PCA & A Spring-Mass System

Paul Abboud

***Abstract*—We are tasked with applying the Principal Component Analysis method on four different datasets of a spring-mass system. The four cases include the ideal case, the noisy case, the horizontal displacement case, and the horizontal displacement and rotation case. We will explore the effect of the PCA algorithm on all four cases and compare and contrast the results of our experiments to understand the effects of noise, displacement and rotation on PCA.**

I. Introduction

Data on four situations of a spring-mass system have been recorded. The situations vary in noise, displacement and rotation. We have gathered a lot of information since each situation has been recorded in three different positions. We can determine the position in each frame by analyzing the light given off by a flashlight attached to the mass. Compiling these coordinates into a tensor which consists of all three camera positions, we can then perform Singular Value Decomposition on the tensor to find the principle components. Finally, we will compare the results of Principal Component Analysis on each of the four cases to determine the accuracy of PCA in representing the system in each case.

II. Theoretical Background

To understand the issue at hand, the following will discuss the key ideas behind Singular Value Decomposition and how it relates to Principal Component Analysis.

A. Singular Value Decomposition

A Singular Value Decomposition decomposes a matrix into three components such that

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

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{m\times n}$ are unitary matrices and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal. U is a matrix with orthonormal columns, V is an orthonormal matrix and S has non-negative singular values in decreasing order along the diagonal. Their product $A\hat{x}$ represents an orthogonal scaling by $V^T\hat{x}$ then an axis-aligned scaling by $S(V^T\hat{x})$ and finally applying resulting coefficients in the orthonormal basis by $U(S(V^T\hat{x}))$. In this application, the columns of U and V can be thought of as the eigenvectors of the correlation matrices, AA^T, between the rows and columns of A. In other words, U and V represent vectors that are consistent in all camera angles while the diagonal elements of Σ are the eigenvalues of the correlation matrices or a scalar for the vectors. Σ helps us determine which components are significant in describing the spring-mass system.

B. Principal Component Analysis

Given a vector X, we can compute the covariance of the rows with the following formula

$$C_X = \frac{1}{n-1}XX^T \tag{2}$$

This is called the covariance matrix which is a very useful tool in minimizing redundancies in the dataset which is the purpose of PCA. Applying SVD to PCA with the $1/(n-1)$ factor we have

$$C_X = AA^T = U\Sigma^2U^T \tag{3}$$

Multiplying by $U^-1 = U^T$ we have

$$Y = U^TX \tag{4}$$

The covariance of Y can be described in the same fashion as Eq.(2) and Eq.(3) where

$$C_Y = \frac{1}{n-1}YY^T = U^TAA^TU = \Sigma^2 \tag{5}$$

Thus the elements of Y aren't correlated since the only non-zero elements of Σ are along the diagonal. Now that we have uncorrelated information in Y, we have essentially reduced the dimension of the data by eliminating redundancies in the original dataset X.

III. Algorithm Implementation

To begin, we load the camera data depending on what test case we would like to examine.

```
test = 1;

load(['./cam1_'num2str(test) '.mat']);
load(['./cam2_'num2str(test) '.mat']);
load(['./cam3_'num2str(test) '.mat']);
```

Next, we will record the coordinates of the mass with respect to the first camera and repeat this process for each camera angle. We do this by first converting the image to grayscale to take advantage of the bright flashlight. Using the area denoted, we send all coordinates outside of the area around the specified coordinate to zero. We then set a new coordinate for the next frame given by the maximum coordinate in the current area and repeat this process for every frame.

```
for i=1:1:s1d
    img = rgb2gray(view1(:,:,:,i));
    img(:,1:Coord(1)-area) = 0;
    img(:,Coord(1)+area:end) = 0;
    img(1:Coord(2)-area,:) = 0;
    img(Coord(2)+area:end,:) = 0;
```

```
        [Z,index] = max(img(:));
        [P1,P2] = ind2sub(size(img),index)
            ;
        X1(i) = P2;
        Y1(i) = P1;
        Coord = [P2, P1];
end
```

After extracting the (x,y) coordinates of the mass for each camera angle, we reduce the size of each matrix so we can concatenate all three matrices into a single data set. Finally, we can perform PCA by computing the SVD on the complete dataset.

```
[M,N] = size(X);
mean = mean(X,2);
X = X - repmat(mean,1,N);
[U, S, V]=svd(X'/sqrt(N-1),'econ');
lam = diag(S).^2;

Y = V' * X;
```

IV. Computational Results

The following graphs depict the energy captured in each dimension and the positional data that corresponds to the most useful components in each case.

A. Case 1: Ideal Case

In Figure 1, we can see that the majority of energy is captured in the first and second dimensions. The positional data captured by these components is illustrated in Figure 2. We can see that the components capture oscillatory behavior which is consistent with the behavior of a spring-mass system.

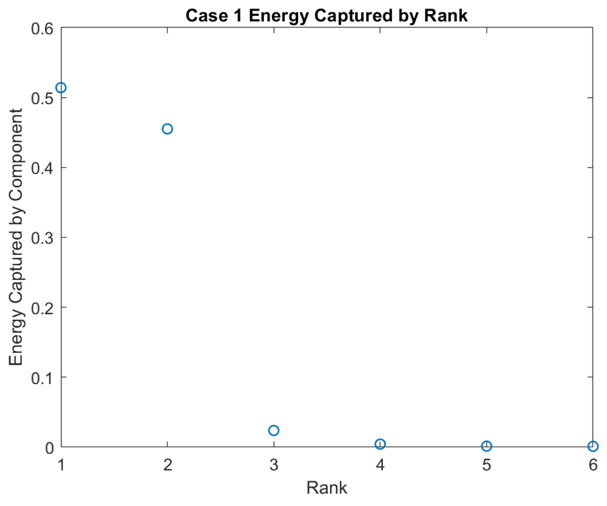

Fig. 1. Energy Captured at Each Dimension in Case 1

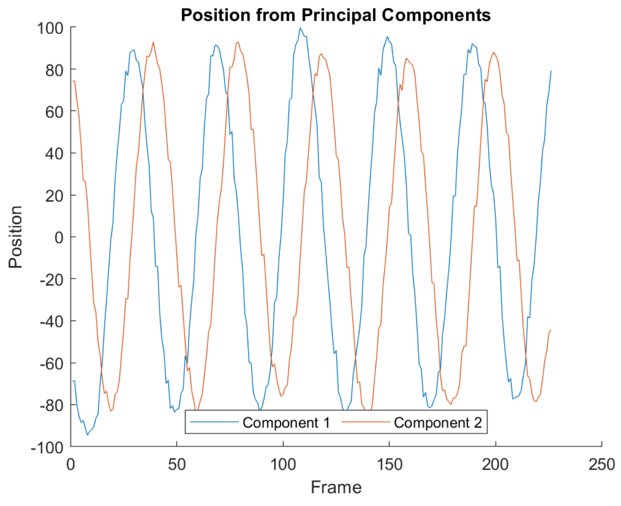

Fig. 2. Position of Mass from the 2 Leading Components in Case 1

B. Case 2: Noisy Case

Figure 3 illustrates which dimensions capture the majority of energy. In this case, over 60% of the energy is described by the first dimension. Although the first component describes the majority of the data, Figure 4 illustrates the chaotic behavior of all components. We can still see oscillatory behavior in the first component but the noise has affected the leading components when compared to the Ideal Case.

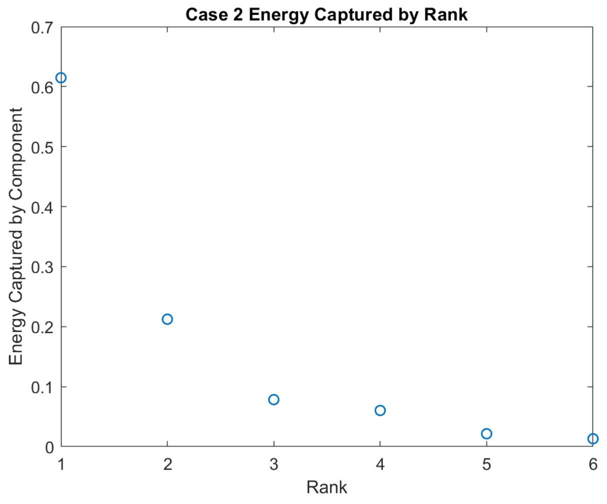

Fig. 3. Energy Captured at Each Dimension in Case 2

C. Case 3: Horizontal Displacement Case

In this case, from Figure 5, we can see that over 50% of the energy is captured in first dimension while the second and third dimensions similarly capture about 20% of the energy. In Figure 6, it seems the first component is exhibiting oscillatory behavior like in the Ideal Case while the second and third components may be describing the horizontal displacement which was not present in previous cases.

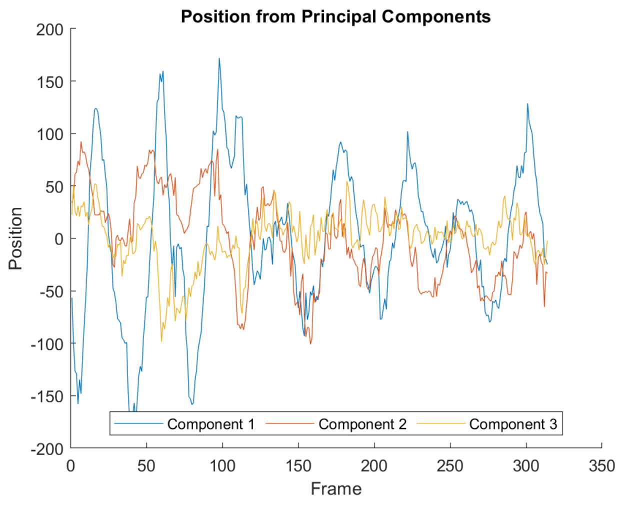

Fig. 4. Position of Mass from the 3 Leading Components in Case 2

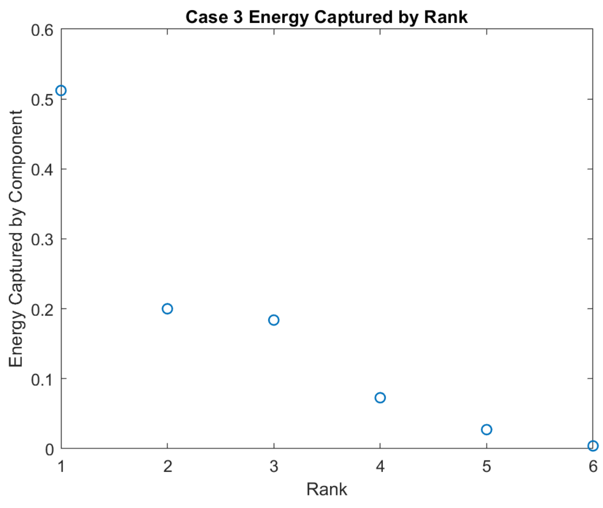

Fig. 5. Energy Captured at Each Dimension in Case 3

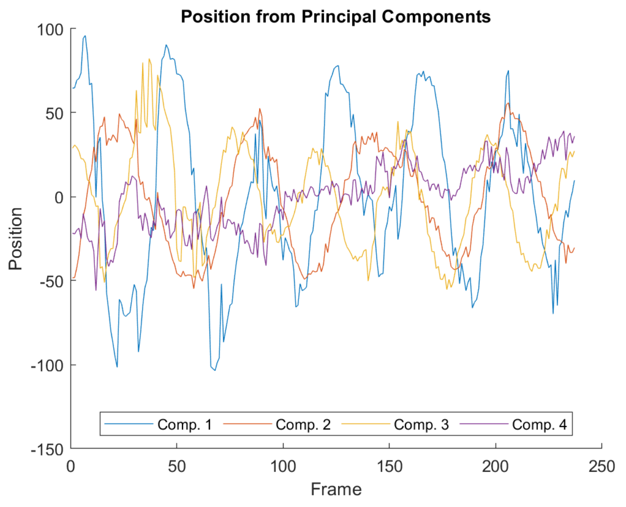

Fig. 6. Position of Mass from 4 Leading Components in Case 3

D. *Case 4: Horizontal Displacement and Rotation*

For the final case, in Figure 7, nearly 50% of the energy is captured in the first rank, 33% is captured in the second rank, 10% in the third rank and 5% in the fourth rank. Figure 8 illustrates the position data from each component. Again, we see the first component exhibiting oscillatory behavior consistent with the Ideal Case, the second component may describe the horizontal displacement similar to Case 3 while the third and fourth components describe the added rotation.

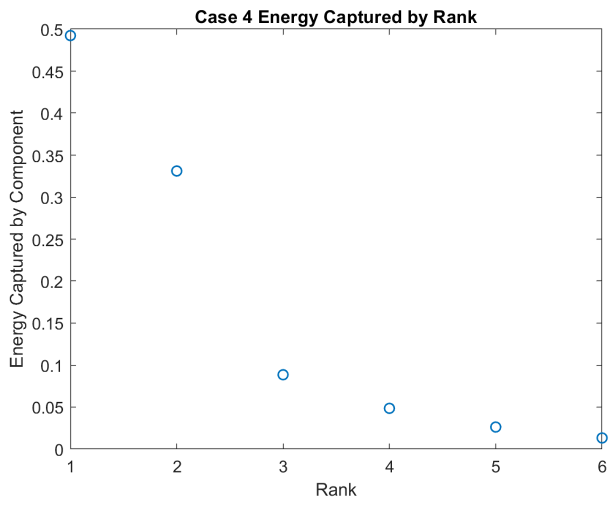

Fig. 7. Energy Captured at Each Dimension in Case 4

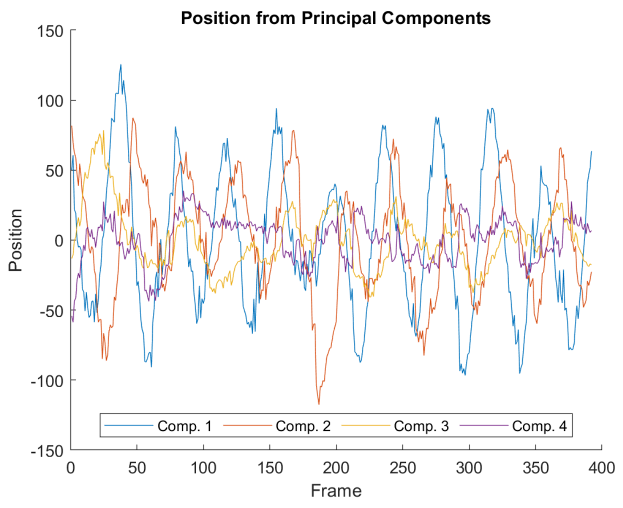

Fig. 8. Position of Mass from 4 Leading Components in Case 4

V. Summary of Results

Without using the laws of motion, we purely analyzed data to extract coordinates from a video. Applying Singular Value Decomposition to perform Principal Component Analysis, we reduced the dimension of the data so that it can be described by only a few components. The components illustrated the motion of the spring-mass system. In each case, we were successful in applying PCA and compared the results of each case to subsequent cases. Although PCA is affected by noise, we can still extract useful data from a system. As the cases became progressively more complex, PCA consistently yielded accurate and lower dimensional data to describe the motion. In this application, PCA has proven to be a powerful tool.

Appendix A

MATLAB Functions

rgb2gray: Converts an image from RGB spectrum to a grayscale image.

svd: Preforms a singular value decomposition on the specified matrix and outputs matrices U, S, V..

repmat: Repeats the matrix by the specified number of times to create a larger matrix.

diag: Extracts the diagonal elements of a matrix.

ind2sub: Converts indices to a subscript determined by the row and columns specified by the input.

Appendix B

MATLAB Code

```
clear all; close all; clc

test = 1;

load(['./cam1_' num2str(test) '.mat'])
    ;
load(['./cam2_' num2str(test) '.mat'])
    ;
load(['./cam3_' num2str(test) '.mat'])
    ;

if test == 1
    view1 = vidFrames1_1;
    view2 = vidFrames2_1;
    view3 = vidFrames3_1;
elseif test == 2
    view1 = vidFrames1_2;
    view2 = vidFrames2_2;
    view3 = vidFrames3_2;
elseif test == 3
    view1 = vidFrames1_3;
    view2 = vidFrames2_3;
    view3 = vidFrames3_3;
else
    view1 = vidFrames1_4;
    view2 = vidFrames2_4;
    view3 = vidFrames3_4;
end

[s1a,s1b,s1c,s1d] = size(view1);
[s2a,s2b,s2c,s2d] = size(view2);
[s3a,s3b,s3c,s3d] = size(view3);

area = 18;
Coord = [322,289];
X1 = zeros(1,s1d);
Y1 = zeros(1,s1d);

for i=1:1:s1d
    img = rgb2gray(view1(:,:,:,i));
    img(:,1:Coord(1)-area) = 0;
    img(:,Coord(1)+area:end) = 0;
    img(1:Coord(2)-area,:) = 0;
    img(Coord(2)+area:end,:) = 0;
    [Z,index] = max(img(:));
    [P1,P2] = ind2sub(size(img),index)
        ;
    X1(i) = P2;
    Y1(i) = P1;
    Coord = [P2, P1];
end

Coord = [239,294];
X2 = zeros(1,s2d);
Y2 = zeros(1,s2d);
for i=1:1:s2d
    img = rgb2gray(view2(:,:,:,i));
    img(:,1:Coord(1)-area) = 0;
    img(:,Coord(1)+area:end) = 0;
    img(1:Coord(2)-area,:) = 0;
    img(Coord(2)+area:end,:) = 0;
    [Z,index] = max(img(:));
    [P1,P2] = ind2sub(size(img),index)
        ;
    X2(i) = P2;
    Y2(i) = P1;
    Coord = [P2, P1];
end

Coord = [355,234];
X3 = zeros(1,s3d);
Y3 = zeros(1,s3d);

for i=1:1:s3d
    img = rgb2gray(view3(:,:,:,i));
    img(:,1:Coord(1)-area) = 0;
    img(:,Coord(1)+area:end) = 0;
    img(1:Coord(2)-area,:) = 0;
    img(Coord(2)+area:end,:) = 0;
    [Z,index] = max(img(:));
    [P1,P2] = ind2sub(size(img),index)
        ;
    X3(i) = P1;
    Y3(i) = P2;
    Coord = [P2, P1];
end

N = min(min(s1d,s2d),s3d);
X = [X1(1:N);Y1(1:N);X2(1:N);Y2(1:N);
    X3(1:N);Y3(1:N)];

[M,N] = size(X);
mean = mean(X,2);
X = X - repmat(mean,1,N);
[U, S, V] = svd(X'/sqrt(N-1));
lam = diag(S).^2;

Y = V' * X;

figure(1)
hold on
plot(Y(1,:))
plot(Y(2,:))

legend('Component 1', 'Component 2','
    Orientation','horizontal','Location
    ','south')
xlabel('Frame')
ylabel('Position')
title('Position from Principal
    Components')

figure(2)
plot(1:6, lam/sum(lam), 'o', '
    Linewidth', 1);
title("Case 1 Energy Captured by Rank
    ");
xlabel("Rank");
ylabel("Energy Captured by Component")
    ;
```