AMATH 482 Homework 3

Tsz Wai Tsui

February 24, 2021

Abstract

We aim to explore the Principal Component Analysis (PCA) method on four datasets. Each one consists of three videos of a spring-mass system, which are taken at different angles. They are also affected by variations of noises. After tracking the positions of the mass, we are able to perform PCA and extract the dynamics from each dataset to tell the effects of noise on the PCA algorithms.

1 Introduction and Overview

In the spring-mass system, the mass is represented by a paint can with a bright spot on top of it, which is used to track the can's position over time. The four datasets represent four cases with different level of noises, which are ideal case (test 1), noisy case (test 2), horizontal displacement (test 3), and horizontal displacement with rotation (test 4). In test 1, there is a small displacement in the z direction and ensuring oscillations. In test 2, test 1 experiment is repeated except for an introduction of camera shakes into the video recording. In test 3, the paint can is released off-center so as to produce motion in the $x-y$ plane as well as the z direction. In test 4, the paint can is released off-center and rotates so as to produce rotation in the $x-y$ plane and motion in the z direction. Camera 1 and 2 recorded the oscillation perpendicular to the floor while camera 3 recorded by rotating 90 degree to the left. In other words, the movement in z direction is shown as moving sideways in camera 3. Since each test was filmed by three cameras at different angles, the data is redundant. PCA helps reduce the redundancies and find out the principal components of our data matrix.

2 Theoretical background

The position of the paint can in z direction can be calculated using a function of time:

$$z(t) = Acos(\omega t + \varphi), \tag{1}$$

where A the amplitude of the oscillation, ω is the angular velocity of the can, and φ is the its initial angular velocity.

PCA is based on the idea of Singular Value Decomposition (SVD), which is a way of factoring matrices:

$$\mathbf{A} = \mathbf{U\Sigma V}^*, \tag{2}$$

where matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are unitary matrices and $\mathbf{\Sigma} \in \mathbb{R}^{m\times n}$ is diagonal. The values σ_n on the diagonal of $\mathbf{\Sigma}$ are called the singular values of the matrix $\mathbf{A}$. The vectors un which make up the columns of $\mathbf{U}$ are called the left singular vectors of $\mathbf{A}$. The vectors v_n which make up the columns of $\mathbf{V}$ are called the right singular vectors of $\mathbf{A}$.

Since we need a matrix to perform PCA, we find out the x (horizontal displacement), y (vertical displacement) coordinates of the can over time, and denote them (x_a, y_a) for camera 1, (x_b, y_b)

for camera 2, and (x_c, y_c) for camera 3. If we make vectors of all the position at each time, we can get the matrix:

$$\mathbf{X} = \begin{bmatrix} x_a \\ y_a \\ x_b \\ y_b \\ x_c \\ y_c \end{bmatrix}. \tag{3}$$

Then, we can compute all the variances, which measures the spread of data, and covariances, which measures how the variables vary with respect to each other, between the rows of **X** with one matrix multiplication:

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^{\mathbf{T}} = \begin{bmatrix} \sigma_a^2 & \sigma_{ab}^2 & \sigma_{ac}^2 & \sigma_{ad}^2 \\ \sigma_{ba}^2 & \sigma_b^2 & \sigma_{bc}^2 & \sigma_{bd}^2 \\ \sigma_{ca}^2 & \sigma_{cb}^2 & \sigma_c^2 & \sigma_{cd}^2 \\ \sigma_{da}^2 & \sigma_{db}^2 & \sigma_{dc}^2 & \sigma_d^2 \end{bmatrix}. \tag{4}$$

Note that to avoid under-predicting the spread of the data, we need to divide by $n-1$, where n is the sample data size. $\mathbf{C_X}$, covariance matrix is a square symmetric matrix, and the larger the covariance, the more redundant the data is. As the goal of principal component analysis is to find a new set of coordinates (a change of basis) so that the variables are now uncorrelated, we can use SVD to diagonalize this matrix so that all off-diagonal elements (covariances) are zero. After that, assuming the full matrix is rank r, we can measure the energy contained in the rank-*N* approximation by:

$$energy_N = \frac{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_N^2}{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_r^2} = \frac{\|X_N\|_F^2}{\|X\|_F^2} \tag{5}$$

3 Algorithm Implementation and development

For all four datasets, we went through the same computational process. First, we load a set of movie files into MATLAB and calculate the number of frames each file has. In a for loop, we convert each frame from RGB into grayscale image using `rgb2gray` and set the RGB values of the parts of image, which do not capture the can's movement to zero. Since there is a bright spot on top of the can and its RGB value should be close to 255, we can track the can's position by finding the x, y coordinates (in pixels) of the maximum value and store them in vectors.
As reflected from the different number of frames, three cameras did not start recording at the same time. Therefore, we need to align the three videos before performing PCA. We look for the minimum value in y coordinates (z direction) and use it as the starting index to make the vectors the same length (201). Then, we stack up the vectors to form the matrix **X** in equation 3 and perform SVD by `svd(A,'econ')`. Note that for performing PCA, we need to subtract the matrix from its row means. At last, we plot the original data, the principal components, the projected data, and the captured energies for each data file.

4 Computational Results

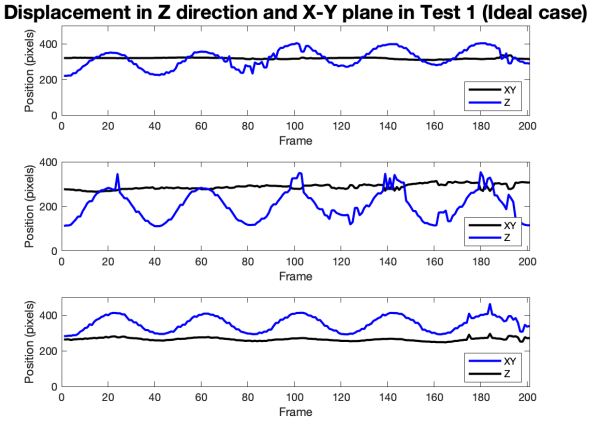

Fig 1: x (X-Y) and y (Z) positions of the can in test 1 for camera 1 (Top), camera 2 (Middle), and camera 3 (Bottom)

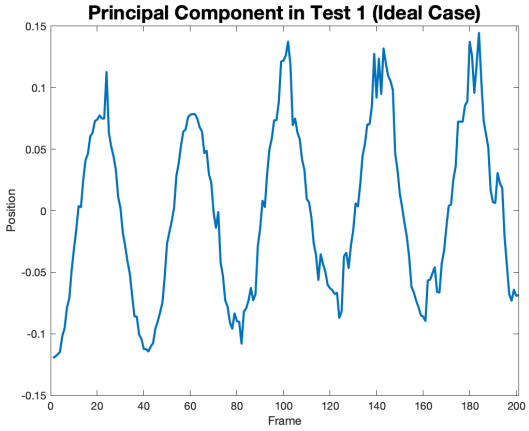

Fig 2: The first principal component in test 1

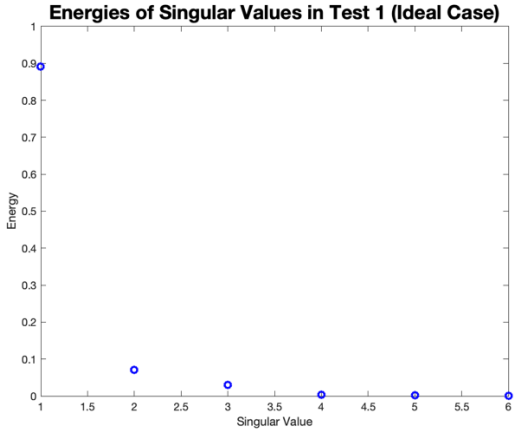

Fig 3: Energies captured at each singular value in test 1

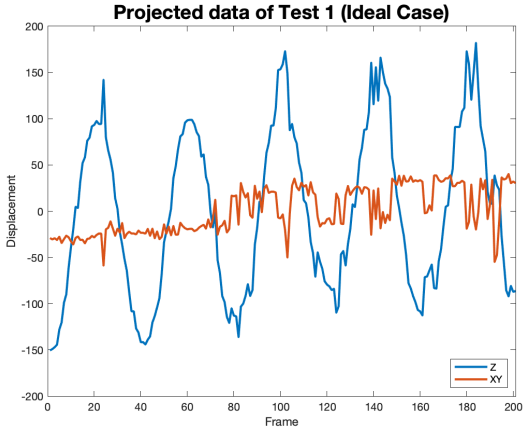

Fig 4: Projected data in the basis of the principal components in test 1

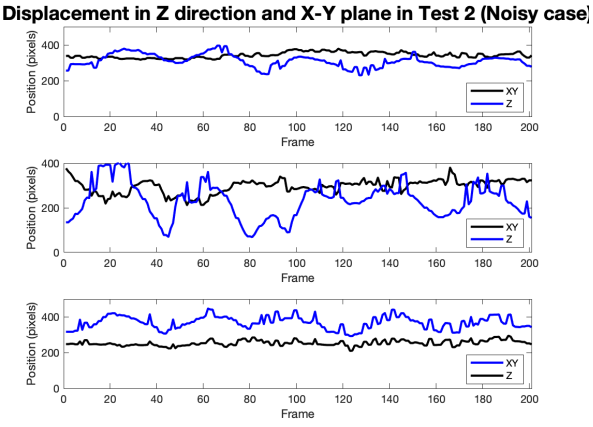

Fig 5: x (X-Y) and y (Z) positions of the can in test 2 for camera 1 (Top), camera 2 (Middle), and camera 3 (Bottom)

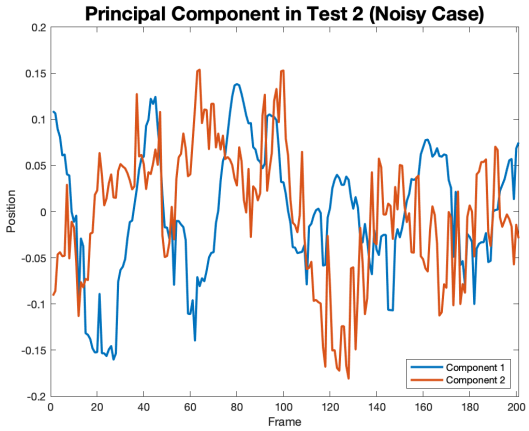

Fig 6: The first two principal components in test 2

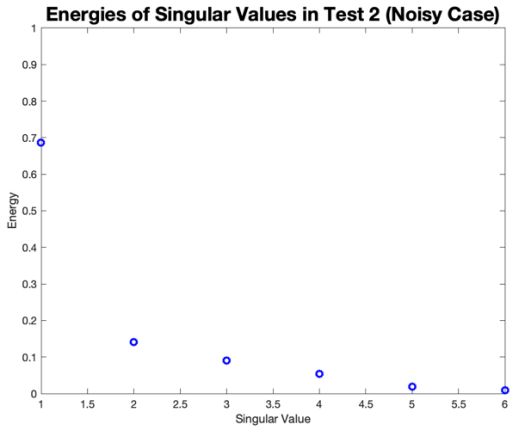

Fig 7: Energies captured at each singular value in test 2

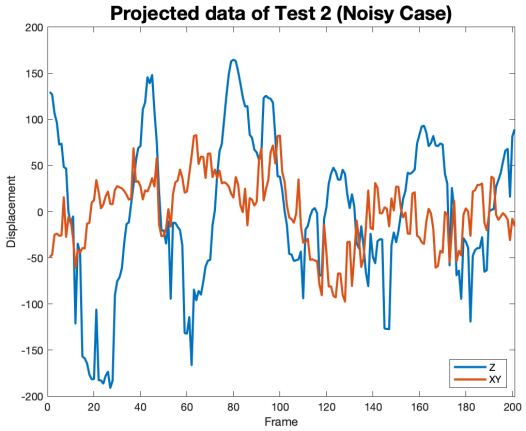

Fig 8: Projected data in the basis of the principal components in test 2

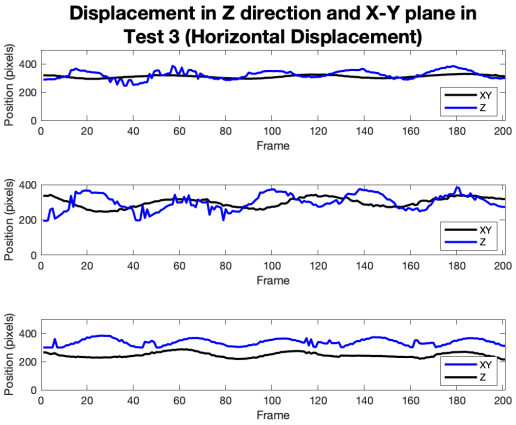

Fig 9: x (X-Y) and y (Z) positions of the can in test 3 for camera 1 (Top), camera 2 (Middle), and camera 3 (Bottom)

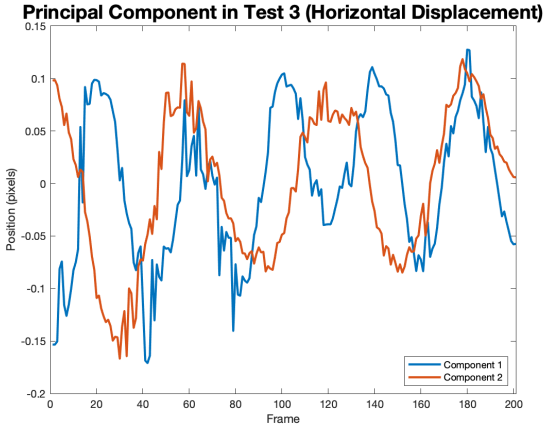

Fig 10: The first two principal components in test 3

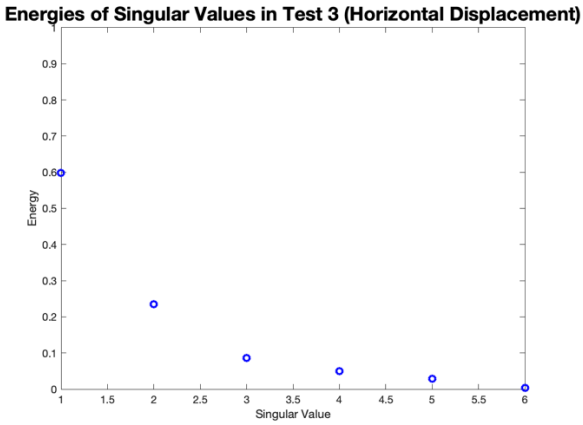

Fig 11: Energies captured at each singular value in test 3

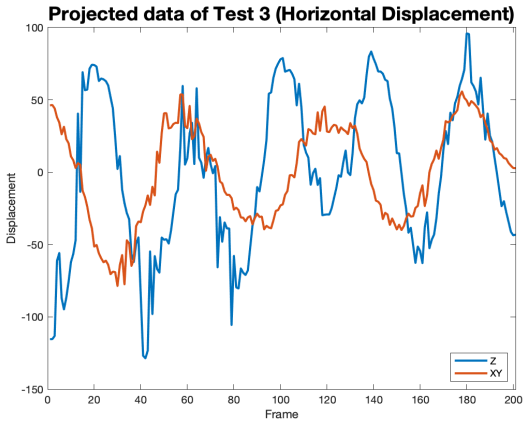

Fig 12: Projected data in the basis of the principal components in test 3

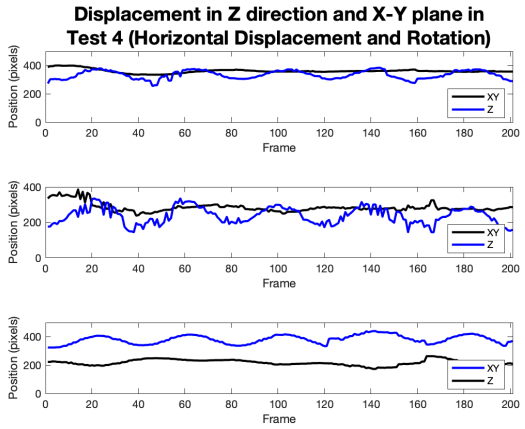

Fig 13: x (X-Y) and y (Z) positions of the can in test 4 for camera 1 (Top), camera 2 (Middle), and camera 3 (Bottom)

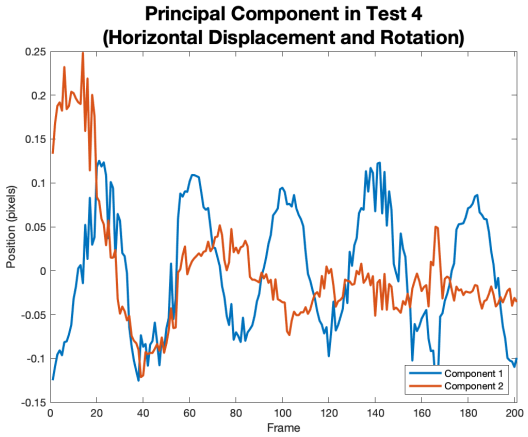

Fig 14: The first two principal components in test 4

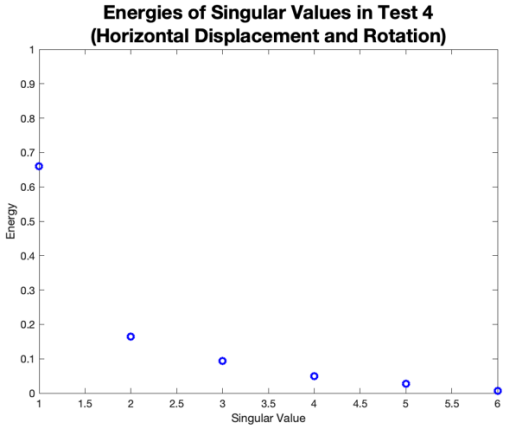

Fig 15: Energies captured at each singular value in test 4

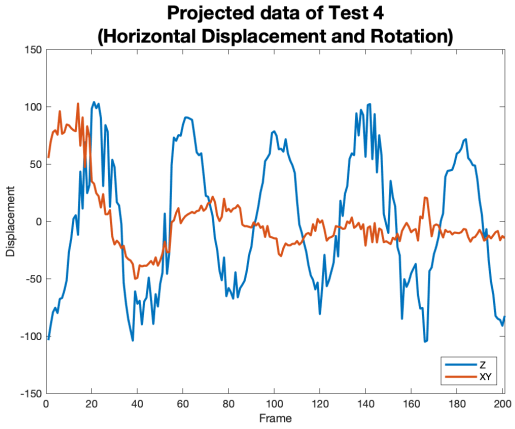

Fig 16: Projected data in the basis of the principal components in test 4

4.1 Test 1 (Ideal Case)

As shown in Fig. 1, there is a displacement of the can in z direction and almost no movement in the X-Y plane. Therefore, it is a one dimension movement. The first principal component shown in Fig.2 is enough to capture the oscillation. In Fig 3, rank 1 approximation can capture around 90% of the original data. Our projected data all show very similar pattern with the original data when describing the movement in z direction. As a result, PCA successfully reproduced the simple harmonic motion.

4.2 Test 2 (Noisy Case)

In test 2, camera shakes are introduced to the video recordings, so we can see both displacement in z direction and on the X-Y plane in Fig. 5. The oscillation in z direction is mostly larger than the on the X-Y plane, which was caused by the camera shakes. In Fig. 7, rank 1 approximation only captures around 70% of original data's energies, which is lower than that of test 1. Therefore, we plotted the first two principal components in Fig. 6. Although the camera shakes make it more difficult to extract the simple harmonic motion, we can still see the first component describe the motion quite well.

4.3 Test 3 (Horizontal displacement)

Instead of X-Y plane displacement caused by camera shakes in test 2, the paint can was released off-center to create both displacement in z direction and on the X-Y plane in test 3. As shown in Fig. 9, the oscillations in z direction is not as big as that in test 1, while the displacement on the X-Y plane in test 3 is larger than that in test 1. In Fig. 11, rank 1 approximation only captures around 60% of original data's energies, which is significantly lower than that of test 1. Since the second singular value captures 23% of the total energy, we plotted the first two principal components in Fig. 10. Similar to test 1 and 2, Fig. 9 and 12 show a larger movement of the can in z direction than that on the X-Y plane in test 3. After performing PCA, our projected data reproduces the can's movement in both vertical and horizontal directions.

4.4 Test 4 (Horizontal displacement and rotation)

In test 4, rotation and horizonal displacement are practiced. Therefore, the displacement plot in Fig. 13 is similar to Fig. 9 for test 3. The movement in z direction is smaller than that in test 1, while the movement on the X-Y plane is larger than that in test 1. In Fig. 15, rank 1 approximation only captures around 65% of original data's energies, which is significantly lower than that of test 1. Since the second singular value captures 17% of the total energy, we plotted the first two principal components in Fig. 14. Both the original data plot and our projected data show that movement in z direction stays almost the same throughout the video but the movement on the X-Y plane decreases as time increases. This matches the videos showing the movement of the paint can transits from a pendulum and rotational motion to a rotational simple harmonic motion.

5 Summary and Conclusions

In conclusion, PCA, an application of SVD allows us to reduce the redundancies of our data and its dimension. It saves a lot of time and space when the original data is very big. Although noises and adding horizontal displacement and rotation slightly affects the results in PCA, we can still accurately project the paint can's movement on a new basis by using the first two principal components. Since all cases' first singular values capture over 60% of the total energies, we can tell that the movement of the paint can in z direction is the largest despite the presence of other dimensional movements.

Appendix A. MATLAB functions

- `load filename` loads data from `filename`
- `implay(filename)` opens the Video Viewer app, displaying the content of the file specified by `filename`. The file can be an Audio Video Interleaved (AVI) file. The Video Viewer reads one frame at a time, conserving memory during playback. The Video Viewer does not play audio tracks.
- `sz = size(A)` returns a row vector whose elements are the lengths of the corresponding dimensions of `A`. For example, if `A` is a 3-by-4 matrix, then `size(A)` returns the vector `[3 4]`.
- `I = rgb2gray(RGB)` converts the truecolor image `RGB` to the grayscale image `I`. The `rgb2gray` function converts `RGB` images to grayscale by eliminating the hue and saturation information while retaining the luminance.
- `[row,col] = ind2sub(sz,ind)` returns the arrays `row` and `col` containing the equivalent row and column subscripts corresponding to the linear indices `ind` for a matrix of size `sz`. Here `sz` is a vector with two elements, where `sz(1)` specifies the number of rows and `sz(2)` specifies the number of columns.
- `X = zeros(sz1,...,szN)` returns an `sz1`-by-...-by-`szN` array of zeros where `sz1`, ... , `szN` indicate the size of each dimension. For example, `zeros(2,3)` returns a 2-by-3 matrix.
- `M = max(A)` returns the maximum elements of an array. If `A` is a vector, then `max(A)` returns the maximum of `A`. If `A` is a matrix, then `max(A)` is a row vector containing the maximum value of each column. If `A` is a multidimensional array, then `max(A)` operates along the first array dimension whose size does not equal 1, treating the elements as vectors. The size of this dimension becomes 1 while the sizes of all other dimensions remain the same. If `A` is an empty array whose first dimension has zero length, then `max(A)` returns an empty array with the same size as `A`.
- `[U,S,V] = svd(A,'econ')` produces an economy-size decomposition of m-by-n matrix `A`:
    - `m > n` — Only the first n columns of `U` are computed, and `S` is n-by-n.
    - `m = n` — `svd(A,'econ')` is equivalent to `svd(A)`.
    - `m < n` — Only the first m columns of `V` are computed, and `S` is m-by-m.
    - The economy-size decomposition removes extra rows or columns of zeros from the diagonal matrix of singular values, `S`, along with the columns in either `U` or `V` that multiply those zeros in the expression `A = U*S*V'`. Removing these zeros and columns can improve execution time and reduce storage requirements without compromising the accuracy of the decomposition.
- `D = diag(v)` returns a square diagonal matrix with the elements of vector `v` on the main diagonal.

Appendix B. MATLAB codes

```
% Clean workspace
clear all; close all; clc
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
%implay(vidFrames1_1)
numFrames1_1 = size(vidFrames1_1,4);
numFrames2_1 = size(vidFrames2_1,4);
numFrames3_1 = size(vidFrames3_1,4);
X1_1 = zeros(1,numFrames1_1);
Y1_1 = zeros(1,numFrames1_1);
for j = 1:numFrames1_1
    X = vidFrames1_1(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:300) = 0;
    gray(:,380:end) = 0;
    gray(1:180,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X1_1(j) = x_indx;
    Y1_1(j) = y_indx;
    %imshow(X); drawnow
end
X2_1 = zeros(1,numFrames2_1);
Y2_1 = zeros(1,numFrames2_1);
for j = 1:numFrames2_1
    X = vidFrames2_1(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:260) = 0;
    gray(:,340:end) = 0;
    gray(1:110,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X2_1(j) = x_indx;
    Y2_1(j) = y_indx;
    %imshow(X); drawnow
end
X3_1 = zeros(1,numFrames3_1);
Y3_1 = zeros(1,numFrames3_1);
for j = 1:numFrames3_1
    X = vidFrames3_1(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:280) = 0;
    gray(:,470:end) = 0;
    gray(330:end,:) = 0;
    gray(1:240,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X3_1(j) = x_indx;
    Y3_1(j) = y_indx;
    %imshow(X); drawnow
end
% align vids
[mini, minindx] = min(Y1_1(1:50));
X1_1 = X1_1(minindx:minindx+200);
Y1_1 = Y1_1(minindx:minindx+200);
[mini, minindx] = min(Y2_1(1:50));
```

```
X2_1 = X2_1(minindx:minindx+200);
Y2_1 = Y2_1(minindx:minindx+200);
[mini, minindx] = min(X3_1(1:50));
X3_1 = X3_1(minindx:minindx+200);
Y3_1 = Y3_1(minindx:minindx+200);
figure(1)
subplot(3,1,1)
plot(X1_1,'k', 'Linewidth', 2);
hold on;
plot(Y1_1,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
title("Displacement in Z direction and X-Y plane in Test 1 (Ideal case)",...
    'Fontsize', 18)
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,2)
plot(X2_1,'k', 'Linewidth', 2);
hold on;
plot(Y2_1,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 400])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,3)
plot(X3_1,'b', 'Linewidth', 2);
hold on;
plot(Y3_1,'k', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
print('Figure 1','-dpng');

% svd
M = [X1_1;Y1_1;X2_1; ...
     Y2_1;X3_1;Y3_1];
[m,n] = size(M);
Mavg = mean(M,2);
M = M - Mavg*ones(1,201);
[U,S,V] = svd(M'./sqrt(n-1), 'econ');
% energy
sig = diag(S);
energy1 = sig(1)^2/sum(sig.^2);
energy2 = sum(sig(1:2).^2)/sum(sig.^2);
energy3 = sum(sig(1:3).^2)/sum(sig.^2);

figure(2)
plot(U(:,1),'Linewidth',2)
title('Principal Component in Test 1 (Ideal Case)','Fontsize', 18)
xlabel('Frame')
ylabel('Position')
```

```
xlim([0 201])
print('Figure 2','-dpng');

figure(3)
plot(sig.^2/sum(sig.^2),'bo','Linewidth',2);
ylabel('Energy')
xlabel('Singular Value')
ylim([0 1])
title('Energies of Singular Values in Test 1 (Ideal Case)','Fontsize', 18)
print('Figure 3','-dpng');

figure(4)
plot(M'*V(:,1:2),'Linewidth',2);
xlabel('Frame')
ylabel('Displacement')
xlim([0 201])
title('Projected data of Test 1 (Ideal Case)','Fontsize', 18);
legend('Z','XY','Location','southeast')
print('Figure 4','-dpng');
%
%% Noisy case
clear all; close all; clc
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
%implay(vidFrames3_2);
numFrames1_2 = size(vidFrames1_2,4);
numFrames2_2 = size(vidFrames2_2,4);
numFrames3_2 = size(vidFrames3_2,4);
X1_2 = zeros(1,numFrames1_2);
Y1_2 = zeros(1,numFrames1_2);
for j = 1:numFrames1_2
    X = vidFrames1_2(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:300) = 0;
    gray(:,420:end) = 0;
    gray(1:230,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X1_2(j) = x_indx;
    Y1_2(j) = y_indx;
    %imshow(X); drawnow
end
X2_2 = zeros(1,numFrames2_2);
Y2_2 = zeros(1,numFrames2_2);
for j = 1:numFrames2_2
    X = vidFrames2_2(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:200) = 0;
    gray(:,420:end) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X2_2(j) = x_indx;
    Y2_2(j) = y_indx;
    %imshow(X); drawnow
end
X3_2 = zeros(1,numFrames3_2);
Y3_2 = zeros(1,numFrames3_2);
```

```
for j = 1:numFrames3_2
    X = vidFrames3_2(:,:,:,j);
    gray = rgb2gray(X);
    gray(1:180,:) = 0;
    gray(:,1:260) = 0;
    gray(:,450:end) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X3_2(j) = x_indx;
    Y3_2(j) = y_indx;
    %imshow(X); drawnow
end
[mini, minindx] = min(Y1_2(1:50));
X1_2 = X1_2(minindx:minindx+200);
Y1_2 = Y1_2(minindx:minindx+200);
[mini, minindx] = min(Y2_2(1:50));
X2_2 = X2_2(minindx:minindx+200);
Y2_2 = Y2_2(minindx:minindx+200);
[mini, minindx] = min(X3_2(1:50));
X3_2 = X3_2(minindx:minindx+200);
Y3_2 = Y3_2(minindx:minindx+200);

figure(5)
subplot(3,1,1)
plot(X1_2,'k', 'Linewidth', 2);
hold on;
plot(Y1_2,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
title("Displacement in Z direction and X-Y plane in Test 2 (Noisy case)",...
    'Fontsize', 18)
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,2)
plot(X2_2,'k', 'Linewidth', 2);
hold on;
plot(Y2_2,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 400])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,3)
plot(X3_2,'b', 'Linewidth', 2);
hold on;
plot(Y3_2,'k', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
print('Figure 5','-dpng');

M = [X1_2;Y1_2;X2_2; ...
```

```
     Y2_2;X3_2;Y3_2];
Mavg = mean(M,2);
M = M - Mavg*ones(1,201);
[U,S,V] = svd(M'/sqrt(200), 'econ');
% energy
sig = diag(S);
energy1 = sig(1)^2/sum(sig.^2);
energy2 = sum(sig(1:2).^2)/sum(sig.^2);
energy3 = sum(sig(1:3).^2)/sum(sig.^2);

figure(6)
plot(U(:,1:2),'Linewidth',2)
title('Principal Component in Test 2 (Noisy Case)','Fontsize', 18)
xlabel('Frame')
ylabel('Position')
legend('Component 1','Component 2','Location','southeast')
xlim([0 201])
print('Figure 6','-dpng');

figure(7)
plot(sig.^2/sum(sig.^2),'bo','Linewidth',2);
ylabel('Energy')
xlabel('Singular Value')
ylim([0 1])
title('Energies of Singular Values in Test 2 (Noisy Case)','Fontsize', 18)
print('Figure 7','-dpng');

figure(8)
plot(M'*V(:,1:2),'Linewidth',2);
xlabel('Frame')
ylabel('Displacement')
xlim([0 201])
title('Projected data of Test 2 (Noisy Case)','Fontsize', 18);
legend('Z','XY','Location','southeast')
print('Figure 8','-dpng');

%% Horizontal Displacement
clear all; close all; clc
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
%implay(vidFrames3_3);
numFrames1_3 = size(vidFrames1_3,4);
numFrames2_3 = size(vidFrames2_3,4);
numFrames3_3 = size(vidFrames3_3,4);
X1_3 = zeros(1,numFrames1_3);
Y1_3 = zeros(1,numFrames1_3);
for j = 1:numFrames1_3
    X = vidFrames1_3(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:280) = 0;
    gray(:,420:end) = 0;
    gray(1:220,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X1_3(j) = x_indx;
    Y1_3(j) = y_indx;
```

```
    %imshow(X); drawnow
end
X2_3 = zeros(1,numFrames2_3);
Y2_3 = zeros(1,numFrames2_3);
for j = 1:numFrames2_3
    X = vidFrames2_3(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:200) = 0;
    gray(:,420:end) = 0;
    gray(1:170,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X2_3(j) = x_indx;
    Y2_3(j) = y_indx;
    %imshow(X); drawnow
end
X3_3 = zeros(1,numFrames3_3);
Y3_3 = zeros(1,numFrames3_3);
for j = 1:numFrames3_3
    X = vidFrames3_3(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:300) = 0;
    gray(:,450:end) = 0;
    gray(1:190,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X3_3(j) = x_indx;
    Y3_3(j) = y_indx;
    %imshow(X); drawnow
end

[mini, minindx] = min(Y1_3(1:30));
X1_3 = X1_3(minindx:minindx+200);
Y1_3 = Y1_3(minindx:minindx+200);
[mini, minindx] = min(Y2_3(1:30));
X2_3 = X2_3(minindx:minindx+200);
Y2_3 = Y2_3(minindx:minindx+200);
[mini, minindx] = min(X3_3(1:30));
X3_3 = X3_3(minindx:minindx+200);
Y3_3 = Y3_3(minindx:minindx+200);

figure(9)
subplot(3,1,1)
plot(X1_3,'k', 'Linewidth', 2);
hold on;
plot(Y1_3,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
title({'Displacement in Z direction and X-Y plane in', ...
    'Test 3 (Horizontal Displacement)'},...
    'Fontsize', 18)
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,2)
plot(X2_3,'k', 'Linewidth', 2);
hold on;
```

```
plot(Y2_3,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 400])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,3)
plot(X3_3,'b', 'Linewidth', 2);
hold on;
plot(Y3_3,'k', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
print('Figure 9','-dpng');

M = [X1_3;Y1_3;X2_3; ...
     Y2_3;X3_3;Y3_3];
Mavg = mean(M,2);
M = M - Mavg*ones(1,201);
[U,S,V] = svd(M'/sqrt(200), 'econ');
% energy
sig = diag(S);
energy1 = sig(1)^2/sum(sig.^2);
energy2 = sum(sig(1:2).^2)/sum(sig.^2);
energy3 = sum(sig(1:3).^2)/sum(sig.^2);

figure(10)
plot(U(:,1:2),'Linewidth',2)
title('Principal Component in Test 3 (Horizontal Displacement)','Fontsize',
18)
xlabel('Frame')
ylabel('Position')
legend('Component 1','Component 2','Location','southeast')
xlim([0 201])
print('Figure 10','-dpng');

figure(11)
plot(sig.^2/sum(sig.^2),'bo','Linewidth',2);
ylabel('Energy')
xlabel('Singular Value')
ylim([0 1])
title('Energies of Singular Values in Test 3 (Horizontal
Displacement)','Fontsize', 18)
print('Figure 11','-dpng');

figure(12)
plot(M'*V(:,1:2),'Linewidth',2);
xlabel('Frame')
ylabel('Displacement')
xlim([0 201])
title('Projected data of Test 3 (Horizontal Displacement)','Fontsize', 18);
legend('Z','XY','Location','southeast')
print('Figure 12','-dpng');
```

```
%% Rotation and Horizontal Displacement
clear all; close all; clc
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
%implay(vidFrames3_4);
numFrames1_4 = size(vidFrames1_4,4);
numFrames2_4 = size(vidFrames2_4,4);
numFrames3_4 = size(vidFrames3_4,4);
X1_4 = zeros(1,numFrames1_4);
Y1_4 = zeros(1,numFrames1_4);
for j = 1:numFrames1_4
    X = vidFrames1_4(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:320) = 0;
    gray(:,450:end) = 0;
    gray(1:220,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X1_4(j) = x_indx;
    Y1_4(j) = y_indx;
    %imshow(X); drawnow
end
X2_4 = zeros(1,numFrames2_4);
Y2_4 = zeros(1,numFrames2_4);
for j = 1:numFrames2_4
    X = vidFrames2_4(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:210) = 0;
    gray(:,420:end) = 0;
    gray(1:140,:) = 0;
    gray(350:end,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X2_4(j) = x_indx;
    Y2_4(j) = y_indx;
    %imshow(X); drawnow
end
X3_4 = zeros(1,numFrames3_4);
Y3_4 = zeros(1,numFrames3_4);
for j = 1:numFrames3_4
    X = vidFrames3_4(:,:,:,j);
    gray = rgb2gray(X);
    gray(:,1:300) = 0;
    gray(:,480:end) = 0;
    gray(1:160,:) = 0;
    gray(300:end,:) = 0;
    [maxi, indx] = max(gray(:));
    [y_indx, x_indx] = ind2sub(size(gray),indx);
    X3_4(j) = x_indx;
    Y3_4(j) = y_indx;
    %imshow(X); drawnow
end
[mini, minindx] = min(Y1_4(1:50));
X1_4 = X1_4(minindx:minindx+200);
Y1_4 = Y1_4(minindx:minindx+200);
[mini, minindx] = min(Y2_4(1:50));
```

```
X2_4 = X2_4(minindx:minindx+200);
Y2_4 = Y2_4(minindx:minindx+200);
[mini, minindx] = min(X3_4(1:50));
X3_4 = X3_4(minindx:minindx+200);
Y3_4 = Y3_4(minindx:minindx+200);
figure(13)
subplot(3,1,1)
plot(X1_4,'k', 'Linewidth', 2);
hold on;
plot(Y1_4,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
title({'Displacement in Z direction and X-Y plane in', ...
    'Test 4 (Horizontal Displacement and Rotation)'},...
    'Fontsize', 18)
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,2)
plot(X2_4,'k', 'Linewidth', 2);
hold on;
plot(Y2_4,'b', 'Linewidth', 2);
xlim([0 201])
ylim([0 400])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
subplot(3,1,3)
plot(X3_4,'b', 'Linewidth', 2);
hold on;
plot(Y3_4,'k', 'Linewidth', 2);
xlim([0 201])
ylim([0 500])
xlabel("Frame")
ylabel("Position (pixels)")
legend('XY','Z','Location','southeast')
hold off;
print('Figure 13','-dpng');

M = [X1_4;Y1_4;X2_4; ...
     Y2_4;X3_4;Y3_4];
Mavg = mean(M,2);
M = M - Mavg*ones(1,201);
[U,S,V] = svd(M'/sqrt(200), 'econ');
% energy
sig = diag(S);
energy1 = sig(1)^2/sum(sig.^2);
energy2 = sum(sig(1:2).^2)/sum(sig.^2);
energy3 = sum(sig(1:3).^2)/sum(sig.^2);

figure(14)
plot(U(:,1:2),'Linewidth',2)
title({'Principal Component in Test 4', ...
    '(Horizontal Displacement and Rotation)'},'Fontsize', 18)
xlabel('Frame')
ylabel('Position')
```

```
legend('Component 1','Component 2','Location','southeast')
xlim([0 201])
print('Figure 14','-dpng');

figure(15)
plot(sig.^2/sum(sig.^2),'bo','Linewidth',2);
ylabel('Energy')
xlabel('Singular Value')
ylim([0 1])
title({'Energies of Singular Values in Test 4', ...
    '(Horizontal Displacement and Rotation)'},'Fontsize', 18)
print('Figure 15','-dpng');

figure(16)
plot(M'*V(:,1:2),'Linewidth',2);
xlabel('Frame')
ylabel('Displacement')
xlim([0 201])
title({'Projected data of Test 4', ...
    '(Horizontal Displacement and Rotation)'},'Fontsize', 18);
legend('Z','XY','Location','southeast')
print('Figure 16','-dpng');
```