University of Washington, Seattle

Introduction to Principle Component Analysis

With Application in Feature Extraction and Pattern Detection in Simple Harmonic Spring Mass System Data

Abstract:

This paper illustrates an overview of principal component analysis and dimensional reduction to perform better data analysis. This technique allows us to extract hidden features from large amount of data by analyzing variances among variables and linearly transform original data to a new orthogonal coordinate system to discover insights. This paper demonstrates an application of principal component analysis by detecting harmonic oscillation movement in a spring-mass system data.

Hung Vinh Ngo
AMATH482
Professor Kutz
Wednesday, January 16, 2019

Section I: Introduction and Overview

With Moore’s law, the enhancement of hardware and advanced storage devices have revolutionized data processing in general. Big data, a term describing the enormous amount of data is recorded through the power of Internet, advanced smart devices such as virtual digital assistant, mobile and software applications. This large amount of data strengthens many learning algorithms such as neural network and reinforcement learning which has been shown to obtain spectacular results in fields such as computer vision, natural language processing and medicine. For these machine learning models to obtain great accuracy, many researchers have to put efforts in building efficient and clean data source to serve as input for these models. This preprocessing task plays a vital role because otherwise, models will mistakenly learn incorrect representation of the problem and produce inaccurate results.A popular technique to reduce the high dimensions of data and decrease the data storage as well as training time for models is dimensionality reduction. This technique works by extracting out important features in our data and can considerably omit variables to reduce the bulk of datasets by either performing feature projection or feature selection. This paper will discuss the implementation of principal component analysis (PCA), which describes the linear transformation of high dimensional data to a space of fewer dimensions.

Section II: Theoretical Background

There are various mathematical algorithms to perform this goal such as t-SNE, autoencoders but PCA works efficiently because it is a linear transformation. Intuitively, PCA perform the dimensionality reduction based on the idea of projecting the original correlated data to another space where its variables are uncorrelated.

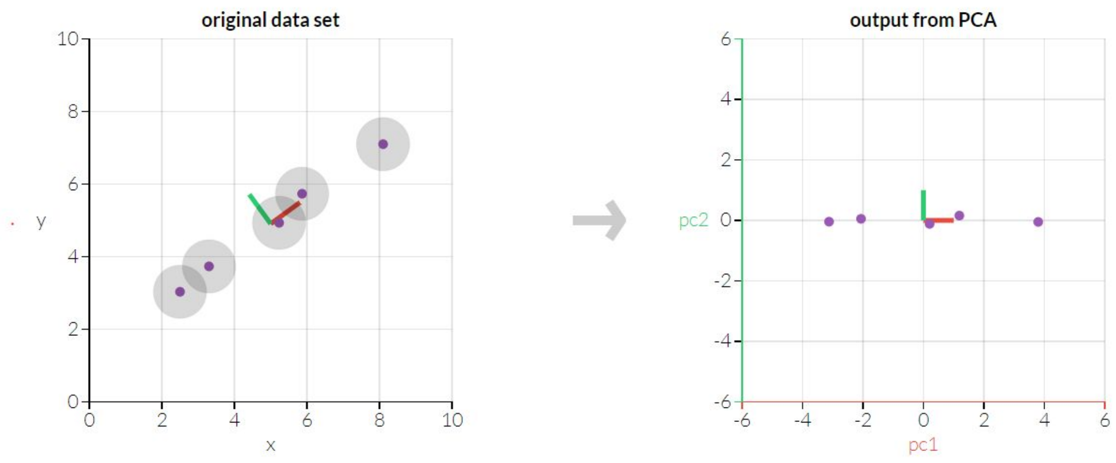


Figure 1: PCA visualized with original correlated 2 dimensional data [1]

Let’s say originally, our data have high dimensions and many of them maybe are correlated with each other. This correlation can be due to many reasons such as extra measurement while the underlying dynamics of the system is in lower dimension or some features are describing related variables like weight and size. Certainly, this is unavoidable since in most situation, we do not know the exact number of variables we should record for analysis, therefore, it is safe to collect as many as possible. PCA exploited the correlated features of our data by using covariance

[1] "Principal Component Analysis explained visually - Setosa.IO." http://setosa.io/ev/principal-component-analysis/. Accessed 25 Feb. 2019.

matrix. A covariance matrix is a square matrix where each row correspond to each variable in our data and each column describe the magnitude of correlation between that variable with other variables inside our data. The bigger in magnitude, the more correlated they are. PCA take the original data, which might be slightly or highly correlated, and project that data to a new coordinate system with the same dimensions where the axes are chosen by measuring the variance in the original data. Intuitively, PCA can be thought of as creating new features by linear combinations of the original features.
Mathematically, instead of working with the original data, we consider working with the new transformed variable that get projected to a new orthogonal coordinate system, which we will call the principal component basis.

$$Y = U'X$$

Equation 1: Matrix transformation to project original data to a new coordinate system where X is our original data and Y is the transformed data.
By taking advantage of the covariance matrix of X is a square and self-adjoint matrix, we can calculate the covariance of Y is equal to the eigenvalues of $(X X^T)$. After performing this PCA transformation, it ensures that the principle component 1 will have the most covariance, the principle component 2 will have the second most variance. Although the transformed data have the same dimension as the original dataset, we can carefully observe and omit which principle component that contributes the least to the variance of the data and thus, reduce our data dimension.

Section III: Algorithm Implementation and Development

This section will apply the idea of PCA to perform dimensionality reduction to discover patterns in high dimensional data and omit correlated features. In this experiment, we will use PCA to perform data analysis to a given spring-mass system to verify the spring movement, which we already know it will behave as a wave as in the equation F = ma. In particular, we have 3 cameras that record the image frames of the spring movement. However, we already know in advance that the mass will ideally move only in up and down direction and assumes it does not move horizontally, then only one camera is needed to capture the movement. Let's suppose we did not know the underlying equation, we want to extract out the features and the behavior of the system only through data.

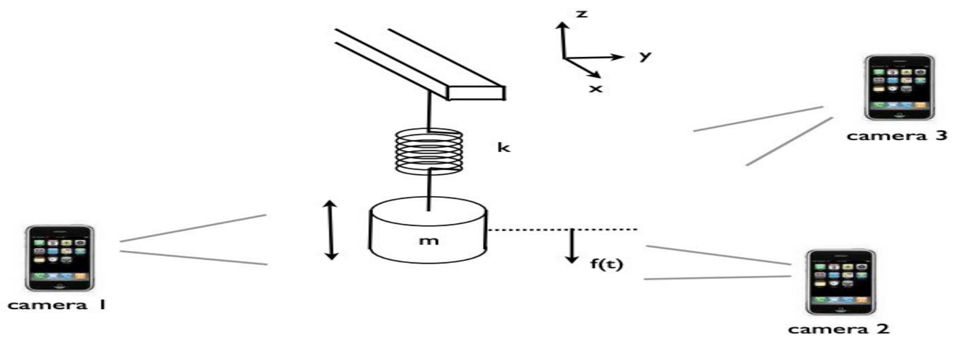


Figure 2: A visual illustration of the spring-mass system experiment to perform data analysis[2]

[2] "AMATH 582 Computation Methods for Data Analysis∗ - University of ...." 5 Feb. 2019, https://faculty.washington.edu/kutz/582.pdf. Accessed 26 Feb. 2019.

First, since we need to study the oscillation of the mass, we need to track the mass's position only and omit the background pixels in the images. There are various complex algorithms to perform object detection with high accuracy, however, in our case, we can take advantage of the mass' color is almost perfectly white and is the brightest object in the image frame. As a result, the approach this paper proposed to track the positions of the mass is by applying a rectangular filter that covers the mass' path and zeros out other pixels in the background to avoid detection error due to other noise by other bright objects in the room such as wall, shoes or watches. Afterwards, since the object becomes the brightest in the filtered frame, we can simply picking out the x and y position that corresponds to the highest pixel value in the grayscale frame. Therefore, for each camera, we can extract x and y position of the mass, so we will have 6 values in total corresponding to each camera. Certainly as mentioned, there are many better ways to detect the exact position of the object such as using complicated algorithm, or instead of finding max value in filtered frame, we can average out the bright pixels in the frame to obtain approximately the center of the mass every frame. This section's aim is to highlight the application of PCA so we can make the tracking as simple as possible, but undoubtedly, better tracking positions can help PCA to extract accurate dynamics much better.

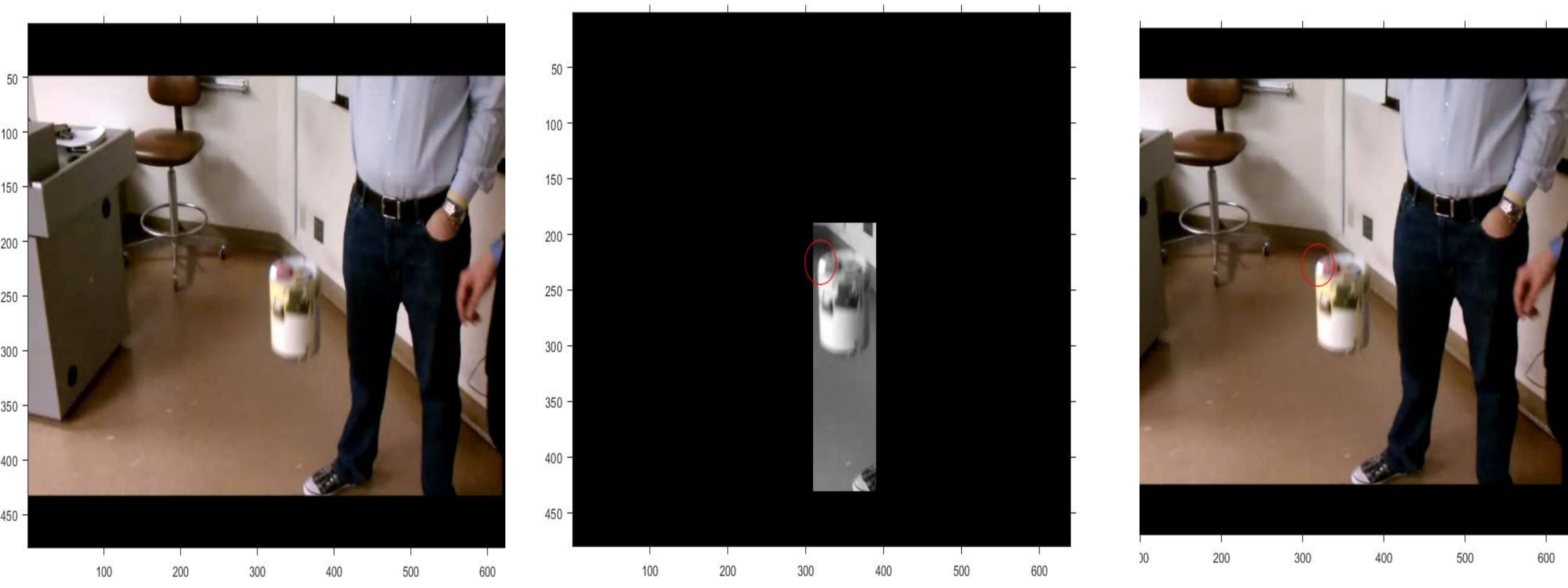


Figure 3: The tracking process visualized. (Left) original video frame with the white can is the target object that we want to extract its movement. (Middle) Applying a rectangular filter in grayscale image and omit pixels outside the rectangular box. (Right) The position of the can is projected back to the original image.

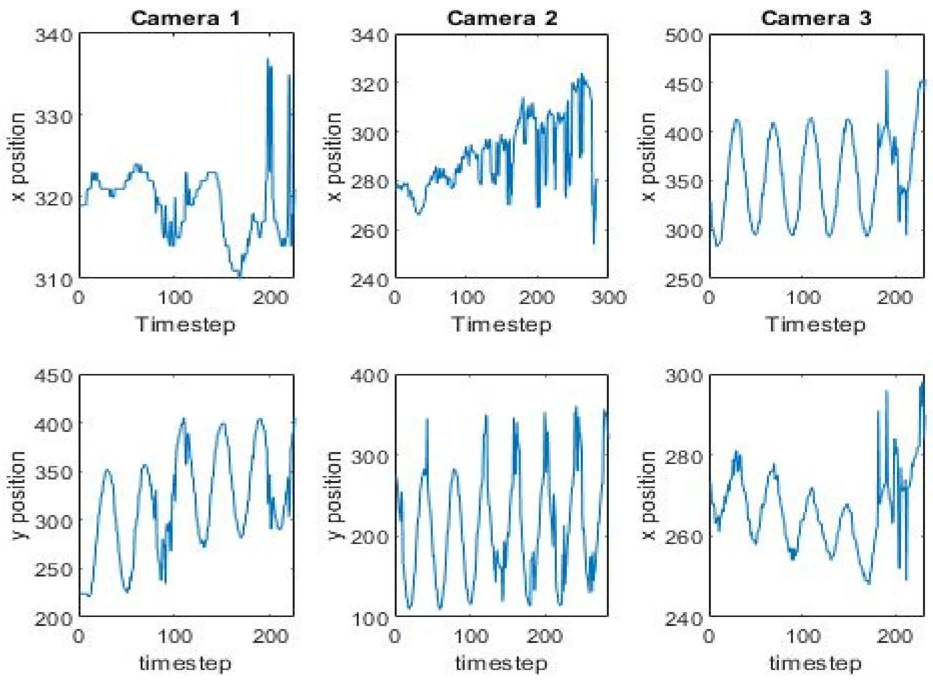


Figure 4: The tracking positions of each camera in each column, top row is x position of the mass, movement from left to right, and bottom row is y position of the mass, up-down movement.

We can clearly see the y values of the first two cameras exhibits wave moment and x position is noise due to the environment and experiment error. The third video is intentionally flipped 90 degree so the positions are reversed, which we designed to see if PCA can pick up that strange pattern in our data. To set up the problem, a state of the system is comprised of 6 values correspond to positions at each camera:

$$\begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{pmatrix}$$

With x_1, x_2 is the position of the mass in camera 1 and vice versa for camera 2 and 3. Afterwards, to build the matrix X to perform PCA, or in particular, singular value decomposition, we stack each state vector as a column for our matrix. In the end, we will have a (6 by number of frames) matrix. However, we need to recognize that the 3 videos are not in the same length, therefore the matrix can not be built with the same length if we use all values. One way we can try naively is to truncate the long videos to be the same length with the shortest video and then we can perform the vector concatenation to obtain the matrix. We will discuss other alternatives to perform this matrix creation later.

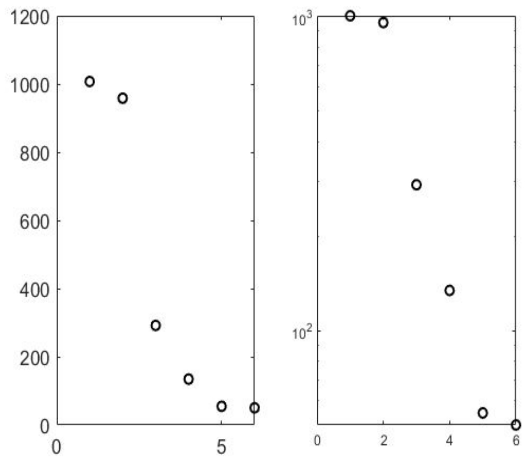


Figure 5: (left) Singular values of X and (right) log plot of the energy of each modes after performing singular value decomposition for the matrix X. From definition of PCA, there are clearly 2 principal directions that capture two variables that have biggest variance in our new dimension.

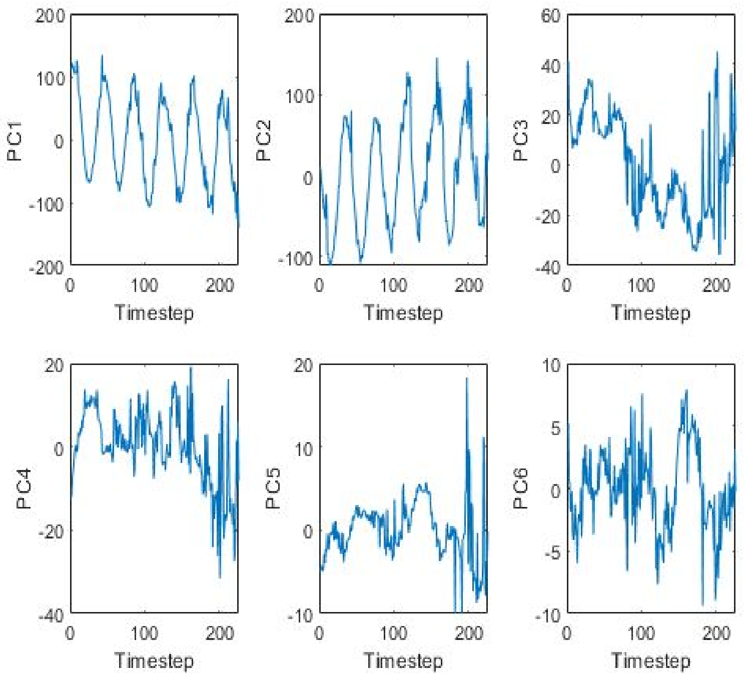


Figure 6: Output from PCA which transform our original data projected to new orthogonal coordinate system with the principal components. Based on Figure 5, the first two components capture the main dynamics of our original data while the remaining four mostly records noise in the original data for reconstruction later.

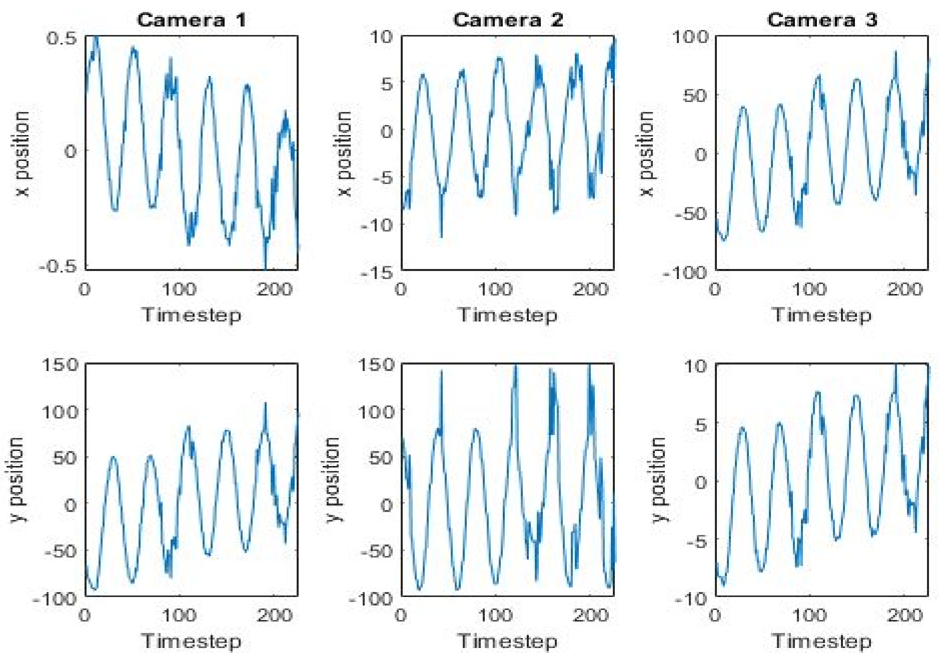
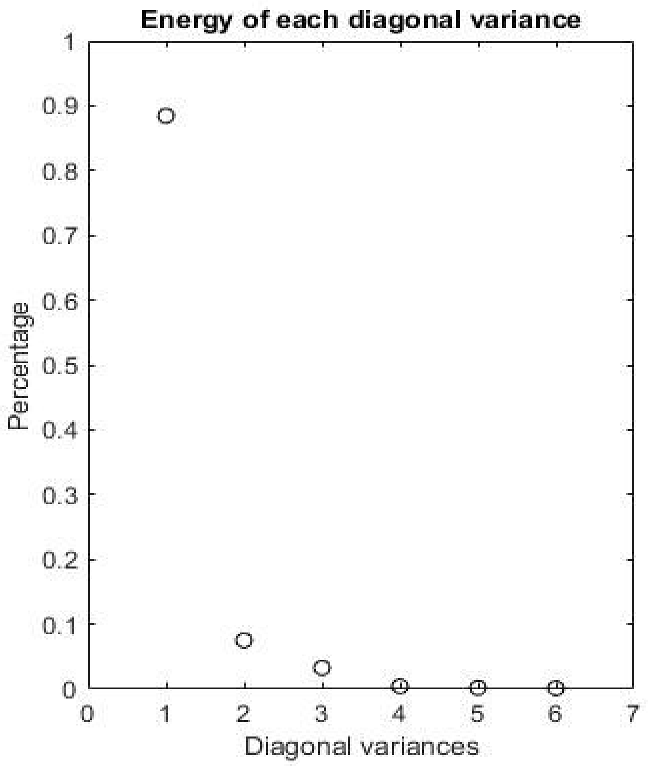
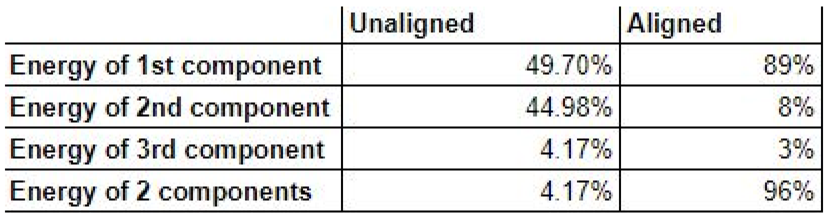


Figure 7: The reconstruction of the original data using only the first principal component. In other words, the projection of the projected principal component coordinates values back to original data.

The results is astonishing when we can discover that the original 6-dimension data can be reduced down to 2-dimension with the first two principal components. However, we expected the ideal result to have one main principal component since the main dynamics is vertical movement while the horizontal should simply be noise. The reason that leads to this computation error is that the matrix X was built with incorrect initialization. Because we attempted to declare each state of the system to be the corresponding timestep in each video, however, the videos do not start at the same position (Figure 3). As a result, it is difficult for PCA to extract features because the oscillation in videos are out of phase with each other that it potentially "interfere" each other's different wave pattern although theoretically, they should be the same wave patterns. One solution to this interference problem is that we can align the videos so that they all start at the same position and then we will start building the matrix and perform PCA.

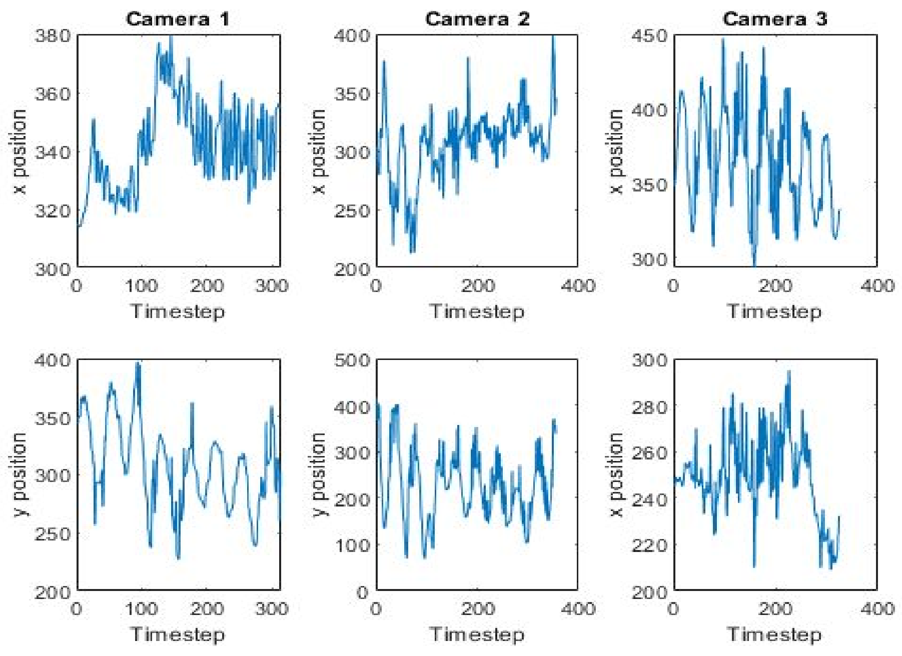




	Unaligned	Aligned
Energy of 1st component	49.70%	89%
Energy of 2nd component	44.98%	8%
Energy of 3rd component	4.17%	3%
Energy of 2 components	4.17%	96%

Figure 8: (Left) The energy in each singular values of the matrix X after aligning video frames. (Right) Comparison of the energy feature extractions of the same videos with and without aligning video frames. After aligning data, the first component has a boost in energy captured at approximately 90% of the original data, which illustrates that the main dynamics of the data can be expressed in one dimension

In the above analysis with the ideal oscillation movement, we have ignored many noise factors including shaky video capturing or even when the mass can actually oscillate in both x and y direction. that might To test the efficiency of PCA, we performed a series of similar experiments, but added more extreme noise factors. In test 2, the video recordings are introduced with camera shake.



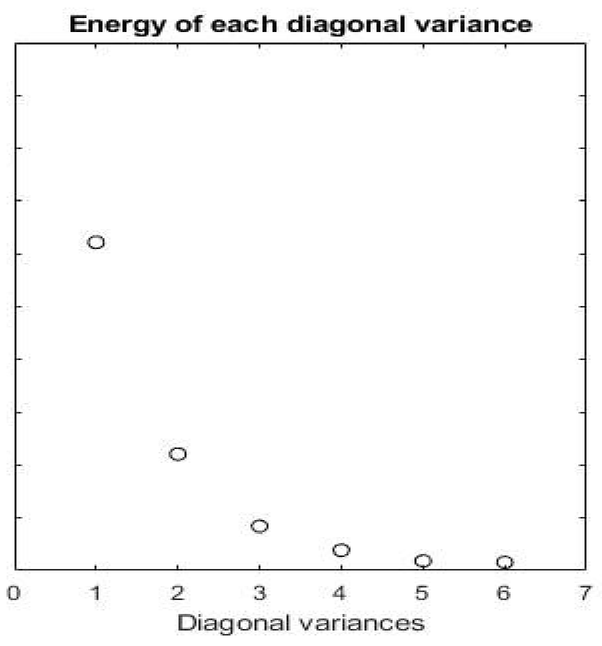
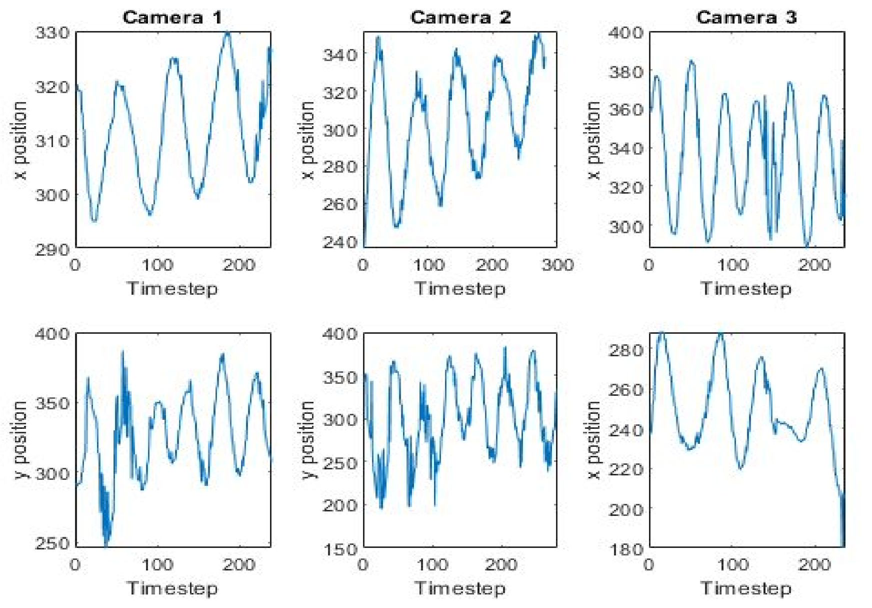


Figure 9: Experiment test 2 with videos are shaken continuously. (left) original object tracking position in each camera (right) The energy As a result, it is harder to detect the harmonic motion. The first mode is able to extract out the oscillation movement in the y direction but also get mistakenly detect patterns in the x direction.
In test 3, the mass is released off-center so as to produce motion in the x−y plane as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations.



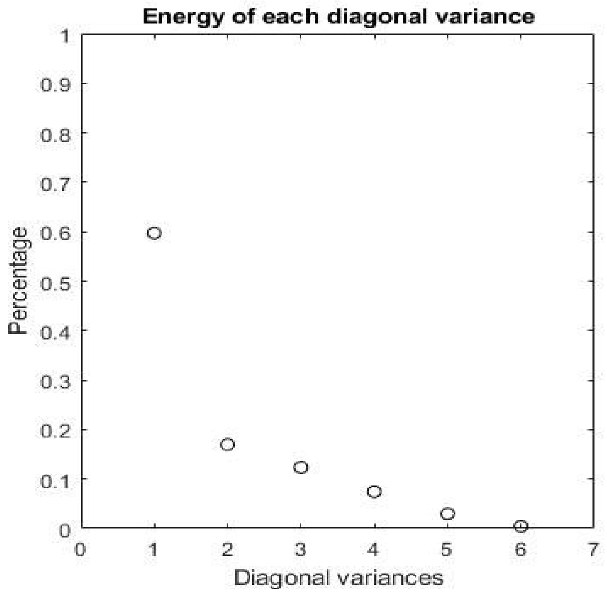


Figure 10: Experiment test 3 (left) original tracking data, (right) energy captured with each principal components

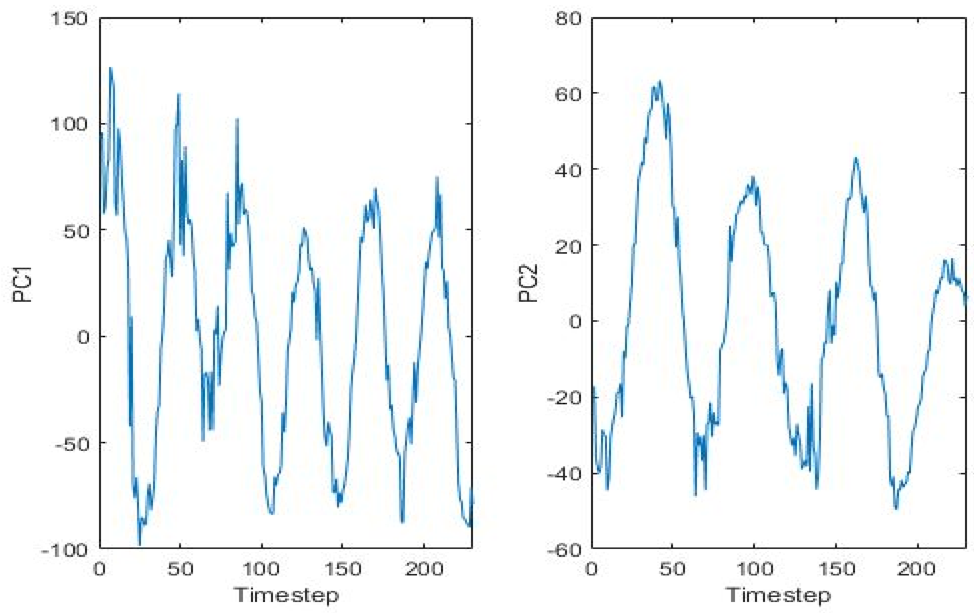


Figure 11: First and second principal components of test 3 which captures roughly 80% energy of the original data. PCA extracts the vertical oscillation of the object (with 6 wavelength) and also the horizontal oscillation (with 4 wavelength). Therefore, PCA successfully captured both the pendulum motion and harmonic oscillation in this case.

In the test case 4, the mass is released off-center and rotates so as to produce motion in the x−y plane, rotation as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations.

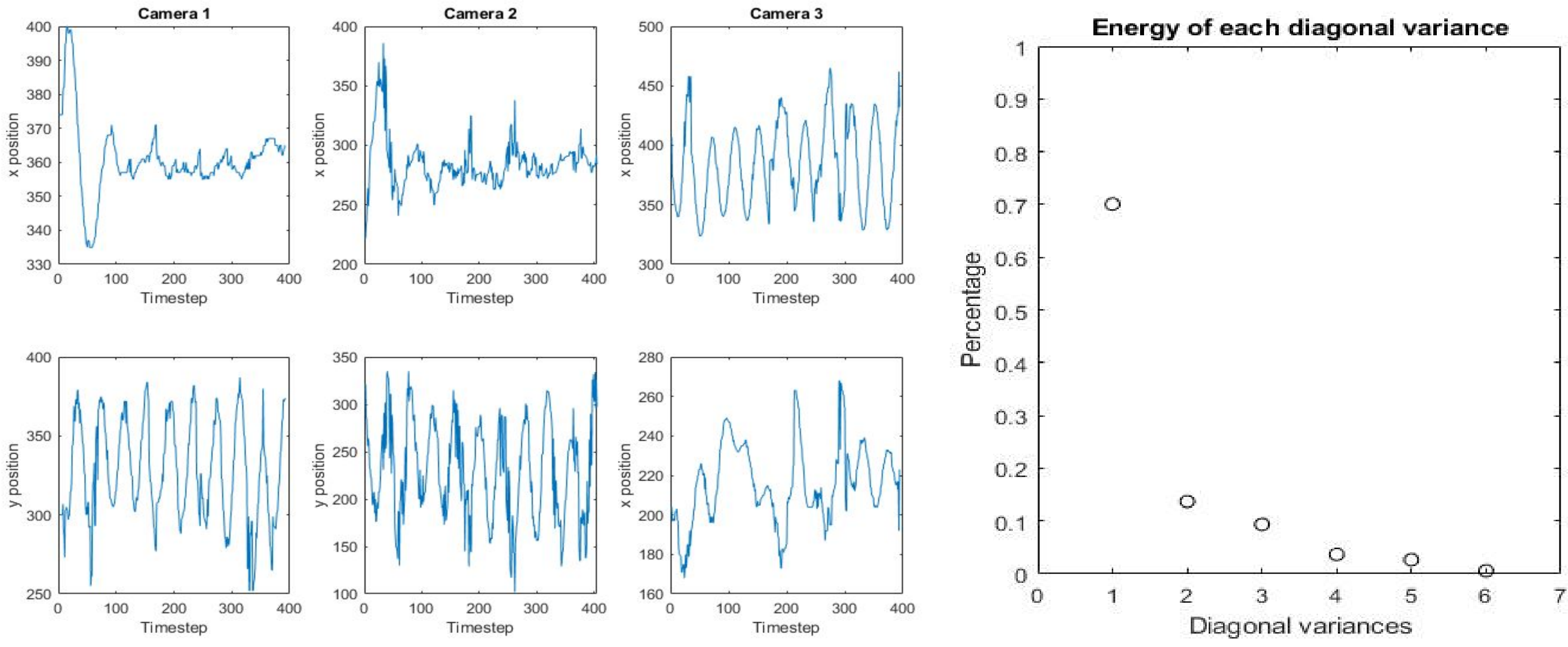


Figure 12: Experiment test 4 (left) original tracking data, (right) energy captured with each principal components

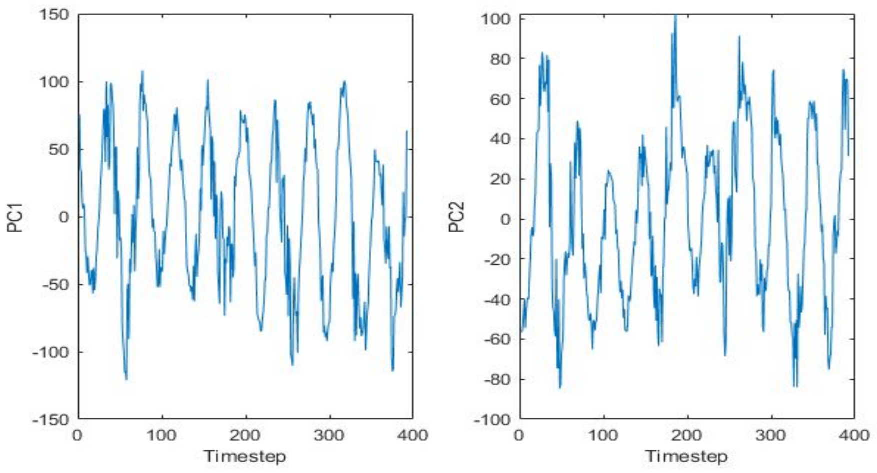


Figure 13: First and second principal components of test 3 which captures roughly 80% energy of the original data. PCA extracts the vertical oscillation of the object and also the horizontal oscillation.

It is worth noticing that in both cases 3 and 4, PCA can extract out two main principle components for x and y oscillations, but has no idea about the rotation factor in the original data. However, comparing the principal components of test 3 and test 4, it is clear to see that test 3 provided a smoother projection and better view of the wave property in the data points. Test 4 projection is more noisy which can be due to the rotation factor that distracts PCA from obtaining the pattern of the can position.To my perspective, to track the rotation measurement in the original data, we will need variables that represents depth so that we can express the change in depth to get the rotation of the object. We can also propose a better way of tracking a landmark position on the can and keep following that landmark across timesteps to see how it translates overtime to extract rotation features.

Section IV: Computational Results

	Test 1	Test 1 (aligned)	Test 2	Test 3	Test 4
Energy of 1st component	49.70%	88%	66.80%	60%	70%
Energy of 2nd component	44.98%	8%	17.30%	12%	14%
Energy of 3rdcomponent	4.17%	3%	6.52%	7%	9%

Table 1 : Energy captured by each principal components in 4 test cases

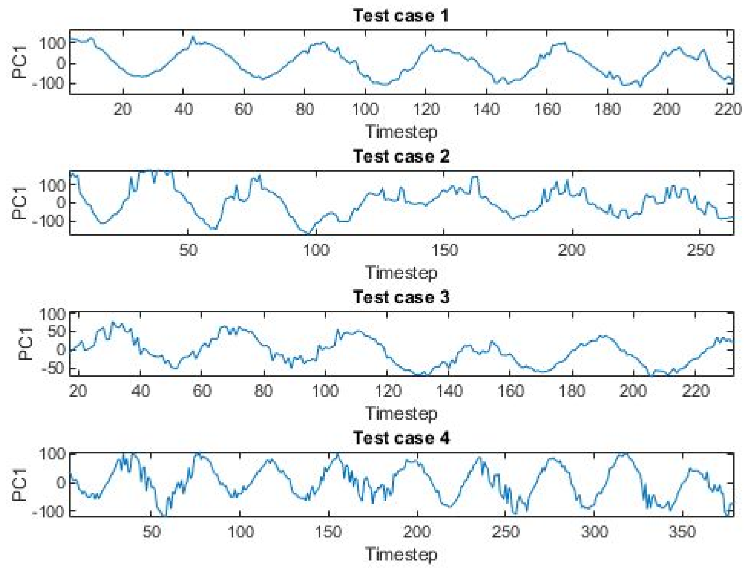


Figure 14: First principle components of 4 test cases successfully obtain wave motion As mentioned in the implementation, the accuracy of our algorithms suffered from using inaccurate and unstable object tracking because our brightest point in the box vary across video frames. We can use better movement tracking algorithms or we can perform averaging pixel values to obtain the center of the object instead of performing max operations to retrieve the object position.

Section V: Summary and Conclusions

This paper gives a gentle introduction to principle component analysis and dimensionality reduction in general. In this era of big data and machine learning, it is extremely vital to perform data mining and extract important features out of high dimensional data. The more noise we can omit, the better our analysis can enhance through either computation resources saving and even easier intuitive understanding of data with less variables. Intuitively, principal component analysis is a technique to project correlated and high dimensional data to another orthogonal coordinate system that reduces the correlation between variables. This new perspective on the original data can extract new features and help us decide which feature we can safely omit. This paper also introduces one application of principal component analysis by feature detection through analyzing spring mass system data and discovering hidden dynamics that is learned through data. This application illustrates how we can learn from data without even knowing the governing equations under systems. A breadth of technological fields can benefit from such an application of principal component analysis such as analyzing big dynamic system like turbulent fluid flow, and even as input to machine learning algorithms and neural networks.

Appendix A: MATLAB functions used and brief implementation explanation

```
close all; clear all; clc;
[U,S,V] = svd(A) performs a singular value decomposition of matrix A, such
that A = U*S*V'.
Frame2im, rgb2gray : Matlab function to work with image data by convert
matrix to image and rgb image to gray image
```

Appendix B: MATLAB Codes

```
%%
close all; clear all; clc;
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
%%
CURRENT_VID = vidFrames1_1;
shape = size(CURRENT_VID);
numFrames = shape(4);
mov = [];

gray_data = zeros(numFrames, shape(1), shape(2));
object_position = zeros(numFrames, shape(1), shape(2));
rows = [];
cols = [];

for k = 1 : numFrames
    mov(k).cdata = CURRENT_VID(:,:,:,k);
    mov(k).colormap = [];
end

for j=1:numFrames
    %X=rgb2gray(frame2im(mov(j)));
    filter = zeros(shape(1), shape(2));
    filter(190:430, 310:390) = 1;
    %height: 190-430, width: 310-390 pixel for the shannon filter for
    %camera 1
    %camera 2: 110- 450, width : 270- 365
    %camera 3: 250-350,200-450,
    X=frame2im(mov(j));
    %gray_data(j, :, :) = double(rgb2gray(X));
    filtered_data = double(rgb2gray(X)) .* filter;
    gray_data(j,:,:) = filtered_data;
    [M,I] = max(filtered_data(:));    [I_row, I_col] =
ind2sub(size(filtered_data),I);
    object_position = gray_data >= 255; %find a region of white
    imshow(uint8(filtered_data));
    hold on;
    plot(I_col, I_row, 'ro', 'MarkerSize', 30);
    rows = [rows I_row];
    cols = [cols I_col];
    drawnow;
    h = gca;
    h.Visible = 'On';
    axis on
end
figure(3)
plot(rows);
figure(4)
plot(cols);
%%
part = "_1";
load("x1" + part + ".mat")
load("y1" + part + ".mat")
x1 = cols;
y1 = rows;
load("x2" + part + ".mat")
load("y2" + part + ".mat")
x2 = cols;
y2 = rows;
load("x3" + part + ".mat")
load("y3" + part + ".mat")
x3 = cols;
y3 = rows;
subplot(2,3,1)
plot(x1);
subplot(2,3,2)
plot(x2);
```

```
subplot(2,3,3)
plot(x3);
subplot(2,3,4)
plot(y1);
subplot(2,3,5)
plot(y2);
subplot(2,3,6)
plot(y3);
%%
length = size(x1); %read above comment
length = length(2);
xs = zeros(6, length);
for i = 1:length
   xs(:, i) = [x1(i); y1(i); x2(i); y2(i); x3(i); y3(i)];
end

%%
%Perform SVD
%U(1,:) * s(1,1) * V(1,:)'
[m,n]=size(xs); % compute data size
mn=mean(xs,2); % compute mean for each row
xs=xs-repmat(mn,1,n); % subtract mean
[u,s,v]=svd(xs/sqrt(n-1)); % perform the SVD
lambda=diag(s).^2; % produce diagonal variances
Y=u'*xs; % produce the principal components projection
%%
[u,s,v] = svd(xs);
sig=diag(s);
figure(7);
subplot(1,2,1), plot(sig,'ko','Linewidth',[1.5])
% axis([0 25 0 50])
set(gca,'Fontsize',[13],'Xtick',[0 5 10 15 20 25])
text(20,40,'(a)','Fontsize',[13])
subplot(1,2,2), semilogy(sig,'ko','Linewidth',[1.5])
% axis([0 25 10^(-18) 10^(5)])
set(gca,'Fontsize',[13],'Ytick',[10^(-15) 10^(-10) 10^(-5) 10^0 10^5],...
'Xtick',[0 5 10 15 20 25]);
text(20,10^0,'(b)','Fontsize',[13])
v2=v(:,1:3); % pull out modes
c=v2'*xs'; % project the modes
% subplot(2,1,2)
% plot(t,-c(1,:),'k',t,c(2,:),'k--',t,c(3,:),'k:','Linewidth',[2])
% set(gca,'Fontsize',[13])
% legend('mode 1','mode 2','mode 3','Location','NorthWest')
% text(1.8,6,'(c)','Fontsize',[13])
```

```
%%
figure(2)
predicted = u' * xs;
subplot(2,3,1)
plot(predicted(1, :));
subplot(2,3,2)
plot(predicted(3, :));
subplot(2,3,3)
plot(predicted(5, :));
subplot(2,3,4)
plot(predicted(2, :));
subplot(2,3,5)
plot(predicted(4, :));
subplot(2,3,6)
plot(predicted(6, :));

%%
%Energy calculation
energy1=sig(1)/sum(sig)
energy2 = sig(2) / sum(sig)
energy3 = sig(3)/ sum(sig)
energy13=sum(sig(1:3))/sum(sig)

%%
%Calculate test result
for j=1:2
    predicted=u(:,1:j)*s(1:j,1:j)*v(:,1:j)'; % modal projections
end
 predicted=u(:,1)*s(1,1)*v(:,1)';
figure(9)
subplot(2,3,1)
plot(predicted(1, :));
subplot(2,3,2)
plot(predicted(3, :));
subplot(2,3,3)
plot(predicted(5, :));
subplot(2,3,4)
plot(predicted(2, :));
subplot(2,3,5)
plot(predicted(4, :));
subplot(2,3,6)
plot(predicted(6, :));
```