Homework 3: PCA

Alyssa Jin

02/20/2020

Abstract

In this assignment, we illustrate various aspects of the Principal Component Analysis (PCA) and its practical usefulness and the effects of noise on the PCA in a spring-mass experiment by bucket. We observe how PCA can reduce redundancy in a spring-mass system on four different tests, which are ideal case, noisy case, horizontal displacement, and horizontal displacement and rotation.

I. Introduction and Overview

The objective of this assignment is to practice Principle Component Analysis (PCA) with spring-mass system. The assignment is based on 12 videos, which are consists of 4 different tests and 3 different camera from angles. We can use PCA to show the 6 dimensions video to only one dimension. By PCA, we can store the measurement in a vector X that contains six different measurements, A, B, C represent different cameras:

$$X = \begin{bmatrix} \overrightarrow{xA} \\ \overrightarrow{yA} \\ \overrightarrow{xB} \\ \overrightarrow{yB} \\ \overrightarrow{xC} \\ \overrightarrow{yC} \end{bmatrix}$$

The 4 different tests are:
· **(test 1) Ideal case:** Add a small displacement of the mass in the z direction. The entire motion is in the z direction with simple harmonic motion.
· **(test 2) Noisy case:** Repeat the ideal case experiment, and introduce camera shake into the video recording.
· **(test 3) Horizontal displacement:** The mass is released off-center so as to produce motion in the $x - y$ plane as well as the z direction. There is both a pendulum motion and simple harmonic oscillations.
· **(test 4) Horizontal displacement and rotation:** The mass is released off-center and rotates so as to produce motion in the $x - y$ plane, and motion in the z direction.

II. Theoretical Background

II.i. Singular Value Decomposition (SVD)

The vector **x** when multiplied by a matrix **A** produces a new vector **y** is aligned in a new

direction with a new length. Generically, matrix multiplication will rotate and stretch (compress) a given vector as prescribed by the matrix $\mathbf{A}$. A singular value decomposition (SVD) is a factorization of a matrix into a number of constitutive components all of which have a specific meaning in applications. In compact matrix notation, $\mathbf{AV} = \widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}$. Therefore the SVD of $\mathbf{A}$ is $\widehat{\mathbf{U}}\widehat{\mathbf{\Sigma}}\mathbf{V}^*$: $\mathbf{U} \in \mathbb{C}^{m\times m}$ is unitary, $\mathbf{V} \in \mathbb{C}^{n\times n}$ is unitary, $\mathbf{\Sigma} \in \mathbb{C}^{m\times n}$ is diagonal with non-negative elements and ordered from largest to smallest from element in left-top to element in right-bottom.
Theorem: Every matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ has a singular value decomposition. Furthermore, the singular values $\{\sigma_j\}$ are uniquely determined, and, if $\mathbf{A}$ is square and the $\{\sigma_j\}$ distinct, the singular vectors $\{\mathbf{u}_j\}$ and $\{\mathbf{v}_j\}$ are uniquely determined up to complex signs.

II.ii. Principal Component Analysis (PCA)

To identify redundant data, we need tp find the covariance between data sets. Covariance measures the statistical dependence/independence between two variables. Strongly statistically dependent variables can be considered as redundant observations of the system.
For example, consider two sets of measurements with zero means expressed in row vector form: $\mathbf{a} = [a_1\ a_2\ ...a_n]$ and $\mathbf{b} = [b_1\ b_2\ ...b_n]$. The variance of $\mathbf{a}$ and$\mathbf{b}$ are $\sigma_a^2 = \frac{1}{n-1}\mathbf{a}\mathbf{a}^T$, $\sigma_b^2 = \frac{1}{n-1}\mathbf{b}\mathbf{b}^T$, and $\sigma_{ab}^2 = \frac{1}{n-1}\mathbf{a}\mathbf{b}^T$. The appropriate covariance matrix is $\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T$, which is an $m \times m$ matrix.
The goal of PCA is removing redundancy and identifying those signals with maximal variance.

III. Algorithm Implementation and Development

We need to use two scripts. The first script is to find the whitest point of the flashlight on bucket, and then track the movement of the point. The second script is to use SVD and PCA to analysis and draw the movement.

III.i. Find the location of the bucket through movement

· starting point of the movement
To have a precise analysis of the movement, we need to focus on a specific point in the video, so I choose the lightest area at the top of the bucket as the first point for the first three tests. Since there is not an obvious area in the bucket that is lighter than other area, so I choose the middle of the top of the bucket.
· Track the movement of the bucket
To find the movement of the bucket, I set a small square around the lightest area, which inside the frame of the video, and contains the lightest area throughout the whole process, but not including any other light area that may affect the consequence of the test.
· Save the data
To analyze the position of the movement by the second script, I need to save all the data.

III.ii. Principal Component Analysis

In this part, I will use SVD and PCA to analyze the data.
· Adjust the time of the experiments

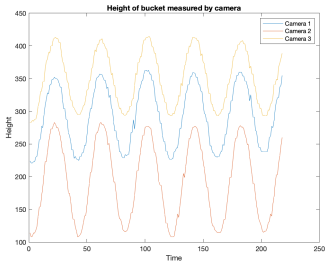

FIGURE 1. Height of bucket measured by camera in test 1

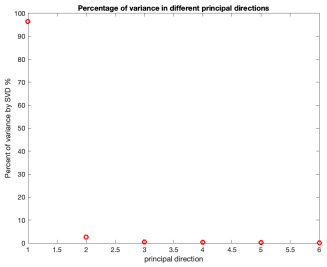

FIGURE 2. Percentage of variance in test 1

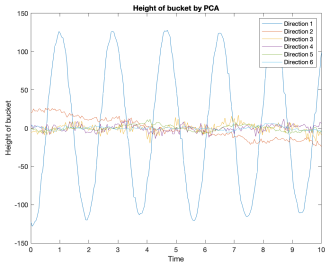

FIGURE 3. Height of bucket by PCA in test 1

To draw the data of same tests in the same picture, I need to adjust the time of three different angles to keep them the same, which means I need to cut the time of the two angles from the beginning.

· **Adjust the position vectors**

Since I need to analyze the data by SVD, the vectors of position should have the same length, so I set the time to be from 0 to 10.

· **SVD and PCA**

Calculate mean and variance of X, then use SVD and PCA to analyze it.

IV. Computational Results

Figure 1, 4, 7, 10 show the height of the bucket recording by three different cameras in 4 different tests. Figure 2, 5, 8, 11 show the variance of six different directions in 4 different tests. Figure 3,6,9, 12 show six principal directions of the data in 4 different tests.

IV.i. (test 1) Ideal case: Figure 1, 2, 3

Test 1 is just a simple harmonic motion.

In figure 1, although the height of the three angles are different, the wave of the three curves are almost the same.

Figure 2, the first plot shows more than 95% of variance. The second plot is about 5%. From the third plot to the sixth plot are all very low, which are almost to 0%.

In figure 3, only the first (blue) curve is like a cosine oscillation. The second (pink) curve affected by noise at the beginning, then decrease to 0, which shows that the amplitude of oscillating mass decreases to zero after some time. The third to sixth curves vibrate little around 0, which are affected very little by noise.

IV.ii. (test 2) Noisy case: Figure 4, 5, 6

Test 2 add camera shake to test 1, which means the oscillating mass will be affected by noise.

In figure 4, The wave of the first (blue) and third (yellow) curves are almost the same. At the beginning, the second (red) curve also has the same wave as the other two, but dropped to the bottom rapidly after very short time.

Figure 5, the first plot shows about 68% of variance, the second and third plots are about

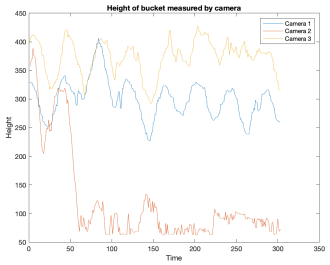

FIGURE 4. Height of bucket measured by camera in test 2

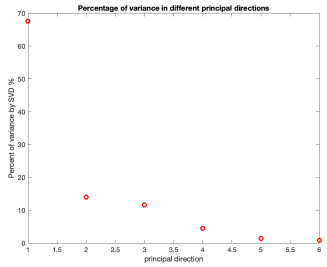

FIGURE 5. Percentage of variance in test 2

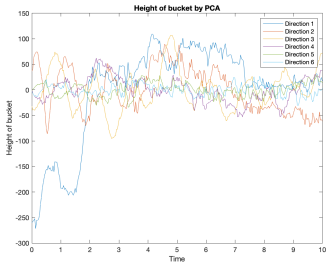

FIGURE 6. Height of bucket by PCA in test 2

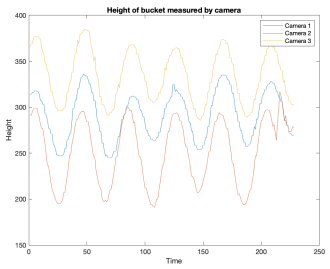

FIGURE 7. Height of bucket measured by camera in test 3

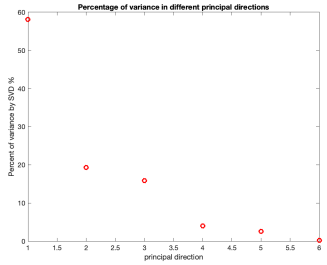

FIGURE 8. Percentage of variance in test 3

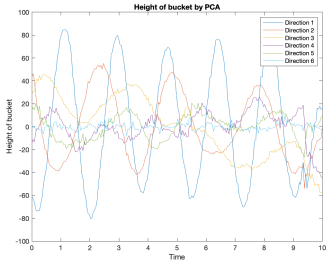

FIGURE 9. Height of bucket by PCA in test 3

13%, the fourth plot is 5%, the fifth and sixth plots are almost to 0%. The differences between adjacent two points are less than test 1. Therefore, SVD works well in noisy case. The curves in figure 6 are very mess, especially the first curve. Therefore, PCA cannot overcome the noise given by the camera.

IV.iii. (test 3) Horizontal displacement: Figure 7, 8, 9

The mass is moved not only the in the $x-y$ plane, as well as the z direction.

In figure 7, although the height of the three angles are different, the wave of the three curves are almost the same, but not as smooth as in figure 1, which is affected by the motion in z direction.

In figure 8, the first plot shows about 58% of variance, the second is about 20%, the third is about 17%, the fourth plot is 5%, the fifth plot is 4%, and sixth plot is at 0%. The slope of decrease of these 6 points are slow, especially the slope between second and third points, and slope between fourth and fifth points. Therefore, SVD doesn't work well in this case.

In figure 9, the first three curves show good sinusoidal motion, but the other three curves are noisy. Therefore, PCA only works well for part of the directions of the data.

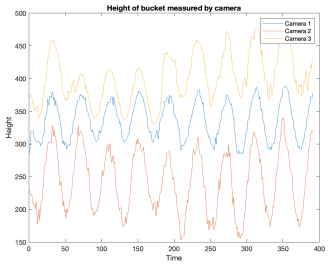

FIGURE 10. Height of bucket measured by camera in test 4

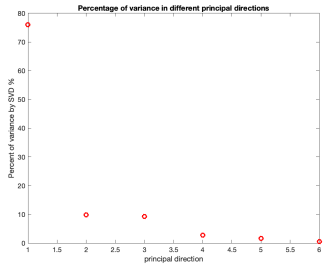

FIGURE 11. Percentage of variance in test 4

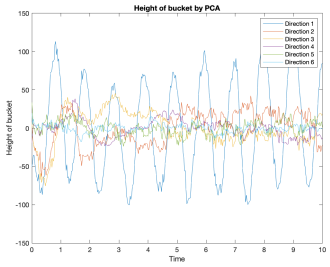

FIGURE 12. Height of bucket by PCA in test 4

IV.iii. (test 4) Horizontal displacement and rotation: Figure 10, 11, 12
The mass is moved in $x-y$ plane and z direction as well as rotates, which leads to difficult in track the flashlight.

In figure 10, although the height of the three angles are different, the wave of the three curves are almost the same. Since I cannot tell the exact point of the flashlight, so it may lead to some vibration in the curve.
In figure 11, the first plot shows about 76% of variance, the second and the third points are about 10%, the fourth plot is 4%, the fifth plot is almost to 0%, and sixth plot is at 0%. The second point drops rapidly. Therfore, SVD works well in test 4.
In figure 12, only the first (blue) curve is like a sine oscillation. The others all looks noisy. Therefore, PCA doesn't work well in test 4.

V. Summary and Conclusions

In this assignment, I use SVD and PCA to track the movement of the bucket in 4 different tests and 3 different angles. I disposal the data, use PCA to remove redundancy and identify those signals with maximal variance. PCA doesn't work well in test 2 and test 4, but works well in test 3. Therefore, I can only get useful information from test 3.

Reference

1. Kutz, J. N. (n.d.). Data Driven Modeling & Scientific Computation
2. Makers of MATLAB and Simulink. (n.d.). Retrieved February 21, 2020, from *https* : *//www.mathworks.com/*

Appendix A. MATLAB functions used and brief implementation explanation

1. $load(filename)$: loads data from filename. If filename is a MAT-file, then loads variables in the MAT-File into the MATLAB workspace.
2. $switch, case, otherwise$: evaluates an expression and chooses to execute one of several groups of statements. Each choice is a case.
3. $X = zeros(sz1, ..., szN)$: returns an sz1-by-...-by-szN array of zeros where sz1,...,szN indicate the size of each dimension.
4. $size(vidFrames)$: read the size of a video.
5. $figure(f)$: makes the figure specified by f the current figure and displays it on top of all other figures.
6. $imshow(filename)$: displays the image stored in the graphics file specified by filename.
7. $[x, y] = ginput(n)$: allows you to identify the coordinates of n points. To choose a point, move your cursor to the desired location and press either a mouse button or a key on the keyboard. Press the Return key to stop before all n points are selected. MATLAB returns the coordinates of your selected points.
8. $[row, col] = find(\_\_)$: returns the row and column subscripts of each nonzero element in array X using any of the input arguments in previous syntaxes.
9. $linspace(x1, x2, n)$: Return n points. The spacing between the points is $(x2-x1)/(n-1)$.
10. $B = repmat(A, r)$: specifies the repetition scheme with row vector r. For example, $repmat(A, [23])$ returns the same result as $repmat(A, 2, 3)$.
11. $[U, S, V] = svd(A)$: performs a singular value decomposition of matrix A, such that $A = U * S * V'$.
12. $x = diag(A)$: returns a column vector of the main diagonal elements of A.

Appendix B. MATLAB codes

Code I: store data

```
% N is the test number
% Test 1: Ideal signal
% Test 2: Noisy signal
% Test 3: Horizontal displacement
% Test 4: Horizontal displacement and rotation
N = 4;

% Load three video data of the same test
load(['cam1_' num2str(N) '.mat']);
load(['cam2_' num2str(N) '.mat'])
load(['cam3_' num2str(N) '.mat'])

% Use the same name for different tests
switch N
    case 1
        vidFrames1 = vidFrames1_1;
```

```
        vidFrames2 = vidFrames2_1;
        vidFrames3 = vidFrames3_1;
    case 2
        vidFrames1 = vidFrames1_2;
        vidFrames2 = vidFrames2_2;
        vidFrames3 = vidFrames3_2;
    case 3
        vidFrames1 = vidFrames1_3;
        vidFrames2 = vidFrames2_3;
        vidFrames3 = vidFrames3_3;
    otherwise
        vidFrames1 = vidFrames1_4;
        vidFrames2 = vidFrames2_4;
        vidFrames3 = vidFrames3_4;
end

% Find and click the first position of the flashlight

% Use the fourth element of size to find the lenghth of the video, and
% create a new vector
position1 = zeros(2,size(vidFrames1,4));
figure();
imshow(vidFrames1(:,:,:,1));
title('Click the whitest part of flashlight');
% Use ginput to click one point
[x1, y1] = ginput(1);
position1(:,1) = [y1; x1];

position2 = zeros(2,size(vidFrames2,4));
figure();
imshow(vidFrames2(:,:,:,1));
title('Click the whitest part of flashlight');
[x2, y2] = ginput(1);
position2(:,1) = [y2; x2];

position3 = zeros(2,size(vidFrames3,4));
figure()
imshow(vidFrames3(:,:,:,1));
title('Click the whitest part of flashlight');
[x3, y3] = ginput(1);
position3(:,1) = [y3; x3];

close all

% Set length and width of the rectangle that may contains the movement of
% flashlight throughout the process
```

```
% Camera 1
% Set the length
lengthsize1 = 20;
%Set the width
widthsize1 = 20;

% Camera 2
lengthsize2 = 20;
widthsize2 = 20;

% Camera 3
lengthsize3 = 20;
widthsize3 = 20;

% Set the size that may extend the video
lengthsizeextend = 0;
widthsizeextend = 0;

% Affirm rectangles that contain the movement of the whitest part of
% flashlight

% Camera 1
for l=2:size(vidFrames1,4)
    % Previous length position of the whitest part of flashlight
    length1 = position1(1,l-1);
    % Previous width position of the whitest part of flashlight
    width1 = position1(2,l-1);

    % Adjust the rectangle, so it won't extend the video
    if (length1 - lengthsize1) < 1
        lengthsizeextend = length1-1 + (length1==1);
    elseif (length1 + lengthsize1) > size(vidFrames1,1)
        lengthsizeextend = size(vidFrames1,1) - length1 + (length1 == size(vidFrames1,1)
    else
        lengthsizeextend = lengthsize1;
    end
    if (width1 - widthsize1) < 1
        widthsizeextend = width1-1 + (width1==1);
    elseif (width1 + widthsize1) > size(vidFrames1,1)
        widthsizeextend = size(vidFrames1,1) - width1 + (width1 == size(vidFrames1,1));
    else
        widthsizeextend = widthsize1;
    end
```

```
    % Find the location of previous bucket in the rectangle
    local1 = vidFrames1((length1-lengthsizeextend):(length1+lengthsizeextend),(width1-wi

    % Find the maximum of all the location
    local_filtered1 = (sum(local1,3)==max(max(sum(local1,3))));

    % Use the maximum of all the location
    [length,width] = find(local_filtered1);
    position1(1,l) = length(1) + (length1-lengthsizeextend)-1;
    position1(2,l) = width(1) + (width1-widthsizeextend)-1;
end


% camera 2
for l=2:size(vidFrames2,4)
    % Previous length position of the whitest part of flashlight
    length2 = position2(1,l-1);
    % Previous width position of the whitest part of flashlight
    width2 = position2(2,l-1);

    % Adjust the rectangle, so it won't extend the video
    if (length2 - lengthsize2) < 1
        lengthsizeextend = length2-1 + (length2==1);
    elseif (length2 + lengthsize2) > size(vidFrames2,1)
        lengthsizeextend = size(vidFrames2,1) - length2 + (length2 == size(vidFrames2,1)
    else
        lengthsizeextend = lengthsize2;
    end
    if (width2 - widthsize2) < 1
        widthsizeextend = width2-1 + (width2==1);
    elseif (width2 + widthsize2) > size(vidFrames2,1)
        widthsizeextend = size(vidFrames2,1) - width2 + (width2 == size(vidFrames2,1));
    else
        widthsizeextend = widthsize2;
    end

    % Find the location of previous bucket in the rectangle
    local2 = vidFrames2((length2-lengthsizeextend):(length2+lengthsizeextend),(width2-wi

    % Find the maximum of all the location
    local_filtered2 = (sum(local2,3)==max(max(sum(local2,3))));

     % Use the maximum of all the location
    [length,width] = find(local_filtered2);
    position2(1,l) = length(1) + (length2-lengthsizeextend)-1;
    position2(2,l) = width(1) + (width2-widthsizeextend)-1;
```

```
end


% camera 3
for l=2:size(vidFrames3,4)
    % Previous length position of the whitest part of flashlight
    lenght3 = position3(1,l-1);
    % Previous width position of the whitest part of flashlight
    width3 = position3(2,l-1);

    % Adjust the rectangle, so it won't extend the video
    if (lenght3 - lengthsize3) < 1
        lengthsizeextend = lenght3-1 + (lenght3==1);
    elseif (lenght3 + lengthsize3) > size(vidFrames3,1)
        lengthsizeextend = size(vidFrames3,1) - lenght3 + (lenght3 == size(vidFrames3,1)
    else
        lengthsizeextend = lengthsize3;
    end
    if (width3 - widthsize3) < 1
        widthsizeextend = width3-1 + (width3==1);
    elseif (width3 + widthsize3) > size(vidFrames3,1)
        widthsizeextend = size(vidFrames3,1) - width3 + (width3 == size(vidFrames3,1));
    else
        widthsizeextend = widthsize3;
    end

    % Find the location of previous bucket in the rectangle
    local3 = vidFrames3((lenght3-lengthsizeextend):(lenght3+lengthsizeextend),(width3-wi

    % Find the maximum of all the location
    local_filtered3 = (sum(local3,3)==max(max(sum(local3,3))));

     % Use the maximum of all the location
    [length,width] = find(local_filtered3);
    position3(1,l) = length(1) + (lenght3-lengthsizeextend)-1;
    position3(2,l) = width(1) + (width3-widthsizeextend)-1;
end

% Save data in .mat
save(['test' num2str(N) '.mat'], 'position1', 'position2', 'position3');
```

Code II: draw data

```
clc;
clear;
close all;
```

```
% N is the test number
% Test 1: Ideal signal
% Test 2: Noisy signal
% Test 3: Horizontal displacement
% Test 4: Horizontal displacement and rotation
N = 1;

% Load the data by the first script
load(['test' num2str(N) '.mat'])

% Cut some of the data at the beginning to make the length of data in one
% test is the same
switch N
    case 1
        position1 = position1(:,9:end);
        position2 = position2(:,18:end);
        position3 = position3(:,8:end);
    case 2
        position1 = position1(:,12:end);
        position2 = position2(:,:);
        position3 = position3(:,12:end);
    case 3
        position1 = position1(:,12:end);
        position2 = position2(:,1:end);
        position3 = position3(:,4:end);
    otherwise
        position1 = position1(:,2:end);
        position2 = position2(:,10:end);
        position3 = position3(:,1:end);
end

% Find the same minimum step time for three angles
minsteps = min([size(position1,2), size(position2,2), size(position3,2)]);

% Save the six positions into a matrix
X = zeros(6,minsteps);
X(1,:) = position1(1,1:minsteps);
X(2,:) = position1(2,1:minsteps);
X(3,:) = position2(1,1:minsteps);
X(4,:) = position2(2,1:minsteps);
X(5,:) = position3(1,1:minsteps);
X(6,:) = position3(2,1:minsteps);

% Plot the height of bucket measured by camera
figure()
plot(X(1,:)), hold on
```

```
plot(X(3,:))
plot(X(6,:))
xlabel('Time');
ylabel('Height');
title('Height of bucket measured by camera');
legend('Camera 1', 'Camera 2', 'Camera 3');
print(gcf,'-dpng','Figure10.png');

% Adjust the time to [0,10] to get the same length for SVD calculation
t = linspace(0,10,minsteps);

% Find mean of each row of the data and subtract it
X = X - repmat(mean(X, 2), 1, minsteps);

% Calculate SVD
[U, S, V] = svd(X / sqrt(minsteps-1));
lambda = diag(S).^2;

% Plot Percentage of variance in different principal directions
figure()
plot((lambda / sum(lambda) * 100) ,'ro','Linewidth',2)
title('Percentage of variance in different principal directions')
xlabel('principal direction')
ylabel('Percent of variance by SVD %')
print(gcf,'-dpng','Figure11.png');

% Use PCA to reduce he redundancy
Y = U' * X;

% Plot height of bucket by PCA
figure()
plot(t,Y(1,:),t,Y(2,:),t,Y(3,:),t,Y(4,:),t,Y(5,:),t,Y(6,:))
legend('Direction 1','Direction 2','Direction 3','Direction 4','Direction 5','Direction
xlabel('Time')
ylabel('Height of bucket')
title('Height of bucket by PCA')
print(gcf,'-dpng','Figure12.png');
```