Understanding Gravity: A Data-Driven Approach

Kuan H. Chen

Chemical Engineering

Applied & Computational Mathematical Sciences

Abstract

Legend has it that after an apple hit Sir Isaac Newton on the head while he was sitting under a tree in the 1600s, he began quest to solve the mystery behind gravity. Somewhere along this quest he came across a series of videos from the future – videos that captured the movement of what appeared to be a spring-mass system. Excited to use these as a basis of research, Newton realized that there were a few problems with the videos – not only were they filmed on different time frames, they were also filmed on different and arbitrary coordinates systems. To make matters even worse, the videos are blurry, presumably because of the shaking hands of the cameraman. Using the Gabor Transformation, signal filtering techniques, and Principle Component Analysis, can Newton understand the content of these videos and solve the universe?

Keywords: Gabor Transformation (GT), signal filtering, Principle Component Analysis (PCA)

Understanding Gravity: A Data-Driven Approach

I. Introduction

All images, colored or not, can be seen as waves of multiple dimensions – black and white images are waves whose signals depend on the spatial coordinates x and y, while colored images include a third dimension that is characterized as a combination of red, blue, and green to different degrees. Videos, on the other hand, can be understood as images with another dimension that depends on time, or as a series of images. In this report, videos documenting 4 cases of spring-mass systems are analyzed – an ideal case, hand-shaking case, pendulum-motion case, and rotating case. Each case is recorded by 3 cameras, each one on a different frame of reference. In the ideal case, the system exhibits displacement in one direction and minimal shaking is seen. In the hand-shaking case, similar motion is seen, but the image is blurred by the moving frame of reference. In the pendulum-motion case, the mass is moving in two directions instead of just one. In the rotating case, the mass moves in one direction, but is also rotating. A bright light emitting a strong signal is placed on top of the mass – by capturing this signal and analyzing how it moves with time, the nature of gravity can be understood. However, given the differing frames of reference among the cameras, whether or not the oscillating system is 1-D in nature must first be addressed. The following section explores the GT, used to isolate a single window of data to analyze, signal-filtering techniques used to locate strong signals on the mass, and the PCA, used to determine the degrees of freedom involved in the system.

II. Theoretical Background

1. **Gabor Transformation:**

The GT allows one to analyze data in an enclosed window while ignoring all outside of it. In the context of image processing, this window will crop out a section of the image for analysis. In this report, the Shannon window is used, and is defined as follows in Equation 1,

$$\mathbf{1.}\ \psi_{Shannon}(x,y) = \begin{cases} 0\ \forall \quad x - x_0 < 0, y - y_0 < 0 \\ 1\ \forall\ x - x_0 < w_x, y - y_0 < w_y \\ 0\ \forall\ w_x < x - x_0, w_y < \ y - y_0 \end{cases}$$

where x, y represent points on an image and w_x and w_y define the size of the window.

2. **Signal Filtering:**

In a black and white image, the amplitude of signals ranges in between 0, pitch black, and 255, pure white. It is reasonable to assume that the aforementioned light on top of the mass will be emitting the strongest signal among the other objects contained in the Shannon Window, thus having an amplitude near 255 in all time frames. The images can be further filtered so as to isolate the strongest signals in the Shannon window by setting a certain threshold value below which signals will be converted to 0.

3. **Principle Component Analysis:**

The PCA is used to determine how much of measured data is redundant. Being able to simplify and only retain key components of large amounts of data allows one to analyze it more efficiently. Suppose matrix $\boldsymbol{X}$ contains all measurements of the spring-mass system, and that $\boldsymbol{U}$ contains all unitary vectors that form a basis of $\boldsymbol{X}$ (all vectors u_i are **linearly independent**). The matrix $\boldsymbol{Y}$, defined as $\boldsymbol{U^*X}$**,** is a low-rank projection of $\boldsymbol{X}$ onto the basis $\boldsymbol{U}$ containing a set of components that describe $\boldsymbol{X}$. Since any matrix of data $\boldsymbol{A}$ can be decomposed into a collection of component matrices $\boldsymbol{U\Sigma V^T}$, $\boldsymbol{U}$ can be found for $\boldsymbol{X}$. This decomposition is called the Singular-Value Decomposition (SVD) and is shown in Equation 2,

$$\mathbf{2.}\, \boldsymbol{A} = \boldsymbol{U\Sigma V^T} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} u_i \sigma_i v_i^T$$

where σ_i represents the standard deviation in the i^{th} dimension. High σ_i suggests high variance, or ${\sigma_i}^2$, in the i^{th} dimension, which in turn represents a strong presence of component u_i in $\boldsymbol{A}$**.** The u_i that corresponds to the highest ${\sigma_i}^2$ is denoted as u_1 and is termed the principle component. The set of u_i that corresponds to high σ_i can be used to find $\boldsymbol{Y}$. Equation 3 can be used as a measure to determine whether an appropriate $\boldsymbol{Y}$ is found.

$$\mathbf{3.}\, f = \sum_{i}^{m} {\sigma_i}^2 \Big/ \sum_{i}^{n} {\sigma_i}^2 \,, m < n$$

When the variation in f is little as i increases, the corresponding matrix is $\boldsymbol{Y}$.

III. Algorithm Implementation and Development

This section details the process of analysis for the ideal spring-mass case. The methods described here may also be applied to the shaking, pendulum, and rotating cases.

Three series of colored images that form separate videos are provided and each represent data taken through the lens of three different cameras. This data is stored in the variable `vidFrames` in MATLAB. Analyzing colored videos is an onerous task as it presents 4 dimensions of data. In order to expedite the analysis, all images are first converted into black and white images to reduce the data set to 3 dimensions. This can be accomplished with the following MATLAB command:

```
for i = 1:numFrames
    vidFramesGray = double(rgb2gray((vidFrames(:,:,:,i))));
```

Where `vidFramesGray` represents the black and white image.

The range in which the mass oscillates through time can be identified, and an appropriate Shannon window can be devised to isolate it. The following MATLAB commands are used to generate a Shannon window:

```
darkFilter = zeros(xFrame, yFrame);
darkFilter((xFrame-shannonWidth)/2+1+leftRightShift:xFrame/2+shannonWidth/2+leftRightShift,...
    (yFrame-shannonHeight)/2+1+topDownShift:(yFrame+shannonHeight)/2+topDownShift)...
    = ones(shannonWidth, shannonHeight);
```

where `darkFilter` represents the Shannon window; `xFrame`, `yFrame` represent the horizontal and vertical length of the images, respectively; `shannonWidth`, `shannonHeight` represent the width and height of the window, respectively; `leftRightShift`, `topDownShift` represent how far from the middle of the image the center of the window must be shifted horizontally and vertically, respectively. Figure 1 shows the effect of the Shannon window on an image taken at an arbitrary time.

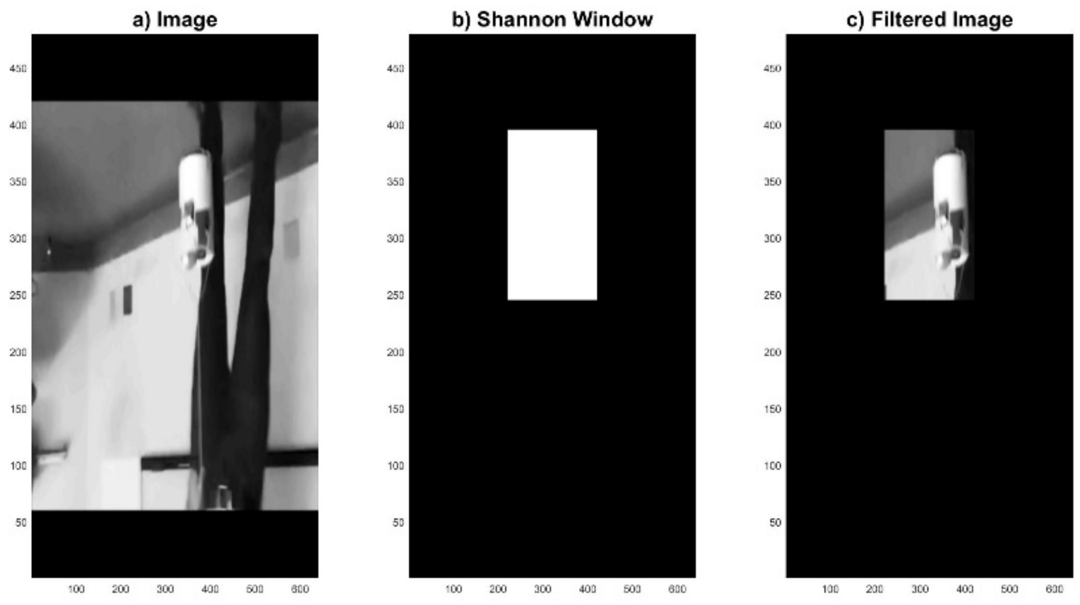


Figure 1. (a) the original image (b) the Shannon window (c) the image after applying the Shannon window

The cropped image in Figure 1. (c) is further filtered so that only the dominant white spots are retained. This is achieved with the following MATLAB commands:

```
toBeIgnored = find(vidFramesGray < threshold);
vidFramesGray(toBeIgnored) = 0;
filteredFrame = vidFramesGray.*(darkFilter');
```

where `threshold` represents the amplitude below which signals are filtered out. Figure 2 shows the filtered image in Figure 1. (c) that uses a `threshold` value of 250.

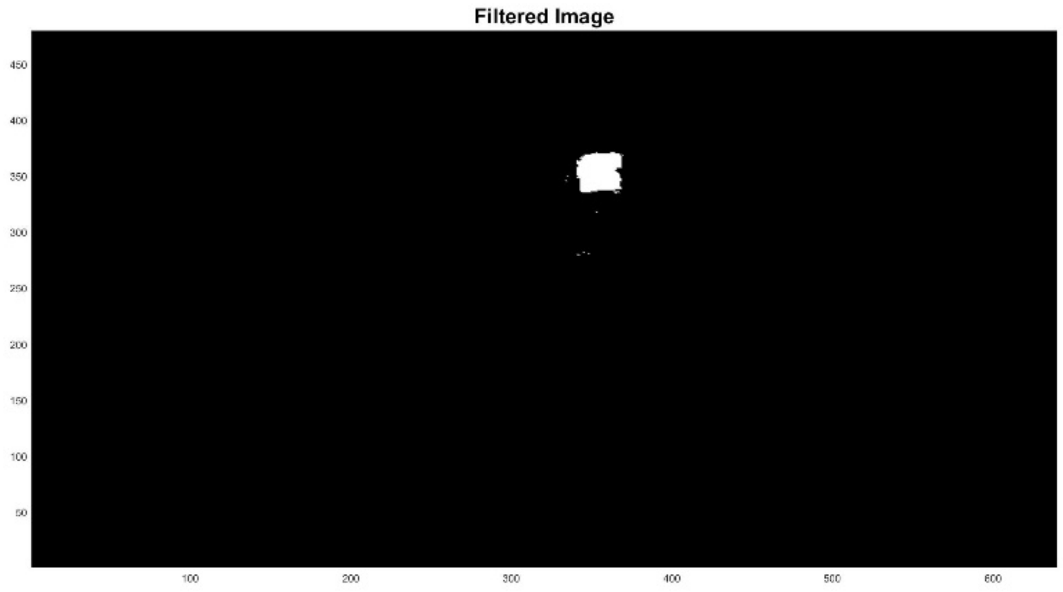


Figure 2. Signals above an amplitude of 250

The maximum value, as well as its linear index in relation to the rest of the data points in the image are then documented in order to find the brightest point's motion through time. This is achieved using the following MATLAB commands:

```
for i = 1:numFrames
    [M,I] = max(filteredFrame(:));
    [mapY, mapX] = ind2sub([yFrame xFrame],I);
    xTrajectory = [xTrajectory mapX];
    yTrajectory = [yTrajectory mapY];
```

where `M` represents the amplitude of the brightest spot in the filtered image; `I` represents the linear index of the maximum amplitude; `mapY` and `mapX` represent the vertical and horizontal coordinates of the brightest spot in the image; `xTrajectory` and `yTrajectory` records the path the brightest spot takes as time progresses.

It should be noted that the brightest spot in each image as time progresses may not correspond to the same point on the actual mass. However, due to the application of the Shannon window and signal filter, the points on the mass that the brightest points in each image correspond to will not be far from each other. The following section presents the paths that the mass takes in each case and explores PCA.

IV. Computational Results

1. Ideal Case:

   Camera 1, 2, and 3 in the ideal case all observe sinusoidal behavior in the vertical direction with high variance, as seen in Figure 3. The presence of the large, seemingly random spikes that are scattered throughout the plots can be explained by the `max` function in the routine presented in the previous section capturing bright spots that correspond to different points on the mass at each point of time. This is due to its nature of returning the first maximum value it finds if multiple points share the same maximum amplitude. Luckily, because of the application of two filters, these spikes do not prevent one from identifying the general path that the mass takes. In the horizontal direction, little variance is observed, as expected.

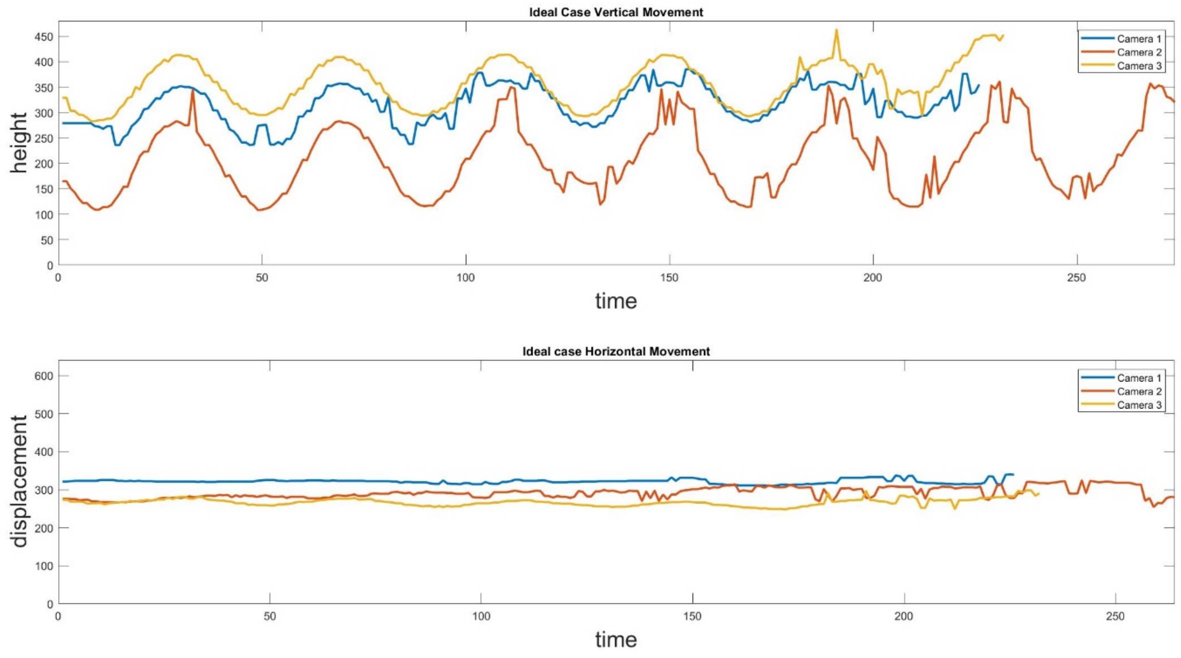
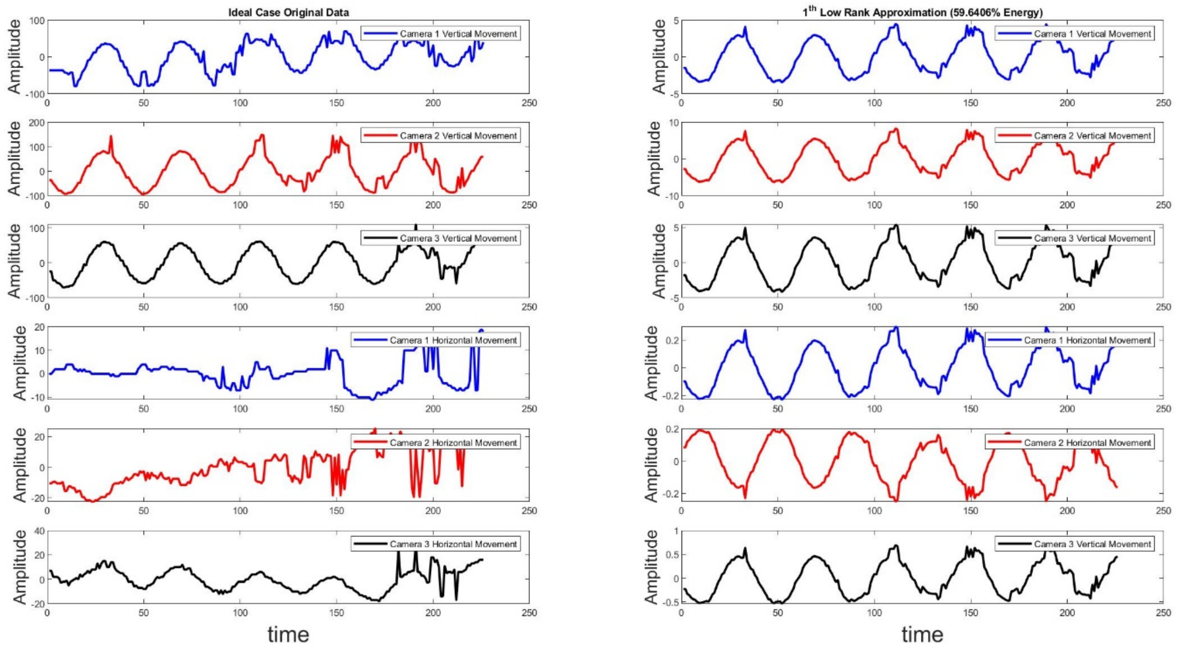


Figure 3. Vertical and horizontal motion of mass in ideal case

Figure 4 presents the 1st and 2nd low-rank approximations of the ideal case original data. As can be seen in Figure 4 (a), the first approximation captures 60% of the original data, implying that the 1st dimension of the unitary matrix $\boldsymbol{U}$ of $\boldsymbol{X}$ contains a significant portion of the original data set. In Figure 4. (b) the 2nd low-rank approximation is seen. Since the percentage of captured data only increased by ~17%, small compared to the 60% captured in the 1st approximation, one can conclude that the first vector in $\boldsymbol{U}$ is sufficient in representing $\boldsymbol{X}$. This implies that in fact, only one camera was needed to capture the motion of the spring-mass system to gain similar insight into the system as 6 cameras do.

(a)



(b)

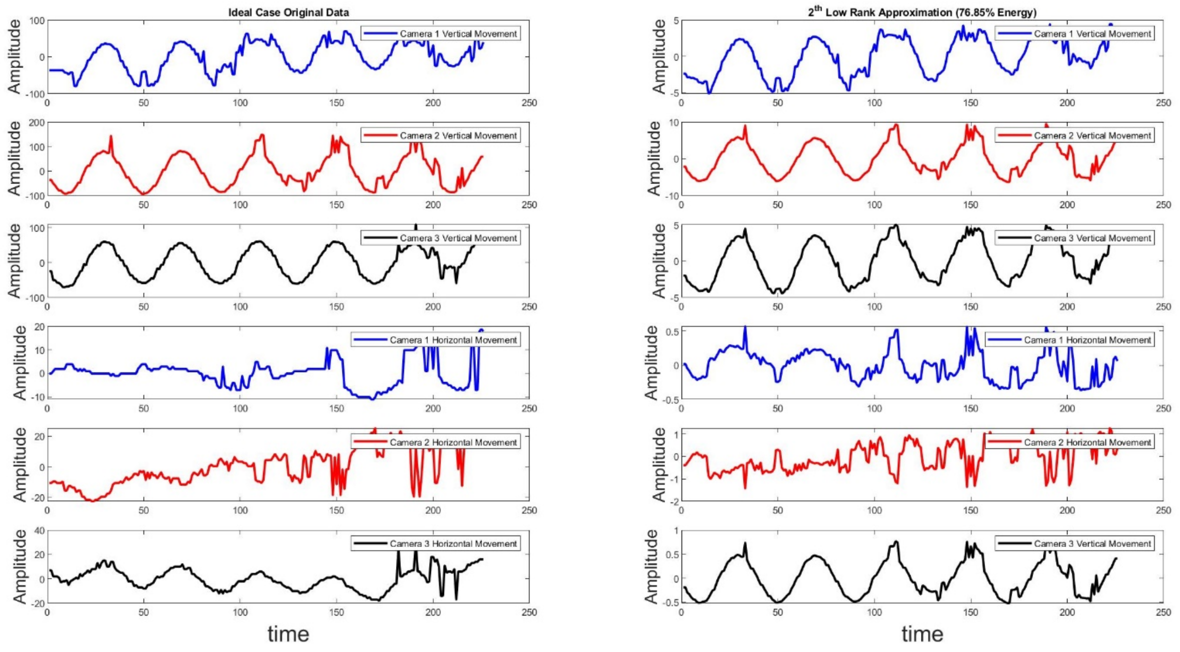
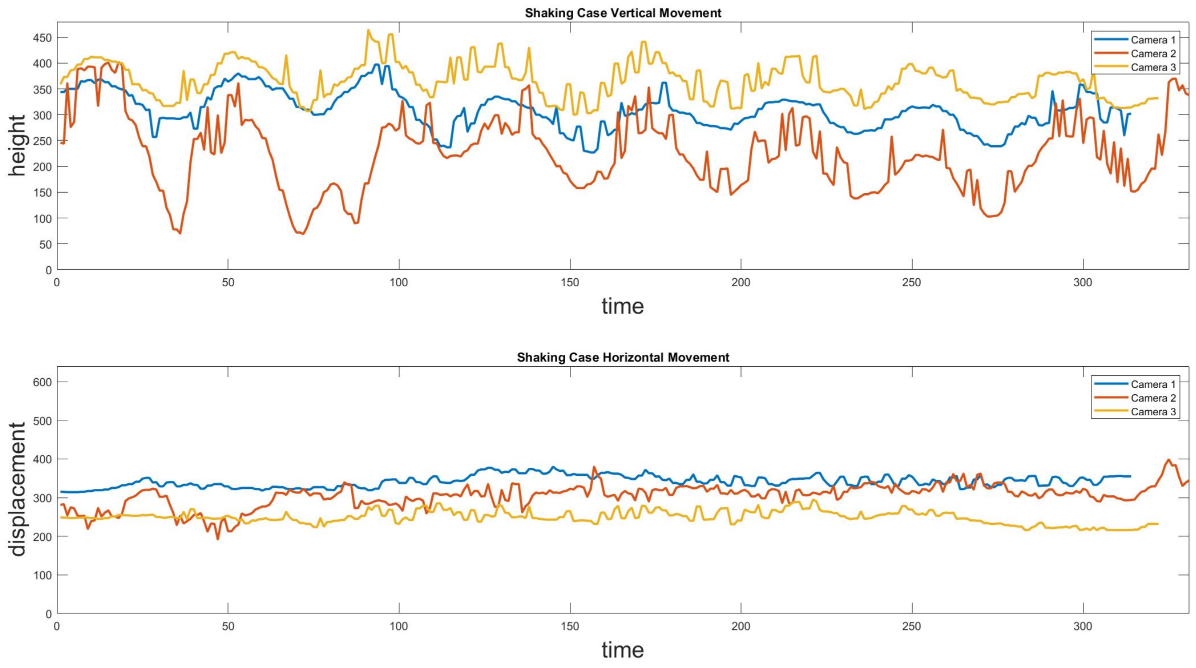


Figure 4. Comparison between original data and 1st and 2nd low rank approximation

2. Shaking Case:

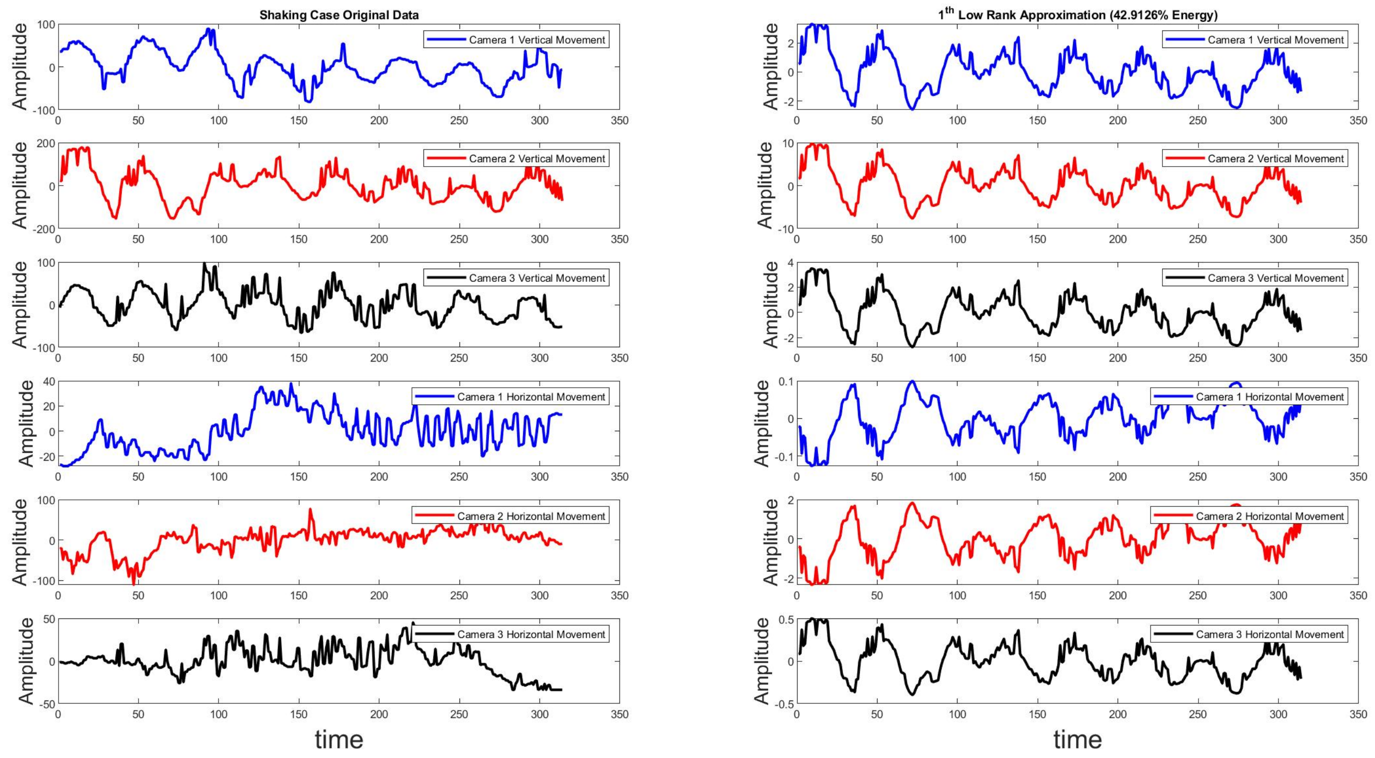
   Camera 1, 2, and 3 in the shaking case all observe sinusoidal behavior in the vertical direction with high variance, as seen in Figure 5. The presence of the large, seemingly random spikes that are scattered throughout the plots can be explained by the significant change in position of the bright spots in the image when it shakes. Luckily, because of the application of two filters, these spikes do not prevent one from identifying the general path that the mass takes. In the horizontal direction, little variance is observed. This implies that the motion in this case is much similar to that in the ideal case.



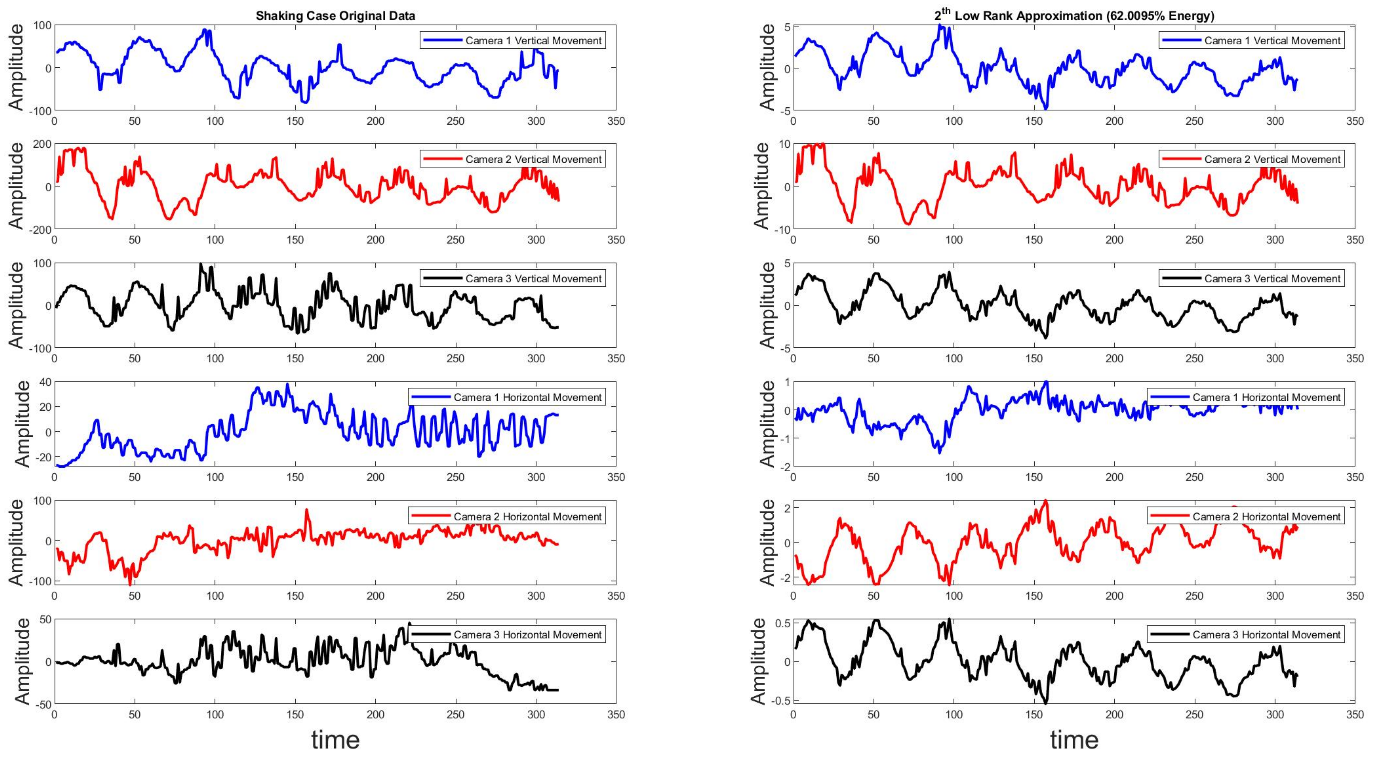
Figrue 5. Vertical and horizontal movement of mass in shaking case

Figure 6 presents the 1st ,2nd and 3rd low-rank approximations of the ideal case original data. As can be seen in Figure 6. (a), the first approximation captures only 42% of the original data, implying that although the 1st dimension of the unitary matrix $\boldsymbol{U}$ of $\boldsymbol{X}$ contains a significant portion of the original data set, others are expected to as well. In Figure 6. (b), the 2nd low-rank approximation is seen. Since the percentage of captured data increased by ~20%, sizable compared to the 42% captured in the 1st approximation, one can conclude that the first two vectors in $\boldsymbol{U}$ contributes significantly to $\boldsymbol{X}$. From the second to third low-rank approximation, ~14% is captured. Here it is not clear whether or not the third component should be included to represent $\boldsymbol{X}$ and other analysis methods may be needed to draw better conclusions. This implies that the PCA may not be the best analysis tool for this system.

(a)


Shaking Case Original Data
1th Low Rank Approximation (42.9126% Energy)
Camera 1 Vertical Movement
Camera 2 Vertical Movement
Camera 3 Vertical Movement
Camera 1 Horizontal Movement
Camera 2 Horizontal Movement
Camera 3 Horizontal Movement
Amplitude
time

(b)


Shaking Case Original Data
2th Low Rank Approximation (62.0095% Energy)
Camera 1 Vertical Movement
Camera 2 Vertical Movement
Camera 3 Vertical Movement
Camera 1 Horizontal Movement
Camera 2 Horizontal Movement
Camera 3 Horizontal Movement
Amplitude
time

(c)

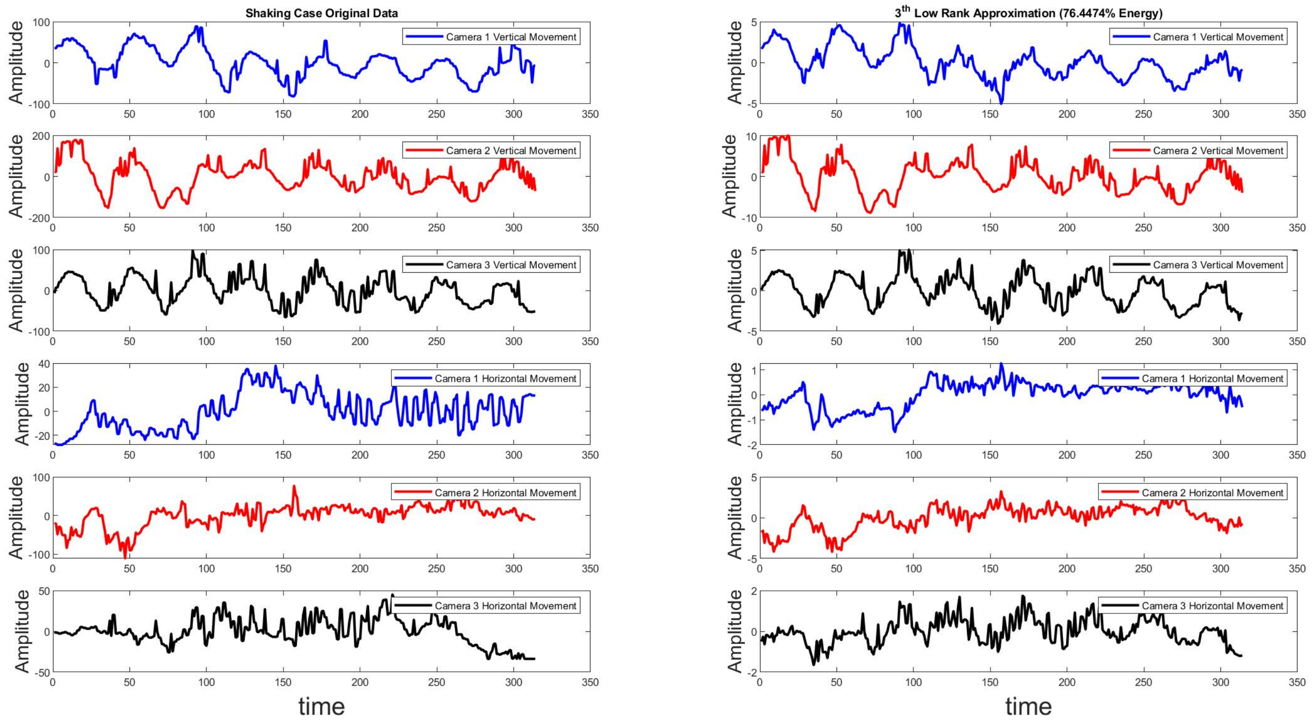


Figure 6. Comparison between original data and 1st, 2nd and 3rd low rank approximation

3. Pendulum Case:

   Camera 1, 2, and 3 in the pendulum case all observe sinusoidal behavior both in the vertical and horizontal direction with high variance, as seen in Figure 7. This implies that the motion in this case will be described by at least two of the low-rank components. The slight dips that are scattered throughout the plots can be explained by the mass occasionally swinging outside of the Shannon Window.

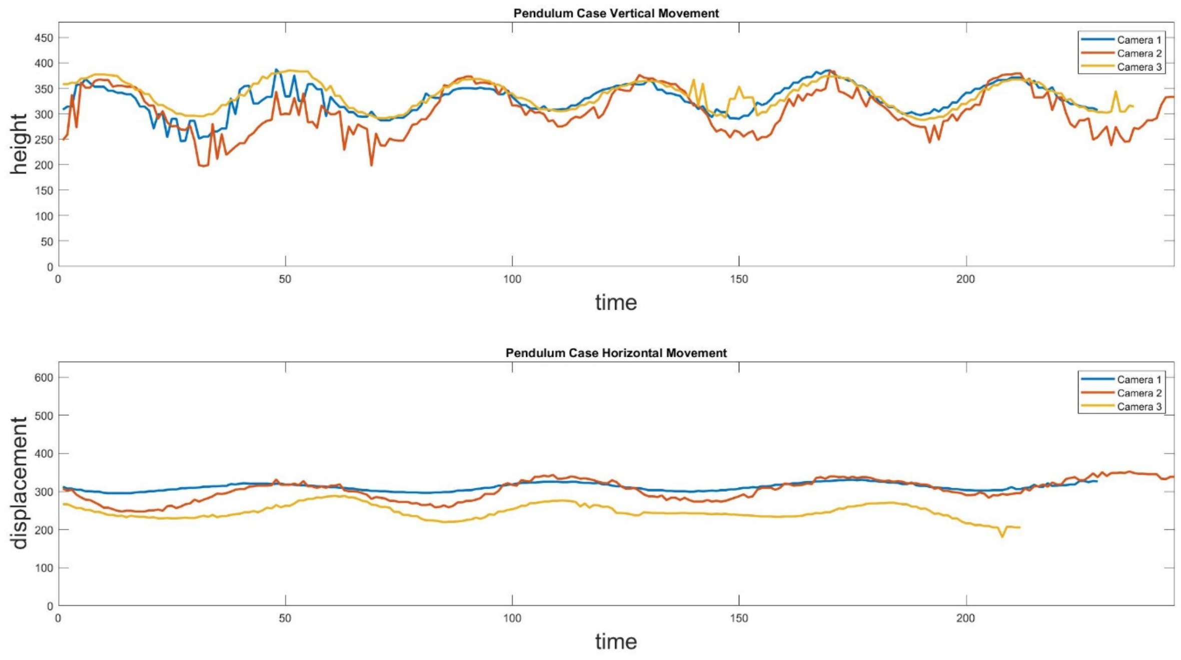
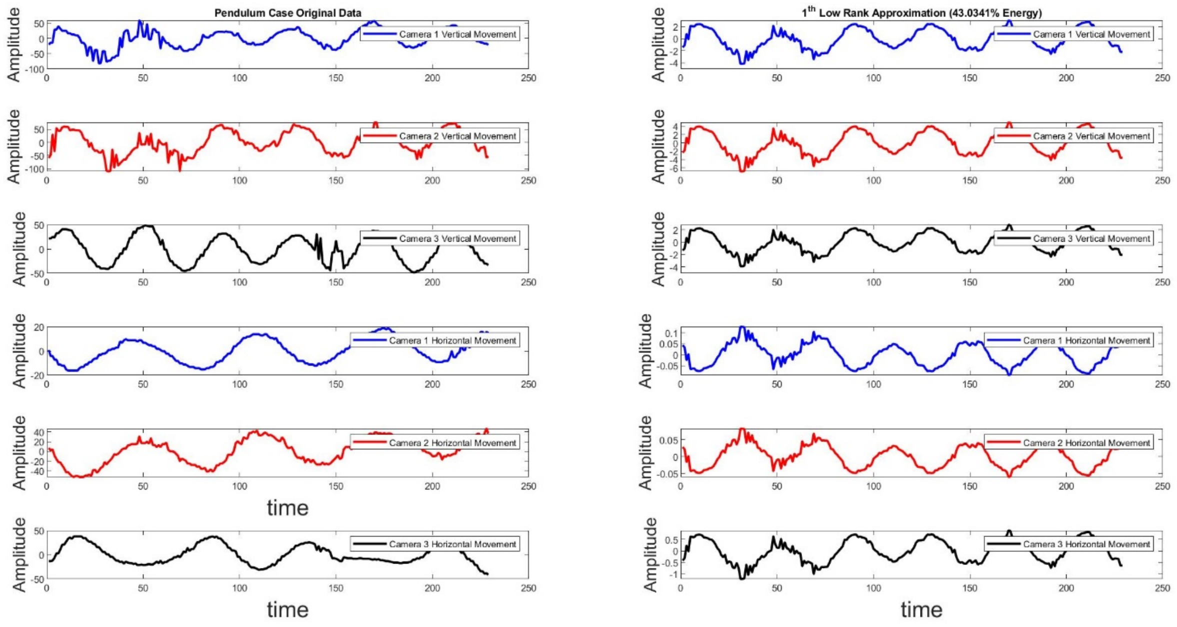


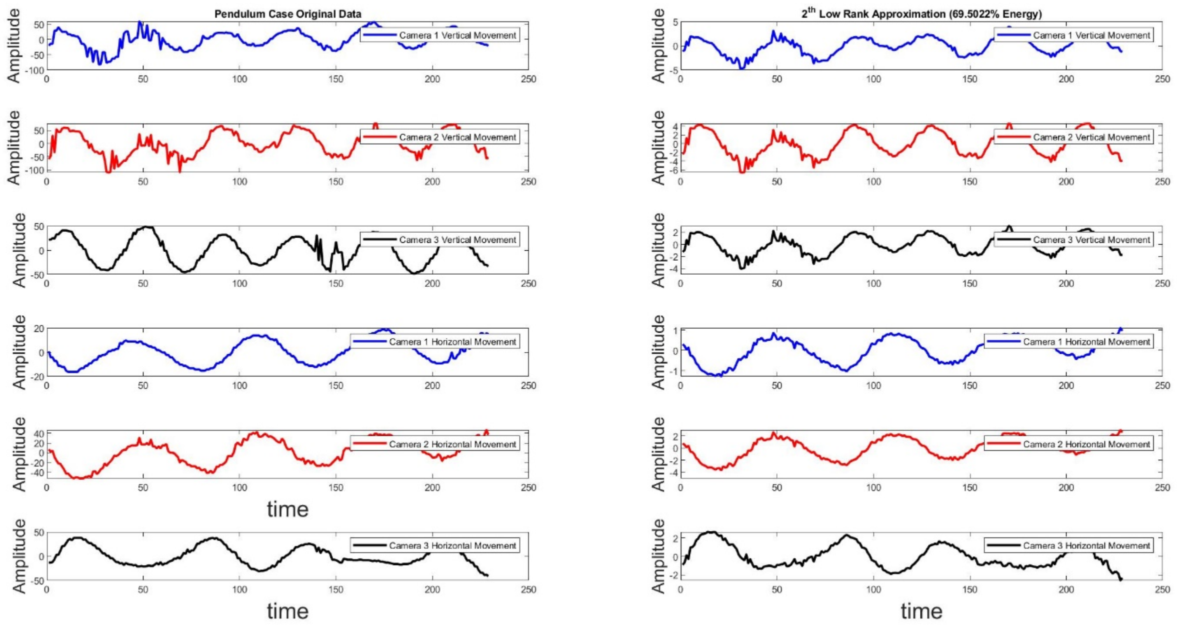
Figure 7. Vertical and horizontal movement of mass in pendulum case

Figure 8 presents the 1st ,2nd and 3rd low-rank approximations of the ideal case original data. As can be seen in Figure 6. (a), the first approximation captures only 42% of the original data, implying that although the 1st dimension of the unitary matrix $\boldsymbol{U}$ of $\boldsymbol{X}$ contains a significant portion of the original data set, others are expected to as well. In Figure 6. (b), the 2nd low-rank approximation is seen. Since the percentage of captured data increased by ~26%, sizable compared to the 42% captured in the 1st approximation, one can conclude that the first two vectors in $\boldsymbol{U}$ contributes significantly to $\boldsymbol{X}$. From the second to third low-rank approximation, ~13% is captured. This implies that in fact, only two cameras were needed to capture the motion of the spring-mass system to gain similar insight into the system as 6 cameras do.

(a)


Pendulum Case Original Data
1th Low Rank Approximation (43.0341% Energy)
Camera 1 Vertical Movement
Camera 2 Vertical Movement
Camera 3 Vertical Movement
Camera 1 Horizontal Movement
Camera 2 Horizontal Movement
Camera 3 Horizontal Movement
Amplitude
time

(b)


Pendulum Case Original Data
2th Low Rank Approximation (69.5022% Energy)
Camera 1 Vertical Movement
Camera 2 Vertical Movement
Camera 3 Vertical Movement
Camera 1 Horizontal Movement
Camera 2 Horizontal Movement
Camera 3 Horizontal Movement
Amplitude
time

(c)

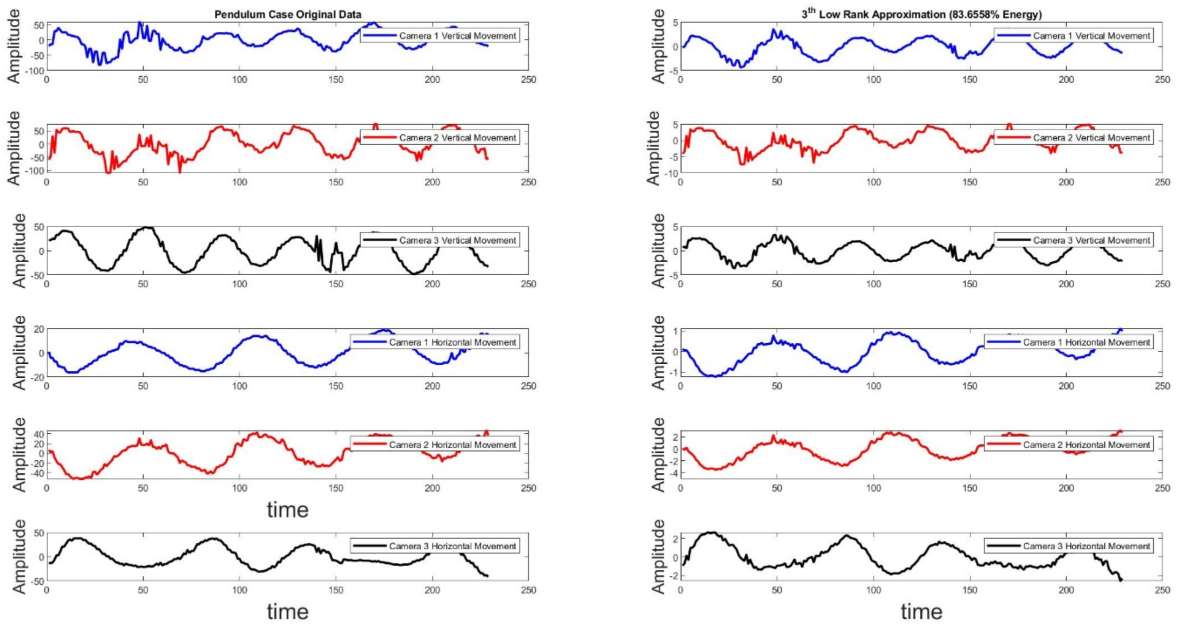


Figure 8. Comparison between original data and 1st, 2nd and 3rd low rank approximation

4. Camera 1, 2, and 3 in the shaking case all observe sinusoidal behavior in the vertical direction with high variance, as seen in Figure 9. The slight dips that are scattered throughout the plots can be explained by the brightest spots rotating out of sight, causing a different, darker point to be selected for analysis. In the horizontal direction, little variance is observed. This implies that the motion in this case is much similar to that in the ideal case.

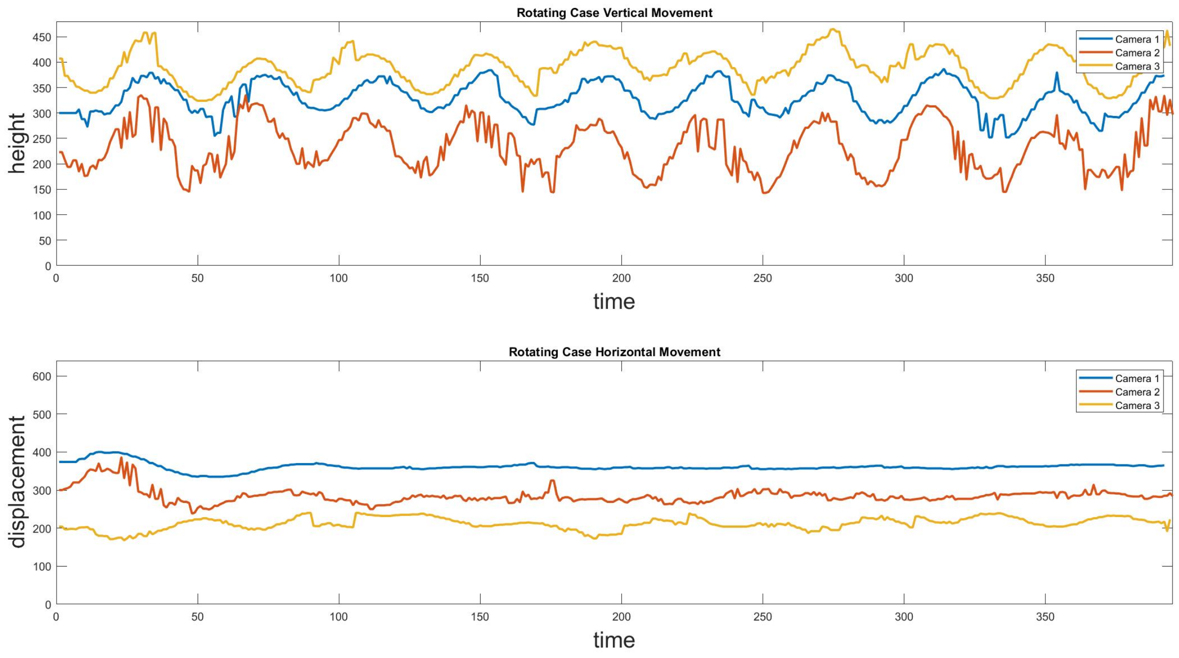
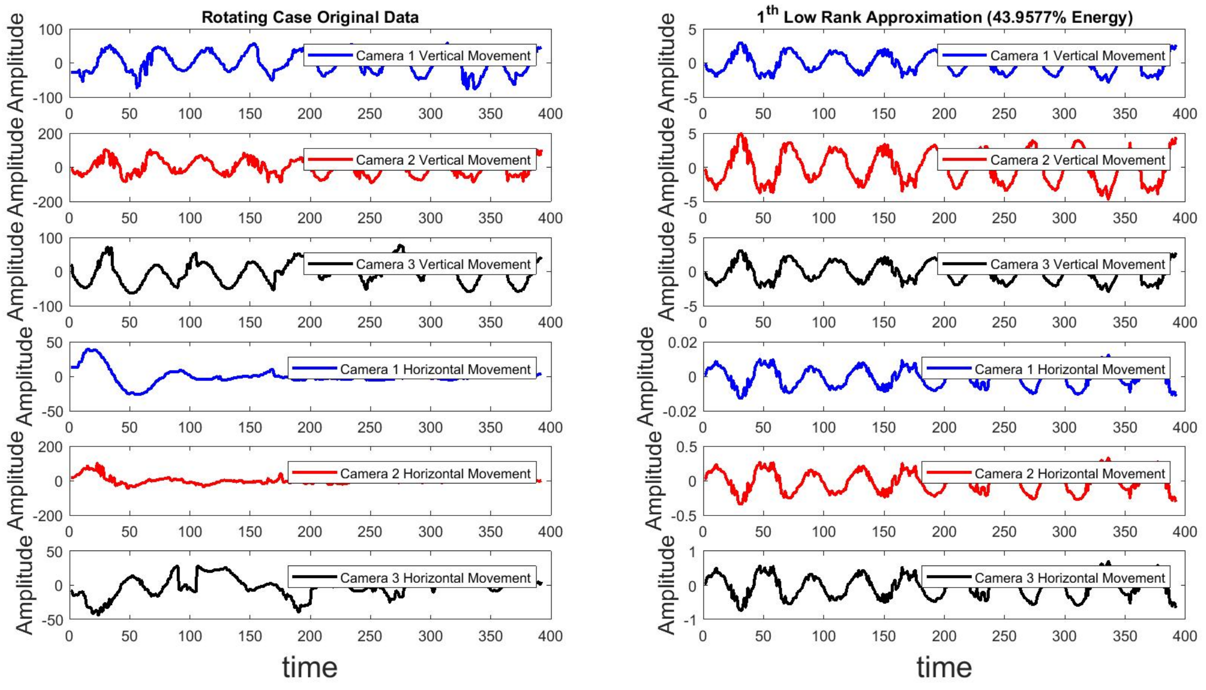


Figure 9. Vertical and horizontal movement of mass in rotating case

Figure 10 presents the 1st and 2nd low-rank approximations of the ideal case original data. As can be seen in Figure 10 (a), the first approximation captures 44% of the original data, implying that the 1st dimension of the unitary matrix $\boldsymbol{U}$ of $\boldsymbol{X}$ contains a significant portion of the original data set. In Figure 10. (b) the 2nd low-rank approximation is seen. Since the percentage of captured data only increased by ~16%, small compared to the 44% captured in the 1st approximation, one can conclude that the first vector in $\boldsymbol{U}$ is sufficient in representing $\boldsymbol{X}$. This implies that in fact, only one camera was needed to capture the motion of the spring-mass system to gain similar insight into the system as 6 cameras do.

(a)



(b)

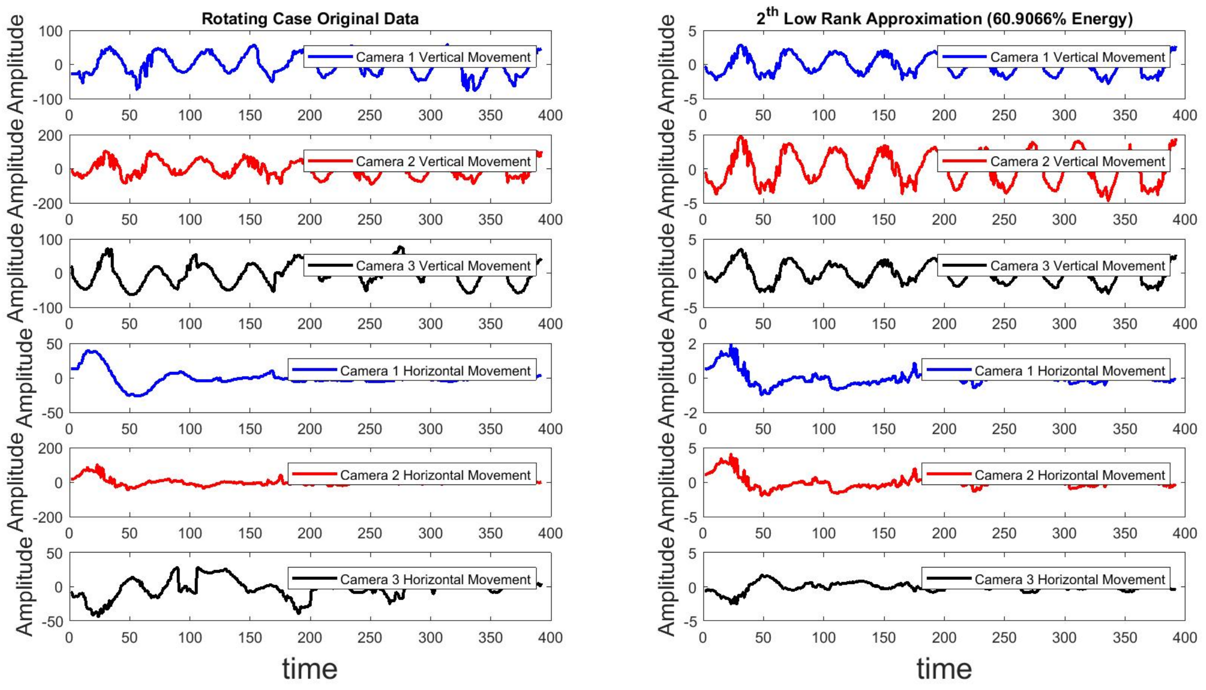


Figure 10. Comparison between original data and 1st and 2nd low rank approximation

V. Conclusion

Figure 11. presents the amount of energy that increases every time an additional component is added to approximates the data set $\boldsymbol{X}$. It can be concluded that the PCA may not be the best analysis method to determine the degree of freedom of the system due to the vague scenarios that arise in determining how many components are needed to find the best reduced matrix $\boldsymbol{Y}$**.** Looks like Newton will need to find better luck elsewhere!

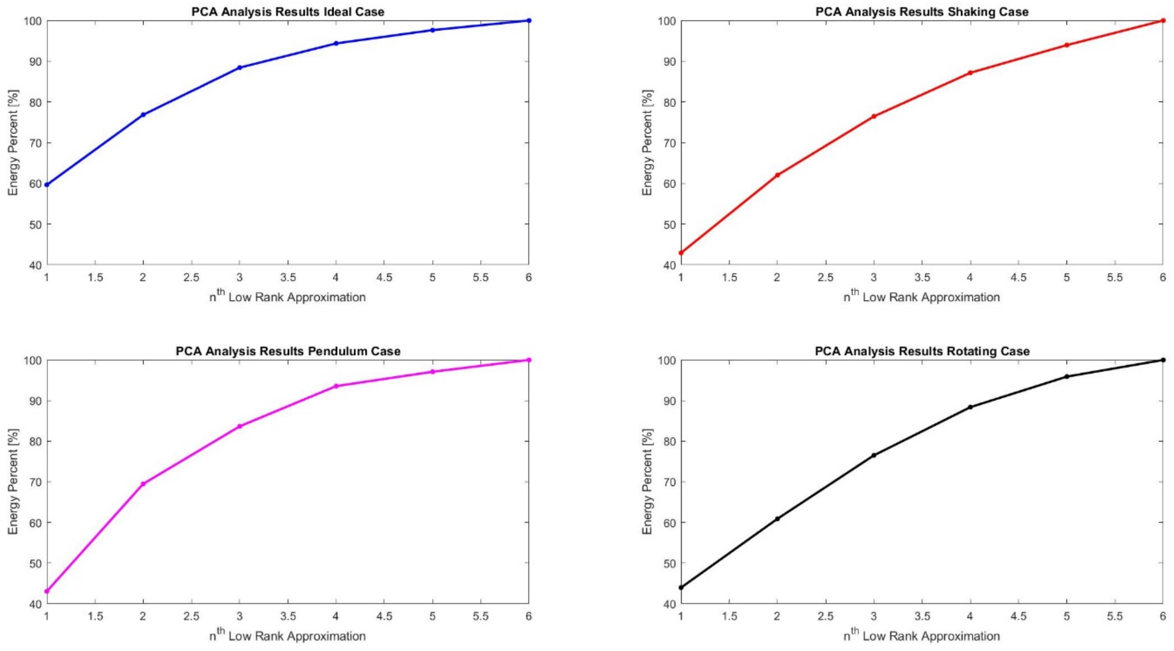


Figure 11. Increases in energy per additional component

Appendix A

- `rgb2gray(vidFrames(:,:,:,i))` – removes color dimension from `vidFrames`
- `ind2sub([yFrame xFrame],I)` – converts linear index `I` into coordinates specified by `[yFrame xFrame]`
- `pcolor(vidFramesGray), shading interp; colormap(gray); drawnow`—plots the spectrogram/image defined by `vidFramesGray`

Appendix B

Camera Reader Function

```
function [tFrame,xTrajectory,yTrajectory] = cameraReader(vidFrames,...
    shannonWidth, shannonHeight,...
    leftRightShift, topDownShift, threshold)
numFrames = size(vidFrames,4);
[yFrame, xFrame, rgbFrame, tFrame] = size(vidFrames);
tFrame = 1:tFrame;
darkFilter = zeros(xFrame, yFrame);
darkFilter((xFrame-
shannonWidth)/2+1+leftRightShift:xFrame/2+shannonWidth/2+leftRightShift,..
.
    (yFrame-
shannonHeight)/2+1+topDownShift:(yFrame+shannonHeight)/2+topDownShift)...
    = ones(shannonWidth, shannonHeight);
xTrajectory = [];
yTrajectory = [];
for i = 1:numFrames
    vidFramesGray = double(rgb2gray((vidFrames(:,:,:,i))));
    toBeIgnored = find(vidFramesGray < threshold);
    vidFramesGray(toBeIgnored) = 0;
    filteredFrame = vidFramesGray.*(darkFilter');
    [M,I] = max(filteredFrame(:));
    [mapY, mapX] = ind2sub([yFrame xFrame],I);
    xTrajectory = [xTrajectory mapX];
    yTrajectory = [yTrajectory mapY];

    %pcaData = [pcaData; mapX; mapY];
    %pcolor(filteredFrame), shading interp
    %colormap(gray); drawnow
end
end
```

Low Rank Approximation Function

```
function [projection] = pcProjection(X, n)
[U,S,V] = svd(X'/sqrt(length(X)-1));
component = (U(:,1)*S(1,1)*V(:,1)')';
projection = component;
for i = 2:n
    component = (U(:,i)*S(i,i)*V(:,i)')';
    projection = projection+component;

end
end
```

```
Main Program

%% Ideal data
clear all; close all; clc
load cam1_1.mat;
load cam2_1.mat;
load cam3_1.mat;
%% shaking data
load cam1_2.mat
load cam2_2.mat
load cam3_2.mat
%% x displacement
load cam1_3.mat
load cam2_3.mat
load cam3_3.mat
%% rotating
load cam1_4.mat
load cam2_4.mat
load cam3_4.mat
%% data shift functions
tShift = @(tEnd, shift) tEnd(shift+1:end)-shift;
xyShift = @(dataSet, shift) dataSet(shift+1:end);
%% Ideal case

[t1_1, x1_1, y1_1] = cameraReader(vidFrames1_1, 250, 150, 50, 70, 240);

[t2_1, x2_1, y2_1] = cameraReader(vidFrames2_1, 120, 280, -20, 0, 240);

[t3_1, x3_1, y3_1] = cameraReader(vidFrames3_1, 210, 130, 50, 50, 240);

t2_1 = tShift(t2_1,10);
y2_1 = xyShift(y2_1,10);
x2_1 = xyShift(x2_1,10);

tLength1 = 1:min([t1_1(end), t2_1(end), t3_1(end)]);

figure(1)
subplot(2,1,1)
plot(t1_1, y1_1,'LineWidth', 2); hold on
plot(t2_1, y2_1,'LineWidth', 2)
plot(t3_1, x3_1,'LineWidth', 2)
title('Ideal Case Vertical Movement')
xlabel('time','FontSize',20)
ylabel('height','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_1,t2_1,t3_1])])
ylim([0 480])

subplot(2,1,2)
plot(t1_1, x1_1,'LineWidth', 2); hold on
plot(tShift(t2_1,10), xyShift(x2_1,10),'LineWidth', 2)
plot(t3_1,y3_1,'LineWidth', 2)
title('Ideal case Horizontal Movement')
xlabel('time','FontSize',20)
ylabel('displacement','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_1,tShift(t2_1,10),t3_1])])
ylim([0 640])

%% shaking case
%close all;
[t1_2, x1_2, y1_2] = cameraReader(vidFrames1_2, 210, 250, 50, 50, 240);

[t2_2, x2_2, y2_2] = cameraReader(vidFrames2_2, 300, 400, 0, 0, 250);

[t3_2, x3_2, y3_2] = cameraReader(vidFrames3_2, 300, 150, 50, 50, 240);

t2_2 = tShift(t2_2,25);
y2_2 = xyShift(y2_2,25);
x2_2 = xyShift(x2_2,25);
t3_2 = tShift(t3_2,5);
y3_2 = xyShift(y3_2,5);
x3_2 = xyShift(x3_2,5);

figure(2)
subplot(2,1,1)
plot(t1_2, y1_2,'LineWidth', 2); hold on
plot(t2_2, y2_2,'LineWidth', 2)
plot(t3_2,x3_2,'LineWidth', 2)
title('Shaking Case Vertical Movement')
xlabel('time','FontSize',20)
ylabel('height','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_2,t2_2,t3_2])])
ylim([0 480])

subplot(2,1,2)
plot(t1_2, x1_2,'LineWidth', 2); hold on
plot(t2_2, x2_2,'LineWidth', 2)
plot(t3_2, y3_2,'LineWidth', 2)
title('Shaking Case Horizontal Movement')
xlabel('time','FontSize',20)
ylabel('displacement','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_2,t2_2,t3_2])])
ylim([0 640])
%% x displacement
clc
[t1_3, x1_3, y1_3] = cameraReader(vidFrames1_3, 120, 350, 0, 0, 250);

[t2_3, x2_3, y2_3] = cameraReader(vidFrames2_3, 200, 350, 0, 0, 250);
```

```
[t3_3, x3_3, y3_3] = cameraReader(vidFrames3_3, 350, 200, 0, 0, 250);

t1_3 = tShift(t1_3,10);
y1_3 = xyShift(y1_3,10);
x1_3 = xyShift(x1_3,10);
%PCA1_3 = pcaShift(PCA1_3,10);
t2_3 = tShift(t2_3,35);
y2_3 = xyShift(y2_3,35);
x2_3 = xyShift(x2_3,35);

figure(3)
subplot(2,1,1)
plot(t1_3, y1_3,'LineWidth', 2); hold on
plot(t2_3, y2_3,'LineWidth', 2)
plot(t3_3, x3_3,'LineWidth', 2)
title('Pendulum Case Vertical Movement')
xlabel('time','FontSize',20)
ylabel('height','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_3,t2_3,t3_3])])
ylim([0 480])

subplot(2,1,2)
plot(t1_3, x1_3,'LineWidth', 2); hold on
plot(t2_3, x2_3,'LineWidth', 2)
plot(tShift(t3_3,25), xyShift(y3_3,25),'LineWidth', 2)
title('Pendulum Case Horizontal Movement')
xlabel('time','FontSize',20)
ylabel('displacement','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_3,t2_3,t3_3])])
ylim([0 640])

%% rotating
%close all; clc
[t1_4, x1_4, y1_4] = cameraReader(vidFrames1_4, 150, 350, 80, 60, 240);

[t2_4, x2_4, y2_4] = cameraReader(vidFrames2_4, 250, 200, 0, 0, 250);

[t3_4, x3_4, y3_4] = cameraReader(vidFrames3_4, 250, 100, 70, -50, 230);

t2_4 = tShift(t2_4,10);
y2_4 = xyShift(y2_4,10);
x2_4 = xyShift(x2_4,10);

figure(4)
subplot(2,1,1)
plot(t1_4, y1_4,'LineWidth', 2); hold on
plot(t2_4, y2_4,'LineWidth', 2)
plot(t3_4, x3_4,'LineWidth', 2)
title('Rotating Case Vertical Movement')
```

```
xlabel('time','FontSize',20)
ylabel('height','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_4,t2_4,t3_4])])
ylim([0 480])

subplot(2,1,2)
plot(t1_4, x1_4,'LineWidth', 2); hold on
plot(t2_4, x2_4,'LineWidth', 2)
plot(t3_4,y3_4,'LineWidth', 2)
title('Rotating Case Horizontal Movement')
xlabel('time','FontSize',20)
ylabel('displacement','FontSize',20)
legend('Camera 1','Camera 2','Camera 3')
xlim([0 max([t1_4,t2_4,t3_4])])
ylim([0 640])

%% PCA Analysis -- Ideal Case
clc
X1 =
[x1_1(tLength1);y1_1(tLength1);x2_1(tLength1);y2_1(tLength1);x3_1(tLength1
);y3_1(tLength1)];
[m1,n1]=size(X1); % compute data size
mn1=mean(X1,2); % compute mean for each row
X1=X1-repmat(mn1,1,n1); % subtract mean
Cx1=X1*X1'; % covariance
[V1,D1]=eig(Cx1); % eigenvectors(V)/eigenvalues(D)
lambda1=diag(D1); % get eigenvalues
[dummy,m_arrange1]=sort(-1*lambda1); % sort in decreasing order
lambda1=lambda1(m_arrange1);
V1=V1(:,m_arrange1);
Y1=V1'*X1; % produce the principal components projection
lambdaTotal1 = sum(lambda1.^0.5);

figure(5)
subplot(6,2,1)
plot(tLength1, X1(2,:),'b','LineWidth',2)
title('Ideal Case Original Data')
legend('Camera 1 Vertical Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,3)
plot(tLength1, X1(4,:),'r','LineWidth',2)
legend('Camera 2 Vertical Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,5)
plot(tLength1, X1(5,:),'k','LineWidth',2)
legend('Camera 3 Vertical Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,7)
plot(tLength1, X1(1,:),'b','LineWidth',2)
legend('Camera 1 Horizontal Movement')
```

```
ylabel('Amplitude','FontSize',15)
subplot(6,2,9)
plot(tLength1, X1(3,:),'r','LineWidth',2)
legend('Camera 2 Horizontal Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,11)
plot(tLength1, X1(6,:),'k','LineWidth',2)
legend('Camera 3 Horizontal Movement')
ylabel('Amplitude','FontSize',15)
xlabel('time','FontSize',20)
percentages1 = [];

for i = 1:6
    percentE1 = sum(lambda1(1:i).^0.5)/lambdaTotal1;
    percentages1 = [percentages1 percentE1];
    projection1 = pcProjection(X1,i);
    subplot(6,2,2)
    plot(tLength1, projection1(2,:),'b','LineWidth',2)
    title(strcat(num2str(i),'^t^h Low Rank Approximation
(',num2str(100*percentE1),'% Energy)'))
    legend('Camera 1 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,4)
    plot(tLength1, projection1(4,:),'r','LineWidth',2)
    legend('Camera 2 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,6)
    plot(tLength1, projection1(5,:),'k','LineWidth',2)
    legend('Camera 3 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,8)
    plot(tLength1, projection1(1,:),'b','LineWidth',2)
    ylabel('Amplitude','FontSize',15)
    legend('Camera 1 Horizontal Movement')
    subplot(6,2,10)
    plot(tLength1, projection1(3,:),'r','LineWidth',2)
    ylabel('Amplitude','FontSize',15)
    legend('Camera 2 Horizontal Movement')
    subplot(6,2,12)
    plot(tLength1, projection1(6,:),'k','LineWidth',2)
    legend('Camera 3 Vertical Movement')
    xlabel('time','FontSize', 20)
    ylabel('Amplitude','FontSize',15)
    pause
end
%% PCA Analysis -- shaking case
clc
tLength2 = 1:min([t1_2(end), t2_2(end), t3_2(end)]);
X2 =
[x1_2(tLength2);y1_2(tLength2);x2_2(tLength2);y2_2(tLength2);x3_2(tLength2
);y3_2(tLength2)];
```

```
[m2,n2]=size(X2);  % compute data size
mn2=mean(X2,2); % compute mean for each row
X2=X2-repmat(mn2,1,n2); % subtract mean
Cx2=X2*X2'; % covariance
[V2,D2]=eig(Cx2); % eigenvectors(V)/eigenvalues(D)
lambda2=diag(D2); % get eigenvalues
[dummy,m_arrange2]=sort(-1*lambda2); % sort in decreasing order
lambda2=lambda2(m_arrange2);
V2=V2(:,m_arrange2);
Y2=V2'*X2; % produce the principal components projection
lambdaTotal2 = sum(lambda2.^0.5);


figure(6)
subplot(6,2,1)
plot(tLength2, X2(2,:),'b','LineWidth',2)
title('Shaking Case Original Data')
legend('Camera 1 Vertical Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,3)
plot(tLength2, X2(4,:),'r','LineWidth',2)
legend('Camera 2 Vertical Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,5)
plot(tLength2, X2(5,:),'k','LineWidth',2)
legend('Camera 3 Vertical Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,7)
plot(tLength2, X2(1,:),'b','LineWidth',2)
legend('Camera 1 Horizontal Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,9)
plot(tLength2, X2(3,:),'r','LineWidth',2)
legend('Camera 2 Horizontal Movement')
ylabel('Amplitude','FontSize',15)
subplot(6,2,11)
plot(tLength2, X2(6,:),'k','LineWidth',2)
legend('Camera 3 Horizontal Movement')
xlabel('time','FontSize',20)
ylabel('Amplitude','FontSize',15)
percentages2 = [];

for i = 1:6
    projection2 = pcProjection(X2,i);
    percentE2 = sum(lambda2(1:i).^0.5)/lambdaTotal2;
    percentages2 = [percentages2 percentE2];
    subplot(6,2,2)
    plot(tLength2, projection2(2,:),'b','LineWidth',2)
    title(strcat(num2str(i),'^t^h Low Rank Approximation
(',num2str(100*percentE2),'% Energy)'))
    legend('Camera 1 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,4)
    plot(tLength2, projection2(4,:),'r','LineWidth',2)
    legend('Camera 2 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,6)
    plot(tLength2, projection2(5,:),'k','LineWidth',2)
    legend('Camera 3 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,8)
    plot(tLength2, projection2(1,:),'b','LineWidth',2)
    legend('Camera 1 Horizontal Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,10)
    plot(tLength2, projection2(3,:),'r','LineWidth',2)
    legend('Camera 2 Horizontal Movement')
    ylabel('Amplitude','FontSize',15)
    subplot(6,2,12)
    plot(tLength2, projection2(6,:),'k','LineWidth',2)
    legend('Camera 3 Horizontal Movement')
c
    pause
end
%% PCA Analysis -- Pendulum Case
tLength3 = 1:min([t1_3(end), t2_3(end), t3_3(end)]);
X3 =
[x1_3(tLength3);y1_3(tLength3);x2_3(tLength3);y2_3(tLength3);x3_3(tLength3
);y3_3(tLength3)];

[m3,n3]=size(X3); % compute data size
mn3=mean(X3,2); % compute mean for each row
X3=X3-repmat(mn3,1,n3); % subtract mean
Cx3=X3*X3'; % covariance
[V3,D3]=eig(Cx3); % eigenvectors(V)/eigenvalues(D)
lambda3=diag(D3); % get eigenvalues
[dummy,m_arrange3]=sort(-1*lambda3); % sort in decreasing order
lambda3=lambda3(m_arrange3);
V3=V3(:,m_arrange3);
Y3=V3'*X3; % produce the principal components projection
lambdaTotal3 = sum(lambda3.^0.5);

figure(7)
subplot(6,2,1)
plot(tLength3, X3(2,:),'b','LineWidth',2)
title('Pendulum Case Original Data')
legend('Camera 1 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
subplot(6,2,3)
plot(tLength3, X3(4,:),'r','LineWidth',2)
legend('Camera 2 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
```

```
subplot(6,2,5)
plot(tLength3, X3(5,:),'k','LineWidth',2)
legend('Camera 3 Vertical Movement')
    ylabel('Amplitude','FontSize',15)
subplot(6,2,7)
plot(tLength3, X3(1,:),'b','LineWidth',2)
legend('Camera 1 Horizontal Movement')
    ylabel('Amplitude','FontSize',15)
subplot(6,2,9)
plot(tLength3, X3(3,:),'r','LineWidth',2)
legend('Camera 2 Horizontal Movement')
    ylabel('Amplitude','FontSize',15)
    xlabel('time','FontSize',20)
subplot(6,2,11)
plot(tLength3, X3(6,:),'k','LineWidth',2)
legend('Camera 3 Horizontal Movement')
ylabel('Amplitude','FontSize',15)
xlabel('time','FontSize',20)
percentages3 = [];

for i = 1:6
    projection3 = pcProjection(X3,i);
    percentE3 = sum(lambda3(1:i).^0.5)/lambdaTotal3;
    percentages3 = [percentages3 percentE3];
    subplot(6,2,2)
    plot(tLength3, projection3(2,:),'b','LineWidth',2)
    title(strcat(num2str(i),'^t^h Low Rank Approximation
(',num2str(100*percentE3),'% Energy)'))
    legend('Camera 1 Vertical Movement')
        ylabel('Amplitude','FontSize',15)
    subplot(6,2,4)
    plot(tLength3, projection3(4,:),'r','LineWidth',2)
    legend('Camera 2 Vertical Movement')
        ylabel('Amplitude','FontSize',15)
    subplot(6,2,6)
    plot(tLength3, projection3(5,:),'k','LineWidth',2)
    legend('Camera 3 Vertical Movement')
        ylabel('Amplitude','FontSize',15)
    subplot(6,2,8)
    plot(tLength3, projection3(1,:),'b','LineWidth',2)
    legend('Camera 1 Horizontal Movement')
        ylabel('Amplitude','FontSize',15)
    subplot(6,2,10)
    plot(tLength3, projection3(3,:),'r','LineWidth',2)
    legend('Camera 2 Horizontal Movement')
        ylabel('Amplitude','FontSize',15)
    subplot(6,2,12)
    plot(tLength3, projection3(6,:),'k','LineWidth',2)
    legend('Camera 3 Horizontal Movement')
    xlabel('time','FontSize',20)
    ylabel('Amplitude','FontSize',15)
    pause
end
%% PCA Analysis -- Rotating Case
tLength4 = 1:min([t1_4(end), t2_4(end), t3_4(end)]);
X4 =
[x1_4(tLength4);y1_4(tLength4);x2_4(tLength4);y2_4(tLength4);x3_4(tLength4
);y3_4(tLength4)];

[m4,n4]=size(X4); % compute data size
mn4=mean(X4,2); % compute mean for each row
X4=X4-repmat(mn4,1,n4); % subtract mean
Cx4=X4*X4'; % covariance
[V4,D4]=eig(Cx4); % eigenvectors(V)/eigenvalues(D)
lambda4=diag(D4); % get eigenvalues
[dummy,m_arrange4]=sort(-1*lambda4); % sort in decreasing order
lambda4=lambda4(m_arrange4);
V4=V4(:,m_arrange4);
Y4=V4'*X4; % produce the principal components projection
lambdaTotal4 = sum(lambda4.^0.5);
percentages4 = [];

figure(8)
subplot(6,2,1)
plot(tLength4, X4(2,:),'b','LineWidth',2)
title('Rotating Case Original Data')
legend('Camera 1 Vertical Movement')
subplot(6,2,3)
plot(tLength4, X4(4,:),'r','LineWidth',2)
legend('Camera 2 Vertical Movement')
subplot(6,2,5)
plot(tLength4, X4(5,:),'k','LineWidth',2)
legend('Camera 3 Vertical Movement')
subplot(6,2,7)
plot(tLength4, X4(1,:),'b','LineWidth',2)
legend('Camera 1 Horizontal Movement')
subplot(6,2,9)
plot(tLength4, X4(3,:),'r','LineWidth',2)
legend('Camera 2 Horizontal Movement')
subplot(6,2,11)
plot(tLength4, X4(6,:),'k','LineWidth',2)
legend('Camera 3 Horizontal Movement')

for i = 1:6
    projection4 = pcProjection(X4,i);
    percentE4 = sum(lambda4(1:i).^0.5)/lambdaTotal4;
    percentages4 = [percentages4 percentE4];
    subplot(6,2,2)
    plot(tLength4, projection4(2,:),'b','LineWidth',2)
    title(strcat(num2str(i),'^t^h Low Rank Approximation
(',num2str(100*percentE4),'% Energy)'))
    legend('Camera 1 Vertical Movement')
    subplot(6,2,4)
    plot(tLength4, projection4(4,:),'r','LineWidth',2)
    legend('Camera 2 Vertical Movement')
    subplot(6,2,6)
    plot(tLength4, projection4(5,:),'k','LineWidth',2)
    legend('Camera 3 Vertical Movement')
    subplot(6,2,8)
    plot(tLength4, projection4(1,:),'b','LineWidth',2)
    legend('Camera 1 Horizontal Movement')
    subplot(6,2,10)
    plot(tLength4, projection4(3,:),'r','LineWidth',2)
    legend('Camera 2 Horizontal Movement')
    subplot(6,2,12)
    plot(tLength4, projection4(6,:),'k','LineWidth',2)
    legend('Camera 3 Horizontal Movement')
    pause
end
%% plot PCA results
% lambda1 = lambda1/max(lambda1);
% lambda2 = lambda2/max(lambda2);
% lambda3 = lambda3/max(lambda3);
% lambda4 = lambda4/max(lambda4);

figure(9)
subplot(2,2,1)
plot(1:length(lambda1),100*percentages1,'b.-
','LineWidth',2,'MarkerSize',15);
title('PCA Analysis Results Ideal Case')
ylabel('Energy Percent [%]')
xlabel('n^t^h Low Rank Approximation')
ylim([40 100])

subplot(2,2,2)
plot(1:length(lambda2),100*percentages2,'r.-
','LineWidth',2,'MarkerSize',15);
title('PCA Analysis Results Shaking Case')
ylabel('Energy Percent [%]')
xlabel('n^t^h Low Rank Approximation')
ylim([40 100])

subplot(2,2,3)
plot(1:length(lambda3),100*percentages3,'m.-
','LineWidth',2,'MarkerSize',15);
title('PCA Analysis Results Pendulum Case')
ylabel('Energy Percent [%]')
xlabel('n^t^h Low Rank Approximation')
ylim([40 100])

subplot(2,2,4)
plot(1:length(lambda4),100*percentages4,'k.-
','LineWidth',2,'MarkerSize',15);
```

```
title('PCA Analysis Results Rotating Case')
ylabel('Energy Percent [%]')
xlabel('n^t^h Low Rank Approximation')
ylim([40 100])
```

```
Testing Program used to determine Shannon window

%clear all; close all; clc
% vidFrames = load('cam1_1.mat'); % 250, 150, 50, 70, 250
% vidFrames = load('cam2_1.mat'); % 120, 280, -20, 0, 250
% vidFrames = load('cam3_1.mat'); % 210, 130, 50, 50, 250

% vidFrames = load('cam1_2.mat'); % 210, 250, 50, 50, 240
vidFrames = load('cam2_2.mat'); % 300, 400, 0 ,0, 250
% vidFrames = load('cam3_2.mat'); % 300, 150, 50, 50 250 horizontal

% vidFrames = load('cam1_3.mat'); % 150, 210, 0, 100
% vidFrames = load('cam2_3.mat'); % 200, 300, -20, 50
% vidFrames = load('cam3_3.mat'); % 300, 200, 30, 0 diagonal

% vidFrames = load('cam1_4.mat'); % 150, 350, 80, 60
% vidFrames = load('cam2_4.mat'); % 200, 250, 0, 0
% vidFrames = load('cam3_4.mat'); % 200, 150, 70, -50
vidFrames = vidFrames.vidFrames2_2;
%%
% close all;
numFrames = size(vidFrames, 4);

[yFrame, xFrame, rgbFrame, tFrame] = size(vidFrames);
darkFilter = zeros(xFrame, yFrame);
shannonWidth = 200;
shannonHeight = 150;
leftRightShift = 0;
topDownShift = 80;
darkFilter(xFrame/2-
shannonWidth/2+1+leftRightShift:xFrame/2+shannonWidth/2+leftRightShift,...
    (yFrame-
shannonHeight)/2+1+topDownShift:(yFrame+shannonHeight)/2+topDownShift)...
    = ones(shannonWidth, shannonHeight);



%%
threshold = 250;
xTrajectory = [];
yTrajectory = [];
PCA = [];

for i = 1:numFrames
    vidFramesGray = double(rgb2gray((vidFrames(:,:,:,i))));
    toBeIgnored = find(vidFramesGray<threshold);
    %vidFramesGray(toBeIgnored) = 0;
    filteredFrame = vidFramesGray.*(darkFilter');
    [M,I] = max(filteredFrame(:));
    [mapY, mapX] = ind2sub([yFrame xFrame],I);
    xTrajectory = [xTrajectory mapX];
    yTrajectory = [yTrajectory mapY];
    PCA = [PCA; mapX; mapY];
    %pcolor(filteredFrame), shading interp
    %colormap(gray); drawnow
end
close all;
figure(10)
% subplot(1,3,1)
pcolor(vidFramesGray), shading interp
colormap(gray); drawnow
% title('a) Image','FontSize',20)
% subplot(1,3,2)
% pcolor(darkFilter'),shading interp
% colormap(gray); drawnow
% title('b) Shannon Window','FontSize',20)
% subplot(1,3,3)
% pcolor(filteredFrame), shading interp
% colormap(gray); drawnow
% title('c) Filtered Image','FontSize',20)
% figure(11)
% pcolor(filteredFrame), shading interp
% colormap(gray); drawnow
% title('Filtered Image','FontSize',20)

%%
plot(1:tFrame,yTrajectory)
```