HW3 Principal Component Analysis

Mingzhong Deng

Feburary 22, 2019

Abstract

This project uses principal component analysis to analyze videos of a hanging mass taken by cameras from three directions. The signal of the hanging mass is tracked by the flashlight located at the top of the mass. In some of the tests, the cameras are shaking or displacement and roration are introduced, so the signals will include noise. Pricinpal component analysis is used based on the idea of singular value decomposition (SVD).

1 Introduction and Background

All matrices can be describe as stretching and rotation of a particular vector. A singular value decomposition (SVD) is a factorization of matrix into a number of constitutive components. SVD is an important tool in many applications, partially due to the ability of matrix dimensional reductions. Principle component analysis (PCA) is an application of SVD. Through the process of PCA, moisy and redundant data can be extract out to ideal and simplified data.

There are four tests in the project. As mentioned above, three cameras are used to capture the movement of the mass. In the first test, the mass is normally moving so the mass performs simple harmonic motion in the z direction. In the second test, the mass is normally moving so the mass performs simple harmonic motion in the z direction but camera shakes are introduced. In the third test, the mass is released off-center so as to produce motion in the xy plane as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations. In the fourth test, the mass is released off-center and rotates so as to produce motion in the xy

plane, rotation as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations.

2 Theoretical Background

SVD splits a matrix into the product of three matrices. $A = U\Sigma V$, where U is a unitary matrix, Σ is a retangular diagonal matrix and V is also a unitary matrix. The matrices U and V are rotational matrices and matrix Σ is a stretching matrix. Performing SVD to a matrix remove redundancy and indentify the signals with maximal variance. SVD is tool of choice for data analysis and dimensional reduction. In fact, the SVD diagonalizes and each singular directions captures as much energy as possible as measured by the singular values σ_j.

PCA produces the eigenvalue decomposition and the projection of the original data onto principle component basis using the diagonalization of the matrix by performing SVD. PCA is a precise method for performing low-dimensional reductions of a given system. Even though PCA is powerful, it may not produce the optimal results.

3 Algorithm Implementation and Development

The algorithm for extracting original data and data after PCA is the same for the four tests of the project.

- Load camera mat files.
- Each video file is saved to a 4D tensor.
- Convert RGB data file to grayscale.
- Isolate the portions of the video to find where the movement occurs by setting all pixels to zero except a small portion.
- Search for the maximum of the data and return the indices of the maximum point to extract the motion of the can.
- Construct the X and Y vectors that contains the indices, truncate data such that all vectors are length 200.

- Create a smooth curve by applying a Shannon window to the original data.
- Form a new matrix with all X and Y vectors and perform SVD to get U, Σ, and V matrices.
- Plot all data, before and after PCA and the energy of each mode.

4 Computational Results

4.1 First Test

Figure 1 shows the raw data (blue line) from three cameras and the smooth data (red line) with Shannon window before PCA. From the figure, we can see that the mass oscillates along the Y direction in camera 1 and 2 but only oscillates in the X direction in camera 3.

Figure 2 shows the data (blue line) from three cameras and the smooth data (red line) with Shannon window after PCA and recreated with the largest two modes. Very similar to figure 1, the mass oscillates along the Y direction in camera 1 and 2 but only oscillates in the X direction in camera 3. Figure 2 neglects some spikes in the curves in figure 1 but preserves mostly the same shape and amplitude. Therefore, saving only the first two modes, the largest two, will be enough to describe the data.

Figure 3 shows the modes and their corresponding energy percentage. In test 1, the first two modes combined contributed to about 95% to the total original data.

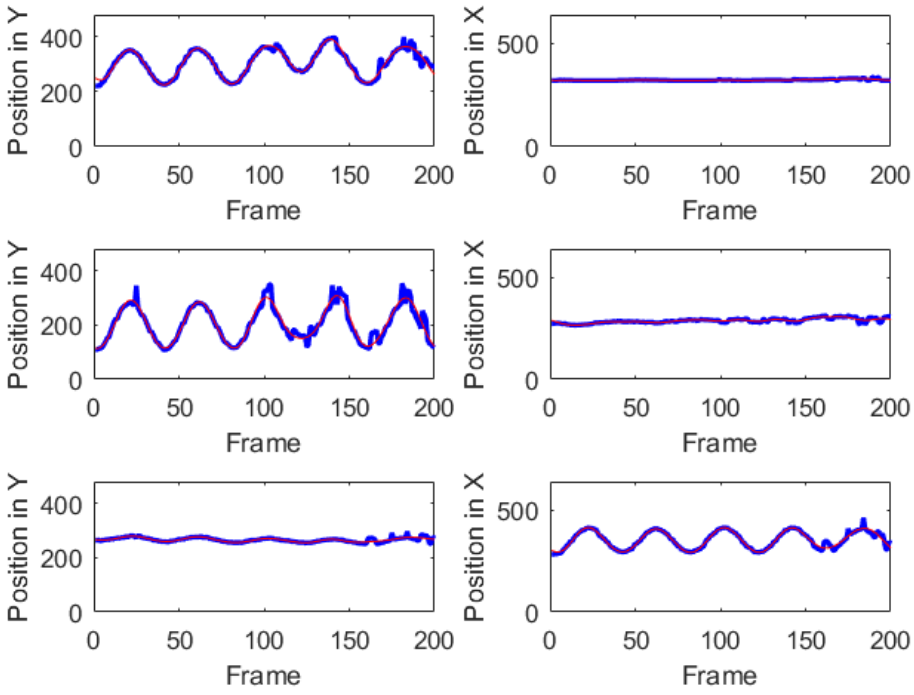

Figure 1: X and Y postion per frame for test 1. Blue line is raw data and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

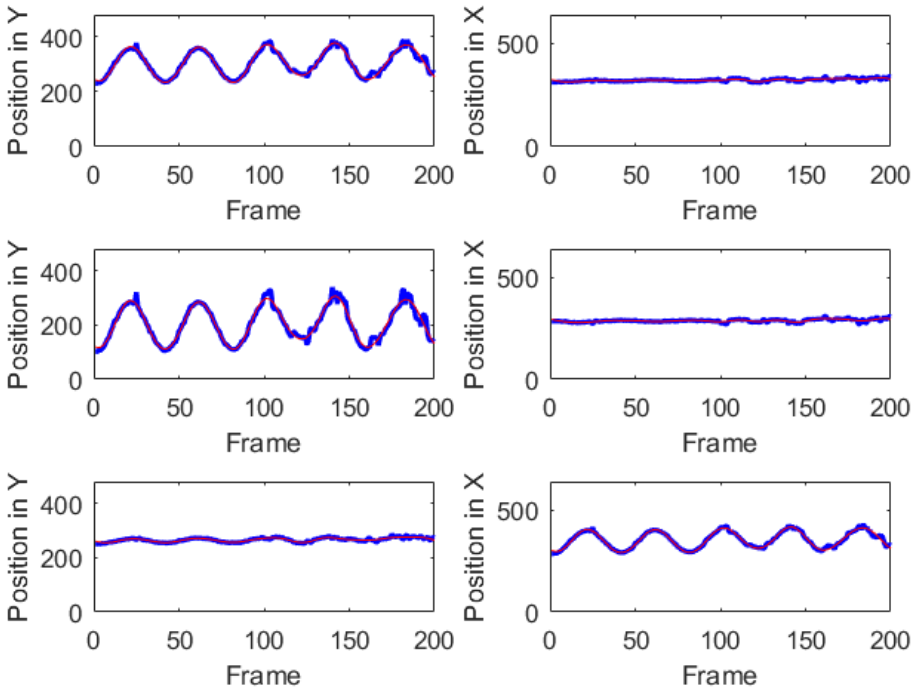

Figure 2: X and Y postion per frame for test 1. Blue line is data after PCA and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

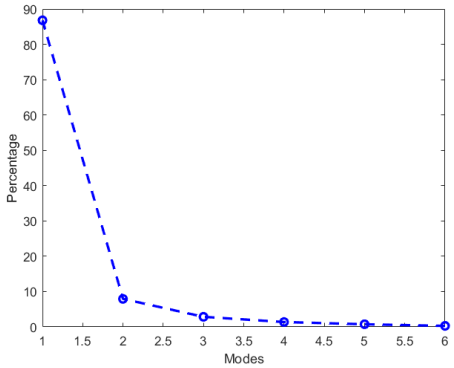

Figure 3: Percentage of energy of each mode in test 1.

4.2 Second Test

Figure 4 shows the raw data (blue line) from three cameras and the smooth data (red line) with Shannon window before PCA. From the figure, we can see that the mass oscillates along the Y direction in camera 1 and 2 but only oscillates in the X direction in camera 3. With the situation of camera shake, movement of the mass in the X direction in camera 1 and 2 is not as steady as the previous test, but they could be treat as flat lines.

Figure 5 shows the data (blue line) from three cameras and the smooth data (red line) with Shannon window after PCA and recreated with the largest two modes. Very similar to figure 4, the mass oscillates along the Y direction in camera 1 and 2 but only oscillates in the X direction in camera 3. Figure 5 neglects some spikes in the curves in figure 4 but preserves mostly the same shape and amplitude. Therefore, saving only the first two modes, the largest two, will be enough to describe the data. After PCA, the effect of camera shake can be preserved. The spikes and shaking in figure 4 is reserved in figure 5.

Figure 6 shows the modes and their corresponding energy percentage. In test 2, the first two modes combined contributed to about 92% to the total original data.

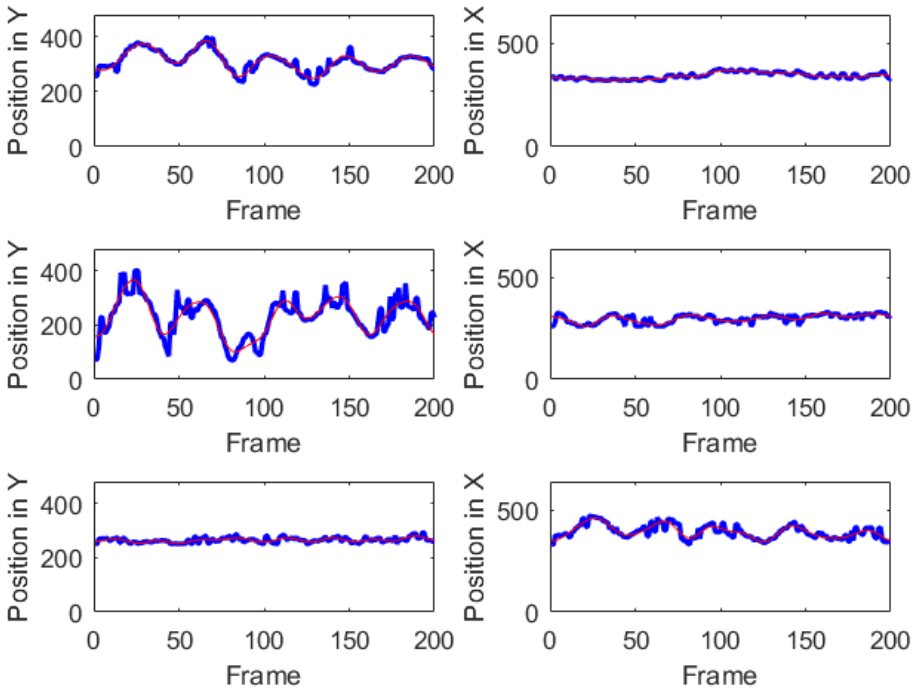

Figure 4: X and Y postion per frame for test 2. Blue line is raw data and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

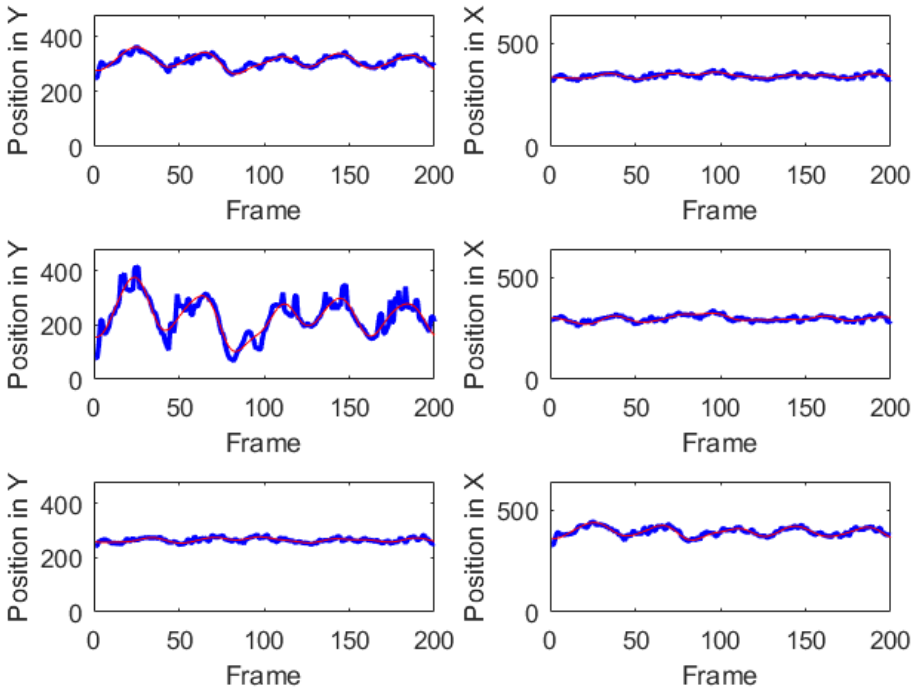

Figure 5: X and Y postion per frame for test 2. Blue line is data after PCA and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

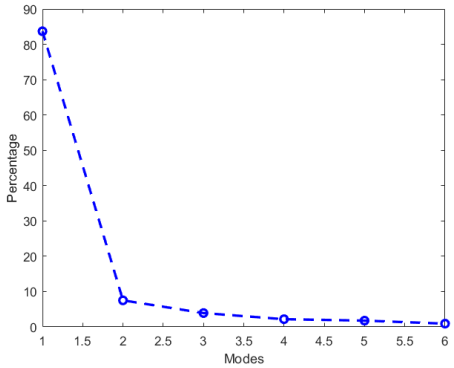

Figure 6: Percentage of energy of each mode in test 2.

4.3 Third Test

Figure 7 shows the raw data (blue line) from three cameras and the smooth data (red line) with Shannon window before PCA. Because the mass is released off-centerd, it is not in simple harmonic motion anymore. From the figure, we can see that the mass oscillates along the Y direction in camera 1 and 2, has some movement in camera 3. The mass oscillates along the X direction in camera 2 and 3, has some movement in camera 1.

Figure 8 shows the data (blue line) from three cameras and the smooth data (red line) with Shannon window after PCA and recreated with the largest two modes. The figure shows similar curves comparing to figure 7, but the amplitude is not significant as figure 7.

Figure 9 shows the modes and their corresponding energy percentage. In test 3, the first two modes combined contributed to about 85% to the total original data. Higher quality data can be captured if using the first three or more modes.

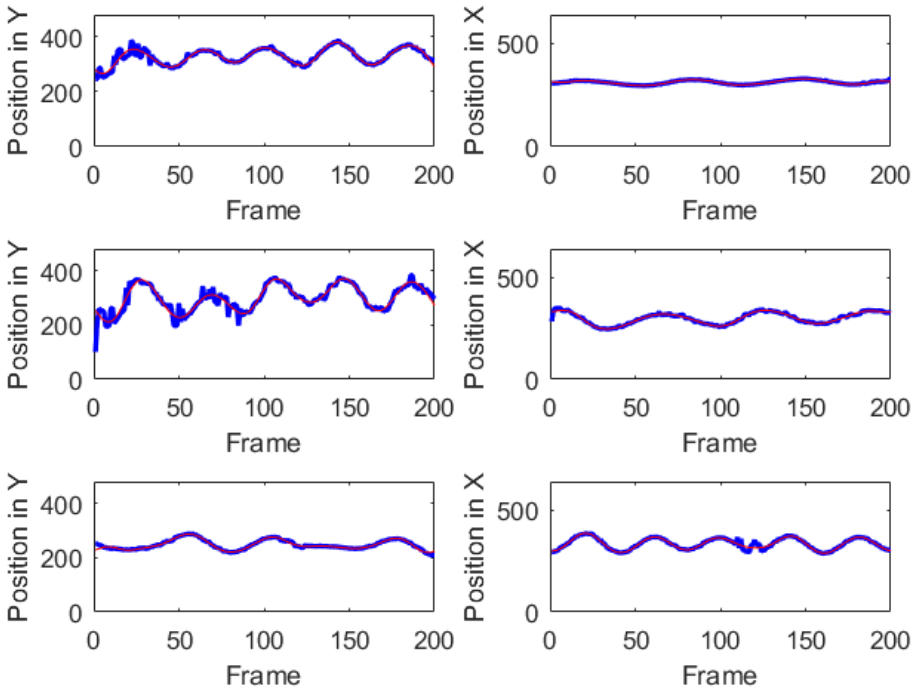

Figure 7: X and Y postion per frame for test 3. Blue line is raw data and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

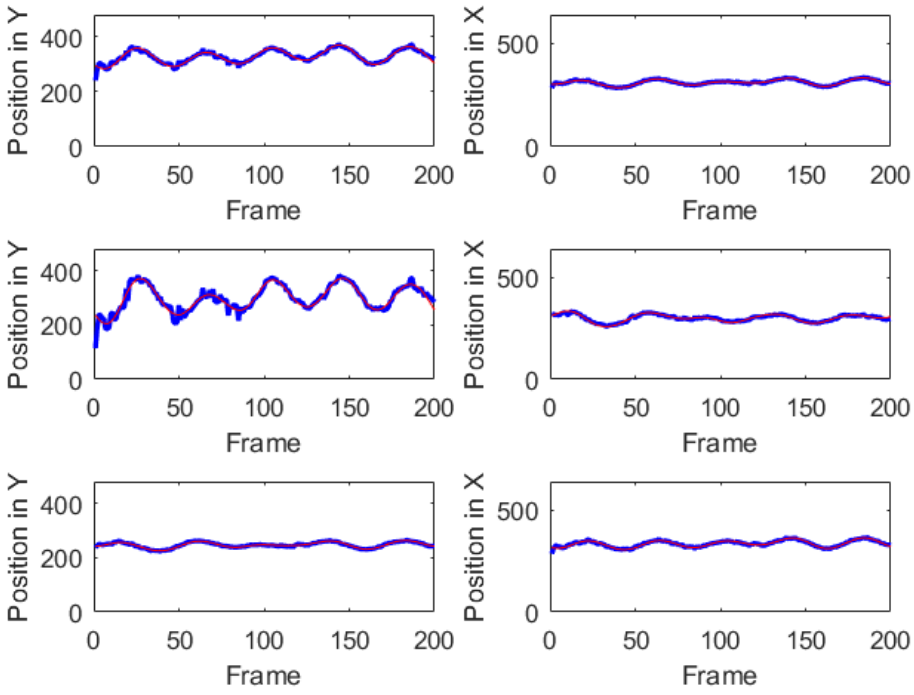

Figure 8: X and Y postion per frame for test 3. Blue line is data after PCA and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

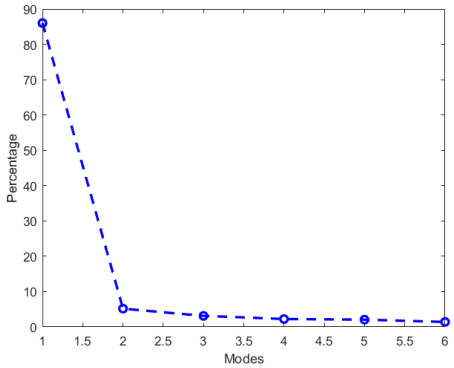

Figure 9: Percentage of energy of each mode in test 3.

4.4 Fourth Test

Figure 10 shows the raw data (blue line) from three cameras and the smooth data (red line) with Shannon window before PCA. From the figure, we can see that the mass oscillates along the Y direction in camera 1 and 2 but only oscillates in the X direction in camera 3. The movement of the mass in the X direction in all three cameras are mostly steady.

Figure 11 shows the data (blue line) from three cameras and the smooth data (red line) with Shannon window after PCA and recreated with the largest two modes. Very similar to figure 10, the mass oscillates along the Y direction in camera 1 and 2. Figure 11 can still capture the spikes in figure 10 but not as significant. Therefore, saving only the first two modes, the largest two, will be enough to describe the data. After PCA, the effect of camera shake and releasing off-centered can be preserved. The spikes in figure 10 is reserved in figure 11.

Figure 12 shows the modes and their corresponding energy percentage. In test 4, the first two modes combined contributed to about 92% to the total original data.

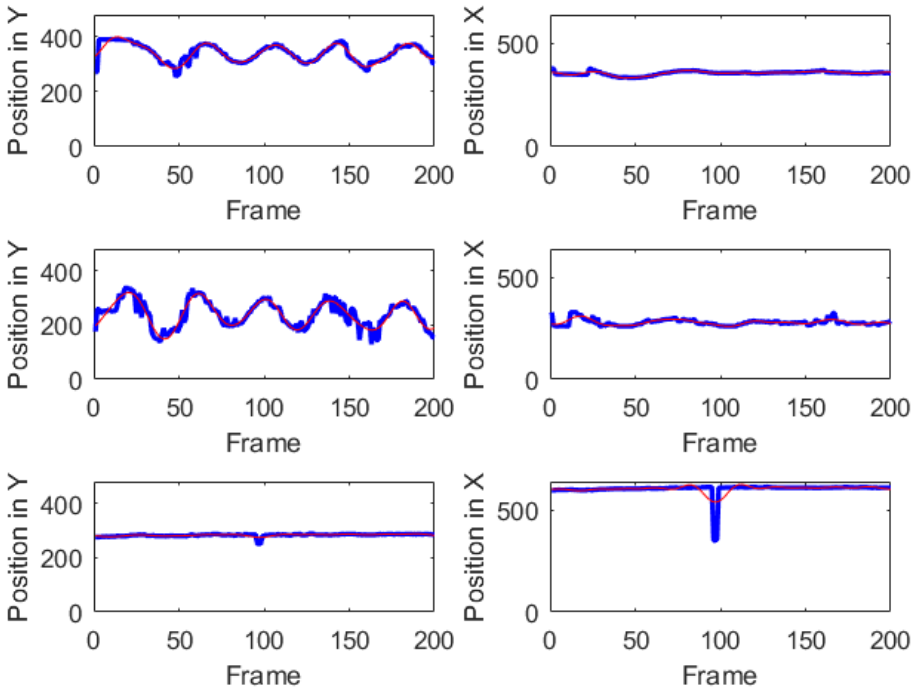

Figure 10: X and Y postion per frame for test 4. Blue line is raw data and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

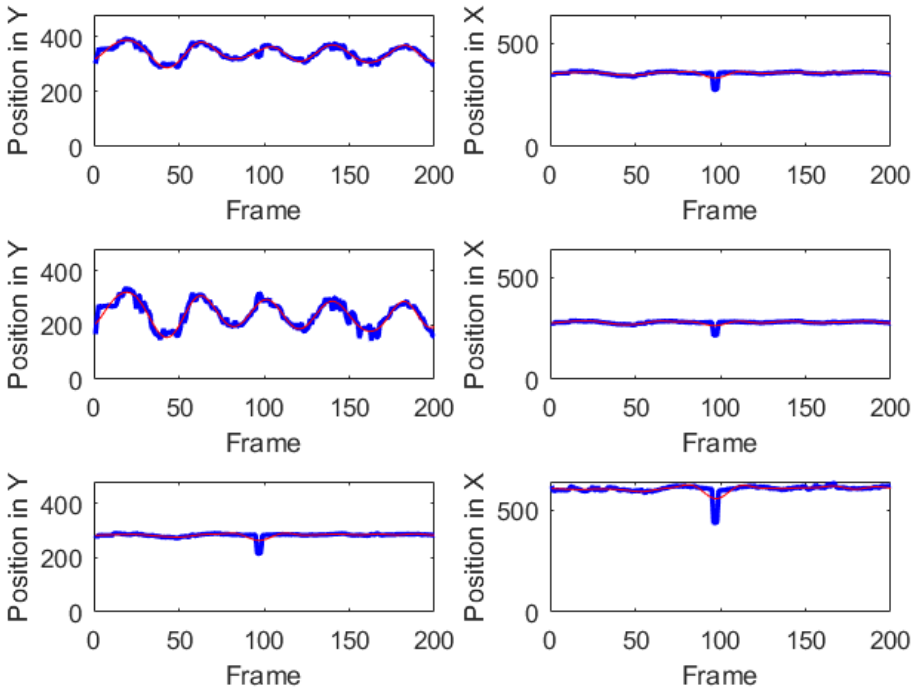

Figure 11: X and Y postion per frame for test 4. Blue line is data after PCA and red line is smooth data. Camera 1 (Top), Camera 2 (Mid), Camera 3 (Bottom).

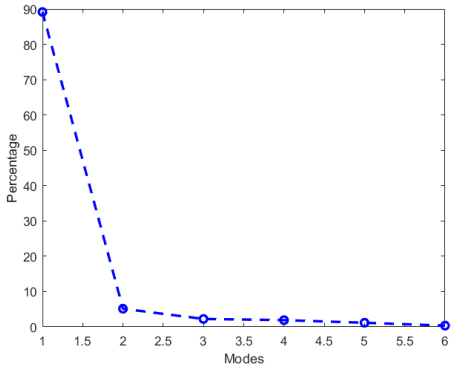

Figure 12: Percentage of energy of each mode in test 4.

5 Summary and Conclusions

In the project, four different kinds of test are analyzed by applying them to PCA. The result of the original data and the data after PCA are mostly in aggrement except minor differences. There are 6 singluar values produced in each test case but using the top 2 to reconstruct the data will produce a close image to the original data. It is a trade off to use original data or decomposed data. Original data gives the most accurate result but takes too much resources for computing and storage. In contrast, using decomposed data after PCA algorithm, dimension will be reduced, therefore, it saves resources for computing and storage but it only gives less accurate data.

6 References

Data-Driven Modeling & Scientific Computation

Appendix A - MATLAB functions

- `rgb2gray`: converts the truecolor image RGB to the grayscale intensity image
- `svd`: [U,S,V] = svd(A) performs a singular value decomposition of matrix A, such that $A = U * S * V'$.
- `diag`: returns a square diagonal matrix with the elements of vector v on the main diagonal.
- `ind2sub`: determines the equivalent subscript values corresponding to a single index into an array.
- `frame2im`: returns the indexed image data X and associated colormap Map from the single movie frame F.

Appendix B - MATLAB code

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

```
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');

%%
frame1 = vidFrames1_1;
frame2 = vidFrames2_1;
frame3 = vidFrames3_1;

f1 = size(frame1, 4);
f2 = size(frame2, 4);
f3 = size(frame3, 4);

for i = 1:f1
    mov1(i).cdata = frame1(:,:,:,i);
    mov1(i).colormap = [];
end
for i = 1:f2
    mov2(i).cdata = frame2(:,:,:,i);
    mov2(i).colormap = [];
end
for i = 1:f3
     mov3(i).cdata = frame3(:,:,:,i);
     mov3(i).colormap = [];
end


%%

X1 = []; Y1 = [];
X2 = []; Y2 = [];
X3 = []; Y3 = [];


for i = 1:f1
    a = rgb2gray(frame2im(mov1(i)));
    a(:,1:320) = 0;
    a(:,380:end) = 0;
```

```
    a(1:200,:) = 0;
    [M, I] = max(a(:));
    [x1, y1] = ind2sub(size(a), I);
    X1 = [X1 x1]; Y1 = [Y1 y1];
end

for i = 1:f2
    a = rgb2gray(frame2im(mov2(i)));
    a(:,1:260) = 0;
    a(:,330:end) = 0;
    [M, I] = max(a(:));
    [x2, y2] = ind2sub(size(a), I);
    X2 = [X2 x2]; Y2 = [Y2 y2];
end

for i = 1:f3
    a = rgb2gray(frame2im(mov3(i)));
    a(1:250,:) = 0;
    a(310:end,:) = 0;
    a(:, 1:260) = 0;
    [M, I] = max(a(:));
    [x3, y3] = ind2sub(size(a), I);
    X3 = [X3 x3]; Y3 = [Y3 y3];
end

%%


[min, I] = min(X1(1:50));
X1=X1(I:I+200); Y1=Y1(I:I+200);
[min, I] = min(X2(1:50));
X2=X2(I:I+200); Y2=Y2(I:I+200);
[min, I] = min(Y3(1:50));
Y3=Y3(I:I+200); X3=X3(I:I+200);


% shannon window
X1fft = fft(X1); X2fft = fft(X2); X3fft = fft(X3);
```

```
Y1fft = fft(Y1); Y2fft = fft(Y2); Y3fft = fft(Y3);
X1fft(10:end-10) = 0; X2fft(10:end-10) = 0; X3fft(10:end-10) = 0;
Y1fft(10:end-10) = 0; Y2fft(10:end-10) = 0; Y3fft(10:end-10) = 0;
X1s = abs(ifft(X1fft)); Y1s = abs(ifft(Y1fft));
X2s = abs(ifft(X2fft)); Y2s = abs(ifft(Y2fft));
X3s = abs(ifft(X3fft)); Y3s = abs(ifft(Y3fft));

figure(1)
subplot(3,2,1), ...
plot(X1, 'b', 'LineWidth', 2); hold on;
plot(X1s, 'r', 'LineWidth', 0.5)
axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
subplot(3,2,2), ...
plot(Y1, 'b', 'LineWidth', 2); hold on;
plot(Y1s, 'r', 'LineWidth', 0.5)
axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');
subplot(3,2,3), ...
plot(X2, 'b', 'LineWidth', 2); hold on;
plot(X2s, 'r', 'LineWidth', 0.5)
axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
subplot(3,2,4), ...
plot(Y2, 'b', 'LineWidth', 2); hold on;
plot(Y2s, 'r', 'LineWidth', 0.5)
axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');
subplot(3,2,5), ...
plot(X3, 'b', 'LineWidth', 2); hold on;
plot(X3s, 'r', 'LineWidth', 0.5)
axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
subplot(3,2,6), ...
plot(Y3, 'b', 'LineWidth', 2); hold on;
plot(Y3s, 'r', 'LineWidth', 0.5)
axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');

%%
Amat = [X1; Y1; X2; Y2; X3; Y3];
Amat_ = [X1s; Y1s; X2s; Y2s; X3s; Y3s];

% SVD
```

```
[u, s, v] = svd(Amat);
Asvd = u(:,1:2)*s(1:2,1:2)*v(:,1:2)';
X1 = Asvd(1,:); Y1 = Asvd(2,:);
X2 = Asvd(3,:); Y2 = Asvd(4,:);
X3 = Asvd(5,:); Y3 = Asvd(6,:);

[u_, s_, v_] = svd(Amat_);
Asvd_ = u_(:,1:2)*s_(1:2,1:2)*v_(:,1:2)';
X1s = Asvd_(1,:); Y1s = Asvd_(2,:);
X2s = Asvd_(3,:); Y2s = Asvd_(4,:);
X3s = Asvd_(5,:); Y3s = Asvd_(6,:);


figure(2)
subplot(3,2,1), ...
plot(X1, 'b', 'LineWidth', 2); hold on;
plot(X1s, 'r', 'LineWidth', 0.5)
axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
subplot(3,2,2), ...
plot(Y1, 'b', 'LineWidth', 2); hold on;
plot(Y1s, 'r', 'LineWidth', 0.5)
axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');
subplot(3,2,3), ...
plot(X2, 'b', 'LineWidth', 2); hold on;
plot(X2s, 'r', 'LineWidth', 0.5)
axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
subplot(3,2,4), ...
plot(Y2, 'b', 'LineWidth', 2); hold on;
plot(Y2s, 'r', 'LineWidth', 0.5)
axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');
subplot(3,2,5), ...
plot(X3, 'b', 'LineWidth', 2); hold on;
plot(X3s, 'r', 'LineWidth', 0.5)
axis([0 200 0 480]); ylabel('Position in Y'); xlabel('Frame');
subplot(3,2,6), ...
plot(Y3, 'b', 'LineWidth', 2); hold on;
plot(Y3s, 'r', 'LineWidth', 0.5)
axis([0 200 0 640]); ylabel('Position in X'); xlabel('Frame');
```

```
%%
figure(3)
plot(diag(s)*100/sum(diag(s)), 'bo--', 'LineWidth', 2);
xlabel('Modes'); ylabel('Percentage');
```