.17% of the data is captured by one dimension. This result is significantly lowered than the previous case. We can also see of effect of the shaking of the camera on the PCA plot. The curves are very skewed.

4.3 Case 3 - Horizontal Displacement

The calculation of energies shows that only 42.90% of the data is captured by one dimension. This result implies that the first principal component cannot capture the trend of the data. Besides the surprisingly low energy value, I also have trouble synchronizing the videos due the violent horizontal displacement. From the PCA plot, we can see that there's sinusoidal movements in the XY-plane as well.

4.4 Case 4 - Horizontal Displacement and Rotation

The calculation of energies shows that 80.37% of the data is captured by one dimension. This result is better than the one in the previous case. I suppose that the motion of painting can is less violent compare to the one in case 3. On both PCA plot and the energy plot, we can see that the first principal component can capture the majority of the data.

5 Summary and Conclusions

The PCA is a super powerful tool for us to perform analysis on principal components of the data. It also provides insights on the relevance of each dimension. In the situation covered in this assignment, the first principal component can indicate the direction of the motion of the painting can. However, by exploring different cases, we can also see that PCA analysis can be easily disturbed by noisy data. When the data does not have dominant trends, the energy of each dimension is relatively lowered. In conclusion, the overwhelming amount of data captured by one dimension in most cases can confirm our idea that the spring-mass system involves a one-dimensional up and down motion.

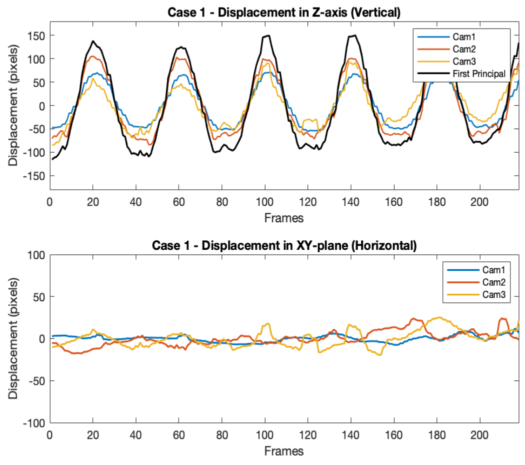

Figure 1: PCA on Test 1: Ideal Case

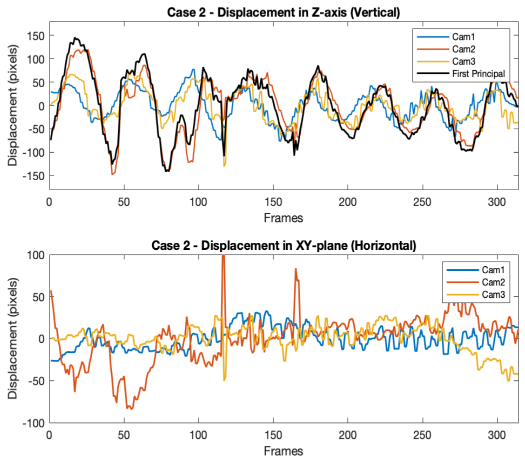

Figure 2: PCA on Test 2: Noisy Case

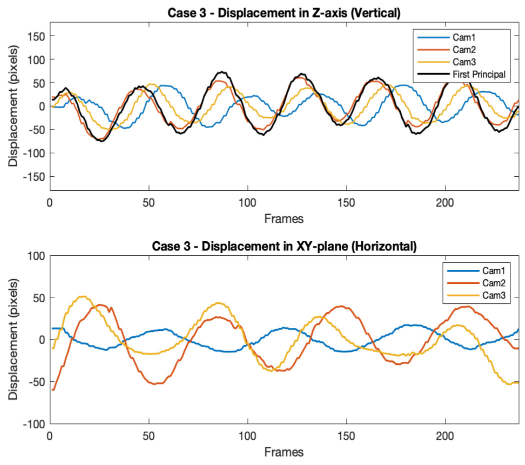

Figure 3: PCA on Test 3: Horizontal Displacement

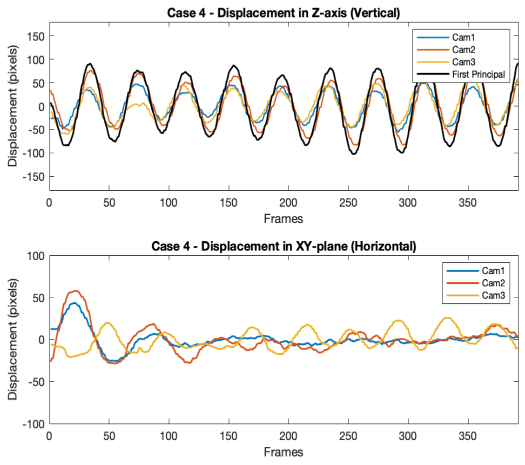

Figure 4: PCA on Test 4: Horizontal Displacement and Rotation

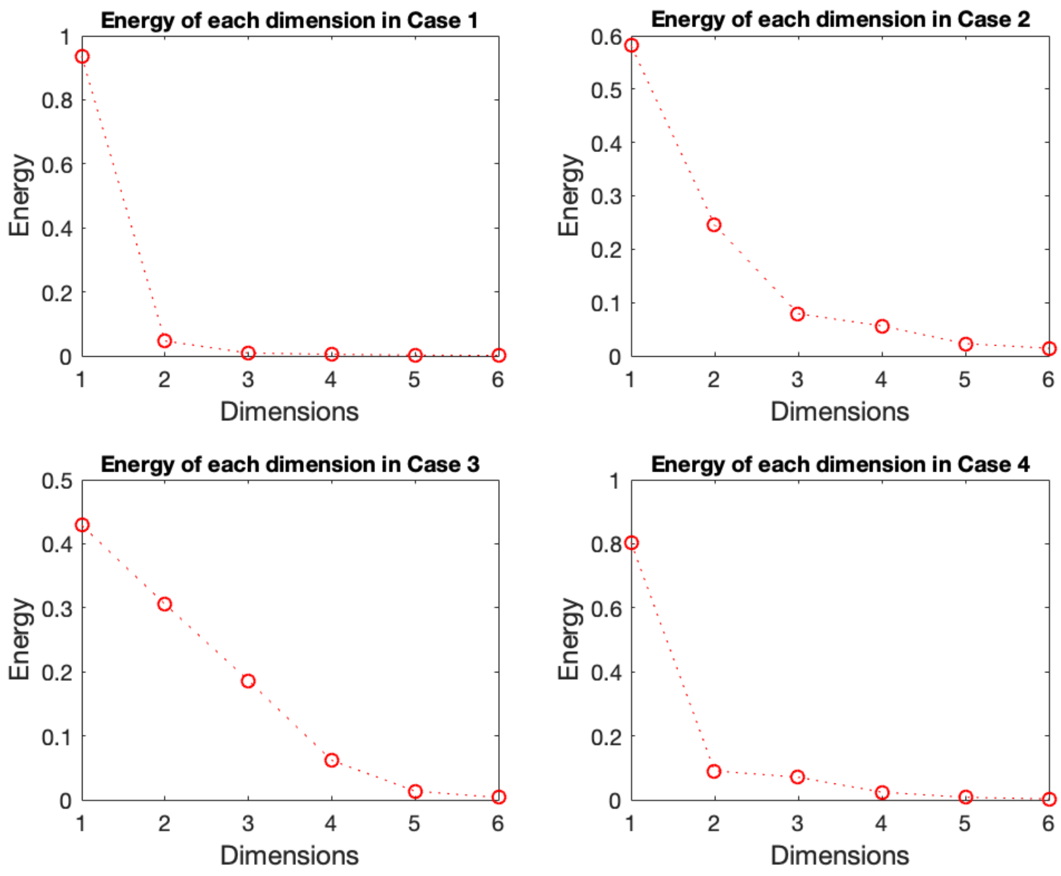

Figure 5: Summary graph of energies of all 4 cases

Appendix A MATLAB Functions

unordered list:

- `implay` Play movies, videos, or image sequences
- `imshow` Shows a frame of the video data
- `rgb2gray` Convert RGB image or colormap to grayscale
- `ind2sub` Convert linear indices to subscripts
- `[U,S,V] = svd(A,'econ')` Singular value decomposition. Produces an economy-size decomposition of m-by-n matrix A

Appendix B MATLAB Code

```
close all; clear all; clc;
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')

%% Test videos
implay(vidFrames1_1);
implay(vidFrames2_1)
implay(vidFrames3_1)

%% Calculate size of frames
numFrames1_1 = size(vidFrames1_1,4);
numFrames2_1 = size(vidFrames2_1,4);
numFrames3_1 = size(vidFrames3_1,4);

%% Cam 1_1
filter = zeros(480,640);
filter(150:450, 300:400) = 1;
cam_1_1_res = zeros(numFrames1_1, 2);

for i = 1:numFrames1_1
    frame = vidFrames1_1(:,:,:,i);
    frame = rgb2gray(frame);
    frame = double(frame);
    frame = frame .* filter;

    pink = frame > 250;
    Index = find(pink);
    [y, x] = ind2sub(size(pink), Index);
    cam_1_1_res(i,:) = [mean(x), mean(y)];
    % imshow(pink); drawnow
end

%% Cam 2_1
filter = zeros(480,640);
filter(100:450, 250:350) = 1;
cam_2_1_res = zeros(numFrames2_1, 2);

for i = 1:numFrames2_1
    frame = vidFrames2_1(:,:,:,i);
    frame = rgb2gray(frame);
    frame = double(frame);
    frame = frame .* filter;

    pink = frame > 250;
    Index = find(pink);
    [y, x] = ind2sub(size(pink), Index);
    cam_2_1_res(i,:) = [mean(x), mean(y)];
    % imshow(pink); drawnow
end

%% Cam 3_1
filter = zeros(480,640);
filter(180:340, 220:520) = 1;
cam_3_1_res = zeros(numFrames3_1, 2);

for i = 1:numFrames3_1
    frame = vidFrames3_1(:,:,:,i);
    frame = rgb2gray(frame);
    frame = double(frame);
    frame = frame .* filter;

    pink = frame > 245;
    Index = find(pink);
    [y, x] = ind2sub(size(pink), Index);
    cam_3_1_res(i,:) = [mean(x), mean(y)];
    imshow(pink); drawnow
end

%% RESULT: Sync all video frames

[minimum, i] = min(cam_1_1_res(1:20, 2));
cam_1_1_trim = cam_1_1_res(i:end, :);
[minimum, i] = min(cam_2_1_res(1:20, 2));
cam_2_1_trim = cam_2_1_res(i:end, :);
[minimum, i] = min(cam_3_1_res(1:20, 1));
cam_3_1_trim = cam_3_1_res(i:end, :);

max_size = min([length(cam_1_1_trim(:,1)) length(cam_2_1_trim(:,1)) length(cam_3_1_trim(:,1)
    )]);
max_frame = 1:max_size;
cam_1_1_trim = cam_1_1_trim(1:max_size, :);
cam_2_1_trim = cam_2_1_trim(1:max_size, :);
cam_3_1_trim = cam_3_1_trim(1:max_size, :);

%%


X = [cam_1_1_trim'; cam_2_1_trim'; cam_3_1_trim'];
pos_mean = mean(X, 2);
X_centered = X - pos_mean;
% X_centered = X - repmat(pos_mean, 1, max_size);

[u,s,v] = svd(X_centered ./ sqrt(max_size - 1), 'econ');
%[u,s,v] = svd(((X_centered ./ sqrt(max_size - 1))));
lambda = diag(s).^2;
Y = u' * X_centered;

save('case_1_energy.mat', 'lambda');

%PCA Plot
figure(1)
subplot(2,1,1)
plot(max_frame, X_centered(2,:), max_frame, X_centered(4,:), max_frame, X_centered(5,:), '
    LineWidth', 1.2)
hold on
plot(max_frame, Y(1,:), 'k','LineWidth', 1.5)
axis([0 max_size -180 180])
ylabel("Displacement (pixels)"); xlabel("Frames");
title("Case 1 - Displacement in Z-axis (Vertical)");
legend('Cam1', 'Cam2', 'Cam3', 'First Principal', 'Fontsize', 5)
subplot(2,1,2)
plot(max_frame, X_centered(1,:), max_frame, X_centered(3,:), max_frame, X_centered(6,:), '
    LineWidth', 1.5)
axis([0 max_size -100 100])
ylabel("Displacement (pixels)"); xlabel("Frames");
```

```
title("Case 1 - Displacement in XY-plane (Horizontal)");
legend('Cam1', 'Cam2', 'Cam3', 'Fontsize', 5)
%Energy Plot
figure(2)
plot(1:1:6,lambda/sum(lambda),'r:o')
title('Energy of each dimension in Case 1','Fontsize',16)
ylabel('Energy','Fontsize',16)
xlabel('Dimensions','Fontsize',16)
```