PCA: Paint Cans Acrobatics

Avery Pong – AMATH 482 – 02.25.2019

0. Overview

Seemingly multidimensional systems are often simpler than they appear. Here, three cameras observed a paint can oscillating in one, two, and three dimensions (degrees of freedom), with variable amounts of noise introduced by camera shake. The position of each white can was taken with respect to the pixilation grid of the cameras, and Principle Component Analysis (PCA) was applied to the collected position data to determine the true—or minimum—number of dimensions in which the system could be described. Pitfalls to the PCA are discussed, as are Fourier Transforms that can help clarify the picture rendered by the principle components.

I. Introduction: Can't Catch These Cans

Four spring-mass displacement experiments were undertaken: the first involved a paint can that was dropped and only moved in the z direction; the second experiment repeated the first, but with camera shake to introduce noise into the system; the third involved a paint can that was dropped off center such that there was pendulum motion in a y-z plane, along with the original oscillation along the z axis; the final, fourth experiment involved oscillation with rotation about an axis, thus exhibiting motion in the x, y, and z directions. Experimental cans had a bright light attached, which allowed better tracking of the can. Data from three cameras was taken from each experiment—known as case 1, 2, 3, and 4 respectively from here. Each camera recorded each motion using a two dimensional representation. For case 1, which only possessed a single degree of freedom, this mean that five redundant measurements existed, since only one measurement is truly needed to record the single dimensional data. In case 4, where there were three degrees of freedom, at least two cameras would have been needed to provide the three measurements necessary to characterize the system. Nonetheless, in all the cases, at least 3 redundancies were introduced in the data describing the systems. To this end, a PCA algorithm was developed to identify the experimental signals—the axes—with the greatest observable data variance, and thus identify the degrees of freedom observed in each of the cases. It was thus hypothesized that PCA analysis could ascertain the correct number of degrees of freedom in each of the four systems.

II. Theoretical Background: the PCA

In the case of the paint can videos, discerning underlying signals from noise and overlapping redundancy proved ripe for application of the PCA, an algorithm that performs dimensionality reduction of a dataset onto its essential variable axes. PCA, at a high level, identifies signals with maximal variance and suggests which sets of data may be statistically dependent. Some preliminary set up, however, is required in order to best use the SVD.

In each of the four cases, three cameras each described the system in two dimensions. All six of the total measures was collected into a matrix, which facilitated the comparison of each of the measured streams of data with each other. This was accomplished by calculating the covariance matrix, which describes overlap between each of the individual data streams by means of taking successive dot products with all the others:

$$C_x = \frac{1}{n-1} X * X^T$$

Here, covariance matrix C_x equals the matrix of data X multiplied by its transpose, such that each vector of data gets compared to every other. Multiplication by a correction factor follows, where n is the number of measurements taken. Importantly, the covariance matrix is a square matrix whose diagonal terms represent the variance of each measurement of data, since the matrix multiples each vector of data with itself. Off-diagonal terms represent covariances between different measurement vectors, and thus describes the overlap that some camera's data might have with other cameras. As a result, large off-diagonal terms indicate redundancies in the data, while smaller diagonal values imply low statistical dependencies between data.

Thus the covariance matrix gives a straightforward way to determine redundancies in the data set: if on-diagonal values are large, the data to which the correspond have the greatest variance and thus account for the majority of the dynamics of the system; secondly, if off-diagonal values are large, that gives a glaring clue that the data to which they correspond are highly statistically dependent.

With the covariance matrix in hand, we hope to find some optimal matrix transform that renders the six sets of data in a way that maximizes all diagonal terms and reduces all off-diagonal terms to zero. Finding this particular transformation would place the data in a frame of reference that removes all of the original redundancies, and finds the axes of the signals' of maximal variance.

The particular matrix that allows for this transformation derives from the singular value decomposition (SVD) of the data matrix X. SVD is a matrix decomposition that is guaranteed to exist for any matrix, and is comprised of the following:

1) Matrix $X \in \mathbb{C}^{mxn}$ has an singular value decomposition $X = U\Sigma V^*$, where
2) $U \in \mathbb{C}^{mxm}$ is a unitary matrix
3) $V \in \mathbb{C}^{nxn}$ is a unitary matrix
4) $\Sigma \in \mathbb{R}^{mxn}$ is a diagonal matrix with values in descending order

Overall, matrix X is broken down into constituent transformations, where V represents a rotation, Σ represents stretching by singular values of the matrix X, and U represents a final rotation in the basis space. In another light more useful for dimensionality reduction, we can say that matrix X can be reduced to a simple, easy to use diagonal matrix Σ when its range is expressed in terms of orthogonal basis vectors in U and the domain is expressed in terms of the orthogonal basis vectors in V. The new set of basis vectors act as maps to a coordinate system that optimally expresses X as a diagonal matrix of singular values represented in descending order. This important fact—that the SVD can diagonalize a matrix to its essential directional components— is what facilitates the realignment of the camera data set to the optimal coordinate basis that can render the optimal, diagonalized covariance matrix described above. This optimal matrix transform makes use of the U matrix of the data matrix:

$$Y = U * X$$

Where Y represents the raw data X that has been transformed into the basis in which its covariance matrix can be ideally expressed as a diagonal matrix. This means that data X has been moved to a basis that optimally expresses it terms of the axes of its dimensional essence, and has thus been transformed to a coordinate system of its principle components.

$$C_Y = \frac{1}{n-1} Y * Y^T$$

$$= \frac{1}{n-1} (UX)(UX)^T$$

$$= \frac{1}{n-1} U(XX^T)U$$

$$= \frac{1}{n-1} U^* U\Sigma^2 UU^* \quad \text{since} \quad XX^T U = U\Sigma^2$$

$$C_Y = \frac{1}{n-1}\Sigma^2$$

Thus, it's shows that the new covariance matrix C_Y of the transformed data Y is a diagonal matrix of singular values. In this optimal coordinate framework, all of the diagonal terms represent the variance of the data while the off diagonal terms are zero, implying that the coordinate system renders the data in a basis that removes the redundancy. The result renders a diagonal matrix Σ that describes the relative energy held in each dimension of data, in descending order. Thus, the PCA analysis in its entirety finally yields the relative importance, or energy, of each dimension within the data set.

III. Algorithm Implementation: SVD and FFT

The analysis required three major steps, which are explicitly described here. First, the data needed to be collected and rendered in MatLab by tracking the bright white pixels on the paint cans. Second, the data was transformed to its optimal coordinate basis by using the U unitary matrix of the SVD of the data matrix. Third, because the data we expect from the system is oscillatory, the Fourier transform was applied to all of the data rendered in the principle components to corroborate the PCA data.

Videos for each case were loaded as .mat structs. Each case study utilized three cameras that recorded the can for different amounts of time, so the analysis was undertaken using the lowest common number of data frames in order to generate a rectangular data matrix. The raw video was filmed in color, so during the read-in, all frames were converted to gray scale using `rgb2gray` so that data could foreseeable have a maximum value according to brightness. The contrast of each frame was increased using `imadjust()` to better facilitate the differentiation between the bright paint cans and the relatively dark background. Videos were subsequently cropped to focus on the portion of the video in which the paint can was oscillating. This step allowed for stray points of brightness that could skew tracking to be removed. The indices of the maximally bright pixels in the data matrix were then obtained: each frame had many maximally bright points on the can since many points registered as bright white (255/255), and thus the average index position of this mass was taken for further analysis. The x and y positions—with respect to the coordinates of the camera used to record the data—were then collected and stored in a larger data matrix `mat`. Each vector's mean was subtracted from the set, such that data could be properly represented as a displacement from the equilibrium spring position.

The singular value decomposition was then determined for each normalized data matrix using `SVD(data, econ)`. This rendered the U matrix that allowed for the transformation of the data matrix `mat` into a space in which it was represented along its principle components. At this juncture, knowing the preliminaries of the assignment, it was surprising when the one degree of freedom case 1 system showed two axes of high variation. In other words, the PCA algorithm would lead one to believe that there were two true dimensions of data, in other words two degrees of freedom. Upon inspection of the data, it became clear that this result derived from the fact that two of the sinusoids rendered from the data were in phase, while a third was out of phase. This was because two of the experimental videos were started at the same time, while the other was started with either a slight delay, or was commenced when the can was at a

different position along its oscillatory trajectory. This had the overall effect of producing two out of phase sine waves of the same frequency.

(Full disclosure I don't have enough of a math background as a biochemist to really say this concretely here) I imagine that because of the linear independence of sufficiently out of phase sinusoids, the PCA algorithm interprets them as completely different axes of movement. This was partially confirmed when all of the sinusoidal data were aligned in phase: phase alignment produced a single, strong variance reading in the first principle component and suggested the existence of only one degree of freedom. I was very tempted to go through the process of aligning all of the sinusoids by hand, for all data from all of the cameras, in all of the experimental cases. This would allow sinuisoids of the same frequency, which very likely corresponded to the same phenomenon varying on the same axis, to align. Moreover, this would also allow different sets of sinusoids that had different frequencies—likely because they might come from different physical phenomena, like z-direction falling or x-y pendulum motion—to differentiate themselves in the analysis, thus correctly manifesting on different principle components. This had the unwanted effect of relying on my biases having already seen the data, and thus this method became fairly unappealing.

Instead, because of the sinusoidal nature of the data, I applied the Fourier transform to the data in principle component space in a confirmatory last step, as a way to determine if out of phase sinusoids were truly representative of different signals, or if the signal was in a way derived from the same frequency source. One caveat of this analysis, especially in case 3 and 4, is that if the frequency of the z-direction motion and the x-y pendulum motion have the same frequency, two dimensions that are truly different would register as the same. However, this chance was small, because the horizontal displacement would have to match the up and down motion governed by the spring constant of in the experiment, and that likelihood, while extant, seemed small. Again, this was just a confirmatory step that corroborated the known dimensionality of the system. It also served as an avenue to uncover some of the draw backs of purely relying on the PCA.

IV. Computational Results: Marble Trajectory

The raw tracking data, principle component breakdown, and the frequency component analysis from each case is explored.

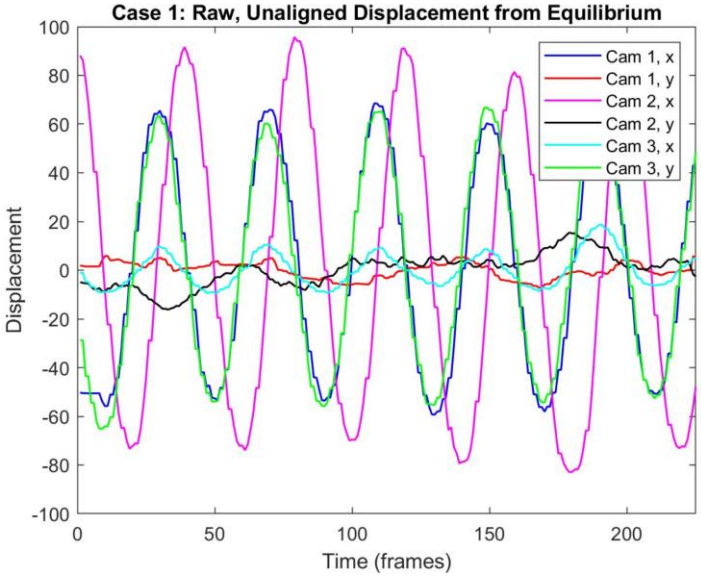

Figure 1. Case 1, Raw camera tracking data.

Figure 1 clearly shows two in phase sinusoids from camera 1 and 3, and also shows a sinusoid of commensurate frequency read in from camera 2. These were representative of the oscillatory movement in the z direction.

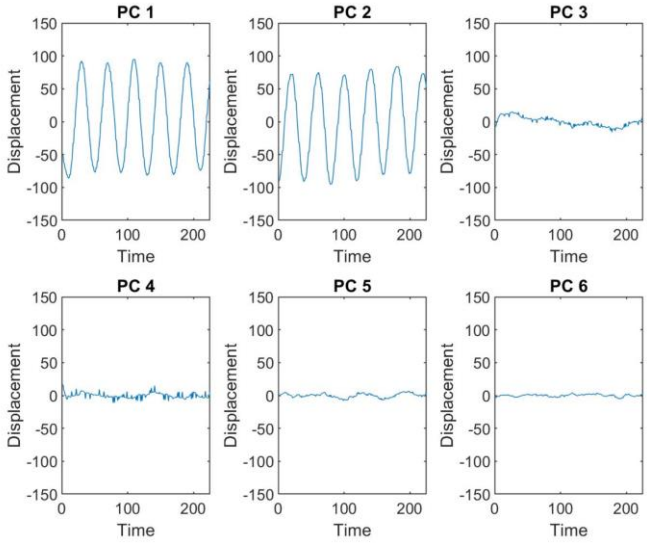

Figure 2. Principle component projections

Case 1 appears to be optimally represented along two principle components, leading one to believe that the system has two degrees of freedom. Data in Figure 2, however, is further processed using the Fourier transform to confirm that the PC1 and PC2 sinusoids have the same frequency.

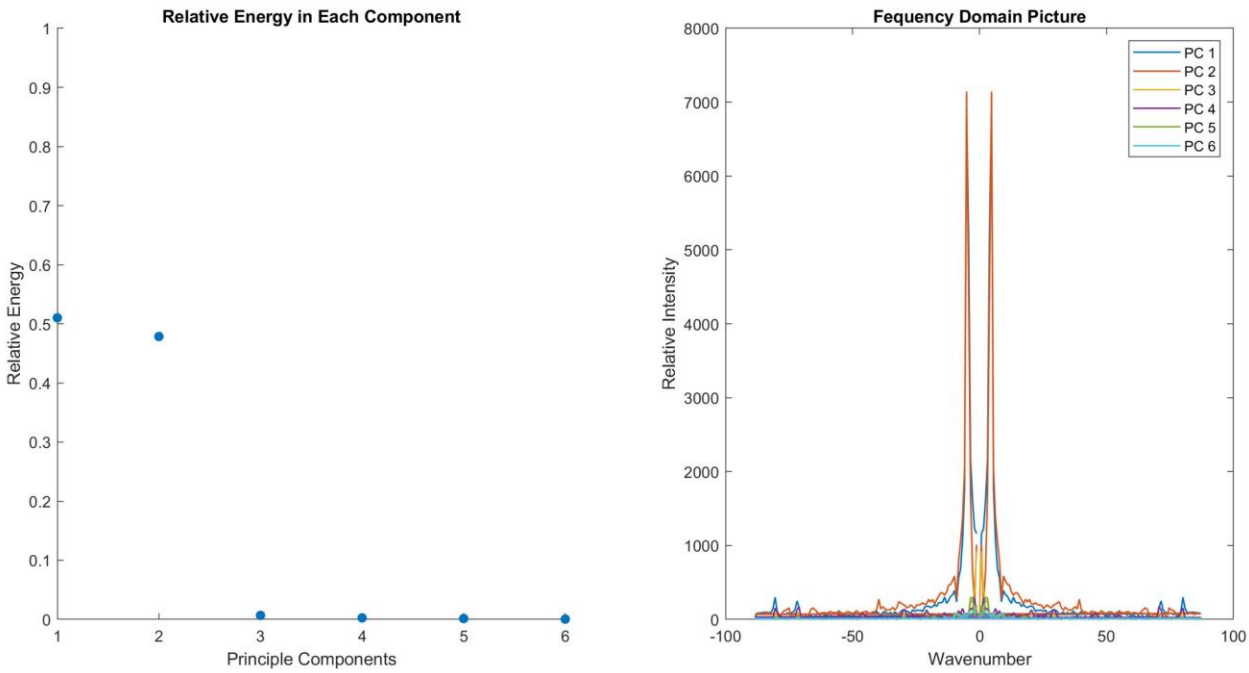

Figure 3a, 3b. Relative energy per PC, and frequency domain break down of the signals

Figure 3 shows that while the first two principle components indeed constitute roughly half of the overall energy of the data, the frequency domain picture shows that the two principle components are really capturing variances that are oscillating at the same frequency. In other words, principle component 1 and 2 are looking at a signal that is, for all intents and purposes, the same, since its frequency is the same. Figure 3b indeed shows that one frequency signature is derived from the data in the principle component space, and in a way goes a step further than the PCA analysis to show that there is indeed only one functional frequency signal derived from the raw data.

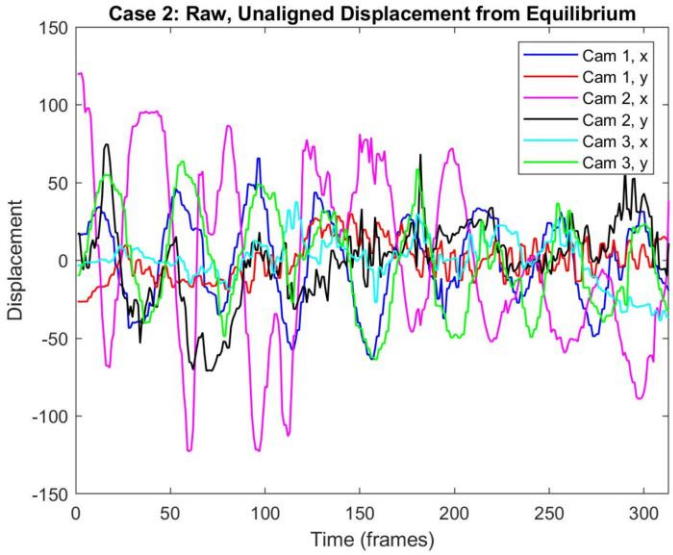

Figure 4. Case 2, Raw camera tracking data.

The data in case 2 is noisy because of camera shake in the video. However, sinusoidal behavior is clearly evident in a number of the camera axes.

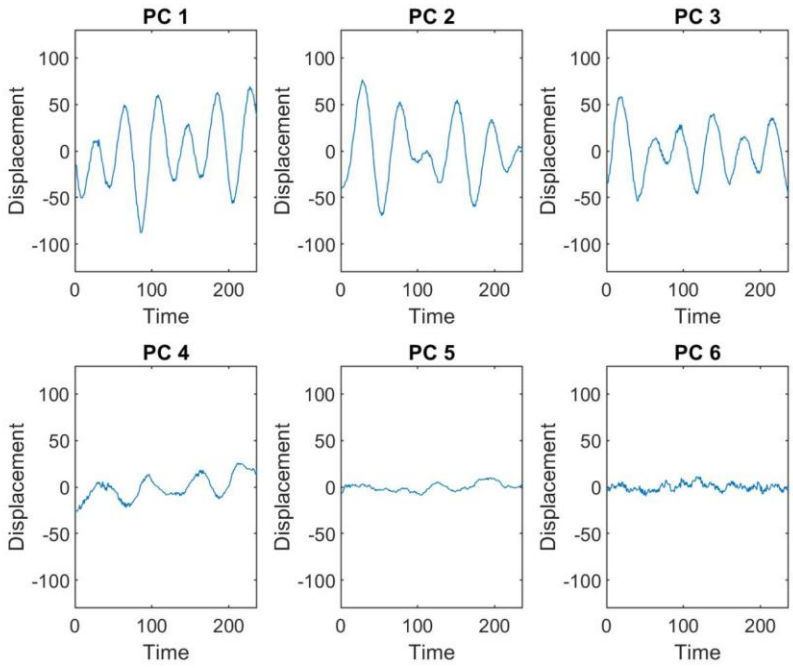

Figure 5. Principle component projections

Case two reveals that a number of out of phase sinusoids describe the data, to the point that even the fourth principle component detects variance along a sinusoidal trajectory to some small extend. Here, the value of the fourier transform in clearing the picture of the data becomes evident.

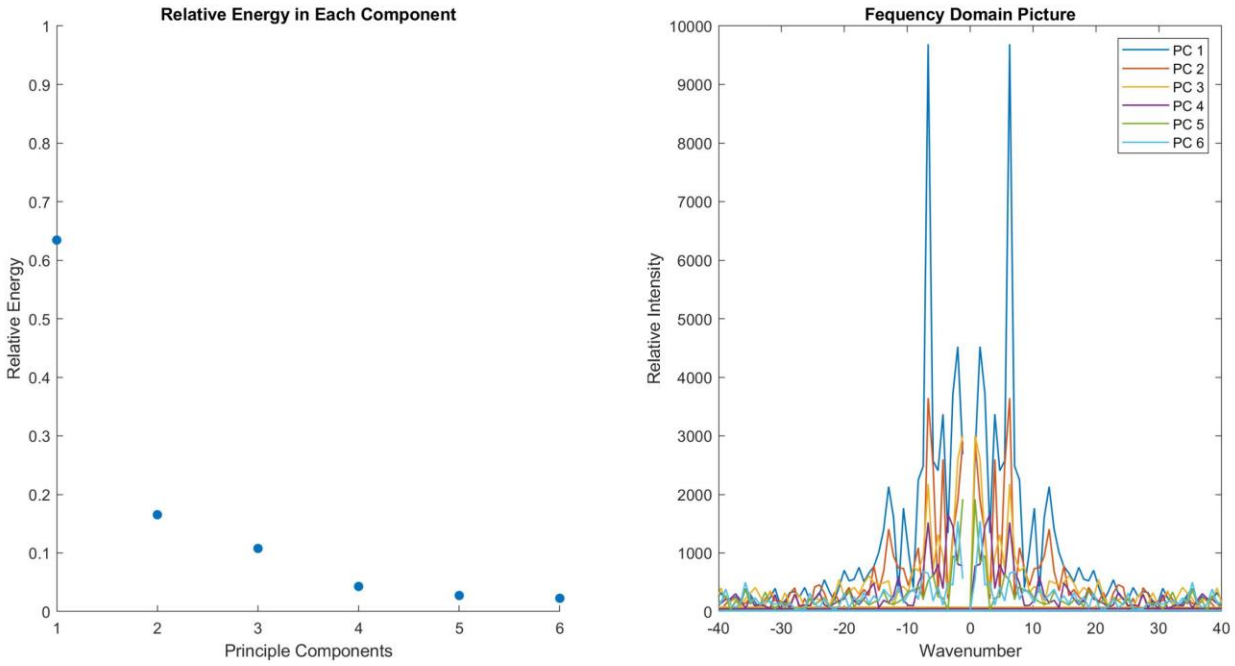

Figure 6a, 6b. Relative energy per PC, and frequency domain break down of the signals

Here, a clear dominance of the first principle component is evident in figure 3a. The frequency domain picture described by Figure 3b corroborates those findings, as there is a clearly dominant frequency (k = 8) amongst the highly noisy data. The PCA analysis, in conjunction

with the analysis facilitated by the Fourier transform, allows us to determine that there is, in all likelihood, one degree of freedom associated with the data.

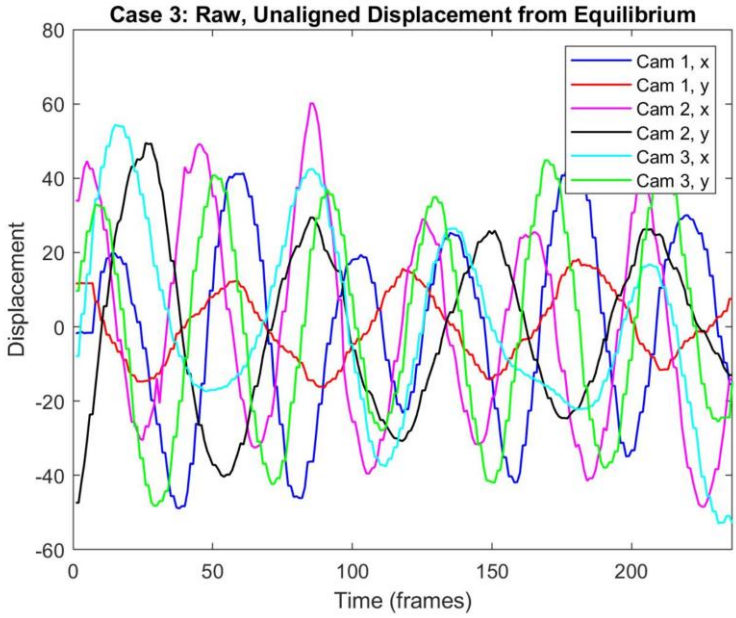

Figure 7. Case 3, raw camera data.

Case 3 reveals a number of out of phase sinusoids with clearly different frequencies. The fact that the waves oscillate at different frequencies is a clear indication that variation along different degrees of freedom may be at play.

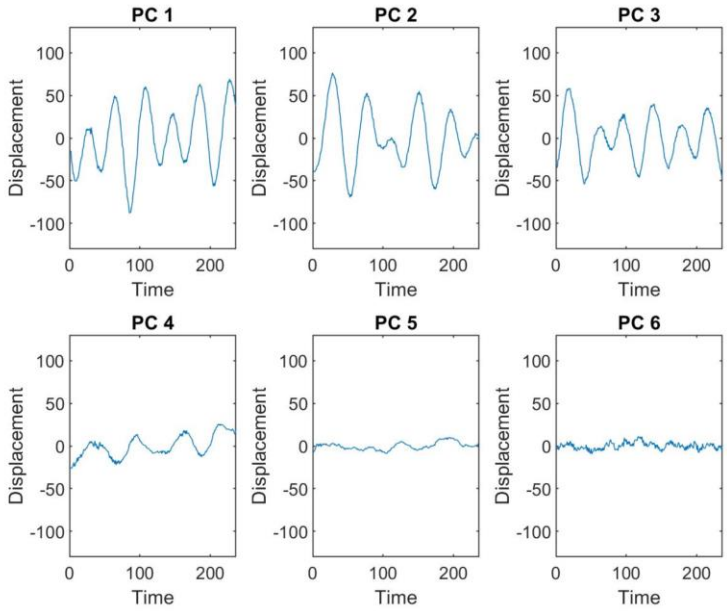

Figure 8. Case 3 principle component projections.

Each of the second, third, and fourth principle components suggest that multiple sinusoids are derived from the raw data, and are indicative of the fact that the system might be two dimensional.

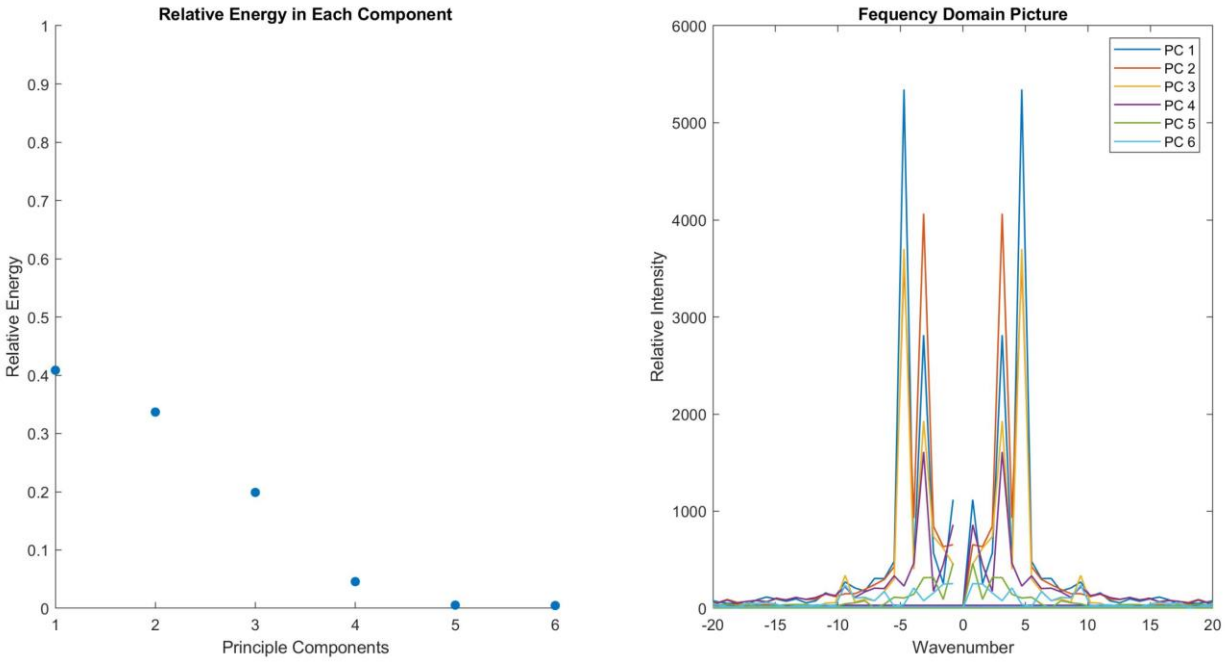

Figure 9a, 9b. Relative Energy and Frequency picture

The mapping of the relative energies clearly shows two dominant components, while the Fourier transform indeed shows that multiple of the principle component axes demonstrate two signals are in the data, where k = 3 and 6 respectively. With the confirmatory data provided by the Fourier transform, it appears that there are two degrees of freedom in the system. Figures for the fourth case show more of the same and are displayed in Appendix C for sake of brevity. The major take away from the fourth case is that only the first principle component truly dominates the data. This means that the oscillatory motion in one of the degrees of freedom was much stronger than in the other two. This was confirmed by watching the video, where the up and down motion of the can was far more detectible by eye than was the sway in the x-z and y-z planes.

V. Summary and Conclusions

PCA provides a dimensionality reduction framework to understand raw data in terms of coordinates that best capture its essential components—its principle components. Here, one, two, and three degree of freedom systems were analyzed to put the PCA algorithm to the test, to see if it could indeed detect the true dimensionality of the data on which it was applied. Many of the cases proved to show that the PCA was effective, especially in the noise Case 2, where it was able to accurately ascertain the single dimensional signal from the noisy six-dimensional forest of sinusoids. This analysis also demonstrated some weakness of the PCA: specifically, out of phase, oscillatory data produces data that is best described along different orthogonal principle components. Fourier transformed were employed to clarify the muddle picture in these particular cases. Overall, like with all experimental procedures, two heads—two strong data analysis methods—is better than one when extracting meaning from data.

Appendix A: MatLab Functions

imajust() – increases contrast of images with a tunable tolerance thresholds along the grayscale.

fft() – allowed fast fourier transform to be applied to oscillatory data

max() – allowed for the maximum pixel data to be extracted for paint can tracking

rbg2gray() – allowed color image matrices to be expressed in grayscale

Appendix B: MatLab Code

```
% HW 3 Avery Pong

% Load in data
cam1_1 = load('cam1_1');
cam2_1 = load('cam2_1');
cam3_1 = load('cam3_1');

% Extract video from struct
video_data1 = [cam1_1.vidFrames1_1];
video_data2 = [cam2_1.vidFrames2_1];
video_data3 = [cam3_1.vidFrames3_1];

% Determine number of frames from last struct component
numFrames1 = size(video_data1,4);
numFrames2 = size(video_data2,4);
numFrames3 = size(video_data3,4);

all_frames = [numFrames1 numFrames2 numFrames3];
min_frames = min(all_frames);               % find frame number of shortest video for plotting
t = (1:min_frames);                         % make time axis for plotting

% Allow collection of x, y data for all 3 cameras
hold_a = []; hold_b = [];                   % Camera 1 collection
hold_c = []; hold_d = [];                   % Camera 2 collection
hold_e = []; hold_f = [];                   % Camera 3 collection

% Set up and clean up
tol = [.02 .98];                            % Set contrast for imadjust
dimen = 6;                                  % Number of cameras


% CAMERA ONE READ
for i = 1:numFrames1

    % Load in Data
    video_play = video_data1(:,:,:,i);      % Read in data per frame
    video_play = rgb2gray(video_play);      % Make grayscale
    video_play = imadjust(video_play, stretchlim(video_play), tol);    % increases contrast
    video_play = video_play(200:430, 300:380); % Make window to exclude non-paint can


    % Find max (white points)
    m = max(video_play(:));                 % Find max the matrix in both component directions
    [x, y] = find(video_play == m);         % find index at which the maxima occur
    hold_a = [hold_a mean(x)];              % store the mean x position of the max white pos of
can
    hold_b = [hold_b mean(y)];              % store the mean y position of the max white pos of
can

%     % Plot video
%     pcolor(video_play);                   % meshgrid display data
%     set(gca,'YDir','reverse');            % invert, right side up
%     colormap(hot);                        % map it HOT
%     drawnow;                              % draw to play as movie
end
```

```
% CAMERA 2 READ
for j = 1:numFrames2
    video_play = video_data2(:,:,:,j);
    video_play = rgb2gray(video_play);
    video_play = imadjust(video_play, stretchlim(video_play), tol);
    video_play = video_play(100:380, 250:330);

    m = max(video_play(:));
    [x, y] = find(video_play == m);
    hold_c = [hold_c mean(x)];
    hold_d = [hold_d mean(y)];

%     pcolor(video_play);
%     set(gca,'YDir','reverse');
%     colormap(hot);
%     drawnow;
end


% CAMERA 3 READ
for k = 1:numFrames3
    video_play = video_data3(:,:,:,k);
    video_play = rgb2gray(video_play);
    video_play = imadjust(video_play, stretchlim(video_play), tol);
    video_play = video_play(220:320, 260:480);

    m = max(video_play(:));
    [x, y] = find(video_play == m);
    hold_e = [hold_e mean(x)];
    hold_f = [hold_f mean(y)];

%     pcolor(video_play);
%     set(gca,'YDir','reverse');
%     colormap(hot);
%     drawnow;

End


% Collect data into data matrix 'MAT'
mat =  [hold_a(1:min_frames);
        hold_b(1:min_frames);
        hold_c(1:min_frames);
        hold_d(1:min_frames);
        hold_e(1:min_frames);
        hold_f(1:min_frames)];


% Clean data and have oscillate about y = 0
[m,n]=size(mat);                        % compute data size
avg = mean(mat,2);                      % compute mean for each row
mat = mat - repmat(avg,1,n);            % subtract mean


% Diagonalize, and run SVD
c_x = (1/(dimen-1))*(mat*mat');         % covariance matrix of raw data
[u,s,v]=svd(mat/sqrt(dimen-1), 'econ'); % SVD of data to find U
Y = u'*mat;                             % Find data in principle component space
c_y = (1/(dimen-1))*Y*Y';               % covariance matrix of transformed data


%%%% Plot principle components %%%%
figure(1)
for i = 1:size(Y,1)
    subplot(2,3,i)
    plot(1:min_frames, Y(i,:))          % Plotted data in coord of principle component
    title(['PC ',num2str(i)])           % Labeled each pc
    ylim([-150 150])
    xlabel('Time')
```

```
    ylabel('Displacement')
end


%%%% Plot Raw Data Each Camera %%%%
figure(2)
plot(t,mat(1,:), 'b', t, mat(2,:), 'r', 'Linewidth', [1]);  % camera 1
title('Case 1: Raw, Unaligned Displacement from Equilibrium')
xlabel('Time (frames)')
ylabel('Displacement')
hold on;
plot(t,mat(3,:), 'm', t, mat(4,:), 'k', 'Linewidth', [1]);  % camera 2
hold on;
plot(t,mat(5,:), 'c', t, mat(6,:), 'g', 'Linewidth', [1]);  % camera 3
xlim([0 min_frames])
legend('Cam 1, x', 'Cam 1, y', 'Cam 2, x', 'Cam 2, y', 'Cam 3, x', 'Cam 3, y');


%%%% Plot distribution of energies across the principle components %%%%
d = diag(c_y);                                  % take diag components of c_y
energy = zeros(6,1);
for i = 1:dimen
    energy(i) = d(i)/sum(d);                    % find relative energy in each component
end

figure(3)
subplot(1, 2, 1)
s_plot_x = (1:dimen);
scatter(s_plot_x,energy, 'filled')              % plot relative energy
title('Relative Energy in Each Component');
xticks([1 2 3 4 5 6])
xlabel('Principle Components')
ylabel('Relative Energy')
ylim([0 1])


% Fourier Analysis
n = length(Y(1,:));                             % fourier modes, just each data pt
L = 8;                                          % across about 8 second video
t2 = linspace(0, L, n+1);                       % n+1 because first point same as last
t = t2(1:n);
k = (2*pi/L)*[0:(n/2-1) -n/2:-1];               % freq comp for fft.

subplot(1, 2, 2)
for i = 1:6
    u = Y(i,:);
    ut = fft(u(1:224));                         % take fft of each vector of pc data
    text = ['PC ', num2str(i)];
    title('Fequency Domain Picture');
    xlabel('Wavenumber');
    ylabel('Relative Intensity');
    plot(k, abs(ut) , '-', 'DisplayName', text, 'linewidth', [1]);
    hold on
end

hold off
legend show
```

Appendix C: Figures for Case 4, described above

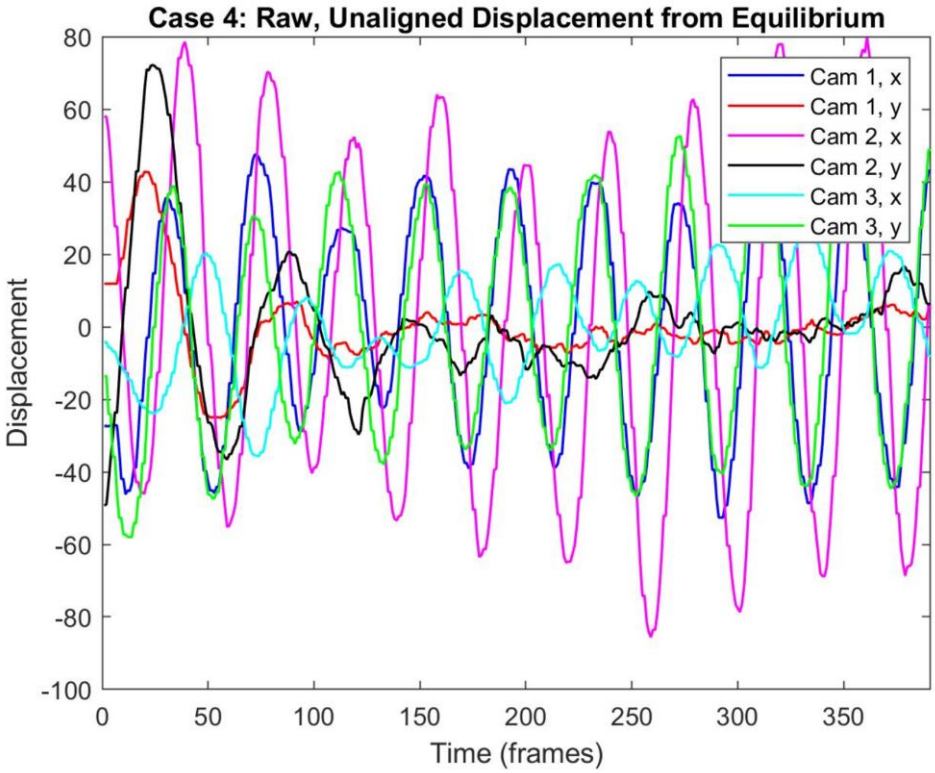

Figure 10. Raw Case 4 data.

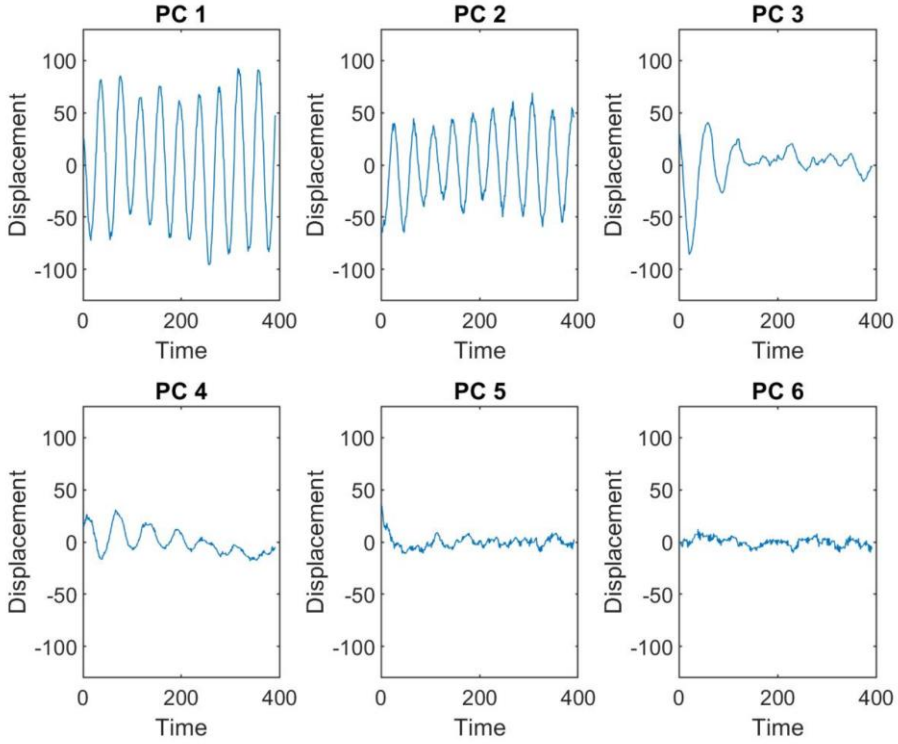

Figure 11. PC projections of Case 4 data.

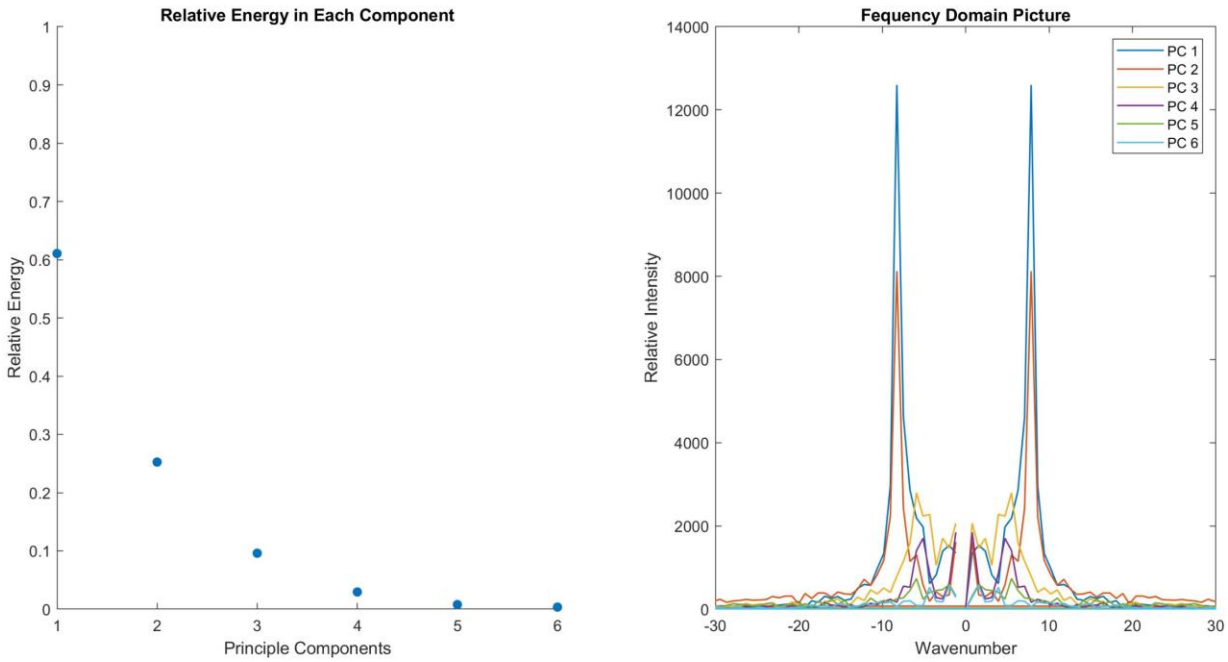

Figure 12. Relative PC energy and Fourier Domain picture.

Again, the case 4 data reveals one dominant principle component. The frequency components in Figure 4b reveal that there is a lot of lower frequency oscillatory activity with clear spikes observed in PC3, indicating there may be more than the single degree of freedom that it would otherwise suggest.