PCA and a Spring-Mass System

Jiajun Bao

Abstract

This project aims to use the Principal Component Analysis (PCA) method to analyze the different datasets from videos of a spring-mass system recorded from cameras at three different angles. For each camera angle, we are provided with four separate cases. One of them is the ideal case, and the other ones have different variations, such as camera shakes. We will use PCA to compare and contrast the different instances.

1 Introduction and Overview

In this project, the goal is to use the Principal Component Analysis (PCA) method to analyze the spring-mass system under four tests with three different camera angles. The first test is the ideal case, in which the motion of the paint can is simple harmonic motion along the z-direction. The second test has the same motion as the first test, but the camera is shaking while recording. The third test is when the paint can is released off-center, which means there is motion in both z direction like the ideal case and motions in the x-y plane. The fourth test adds rotation when releasing off-center, so there is a rotation in the x-y plane and motion in the z-direction. For tracking the motion of the paint can, we extracting the coordinates of the light that is located on the top of the paint can at different time frames for all three cameras. This leads to a potential problem that the dimension of collected data from all three cameras is much higher than the actual data representing the motion. We utilize the low-rank approximations to find which principal components are the important ones that capture most of the energy. For the cases involving horizontal displacement or rotation, PCA will give us more information about the system's motion compared to the ideal case.

2 Theoretical Background

2.1 The Singular Value Decomposition

The Singular Value Decomposition (SVD) is a very powerful tool that is widely used in data analysis to produce low-rank approximations of a data set, which is described as following:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ 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 Σ are the singular values of the matrix A. The vectors u_n which makes up the columns of U are the left singular vectors of A. The vectors v_n which make up the columns of V are the right singular vectors of A. The SVD tells us how to keep the most amount of information while keeping the fewest number of dimension of the data. The large singular values help us pick out which directions are most important.

2.2 Principal Component Analysis

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 and each variables contains completely new information. This is what we need in this project as we have a six dimensions of data in the format: $\mathbf{X} = [x_a; y_a; x_b; y_b; x_c; y_c]$, which is composed by three sets of the x-y coordinates of the light on top of the paint can from three cameras in each time frame. This matrix has redundancies and PCA helps us to reduce the redundancies. We first compute the covariances matrix, and the eigenvalues of the covariance matrix are the squares of the scaled singular values, making it an unbiased estimator. The steps is shown in the following:

Consider

$$\mathbf{A} = \frac{\mathbf{1}}{\sqrt{\mathbf{n-1}}}\mathbf{X}, \mathbf{C_x} = \frac{\mathbf{1}}{\mathbf{n-1}}\mathbf{X}\mathbf{X^T} = \mathbf{A}\mathbf{A^T} = \mathbf{U}\mathbf{\Sigma^2}\mathbf{U^T} \tag{2}$$

The data in the new coordinates is

$$\mathbf{Y} = \mathbf{U^T X} \tag{3}$$

The covariance of $\mathbf{Y}$ is

$$\mathbf{C_x} = \frac{\mathbf{1}}{\mathbf{n-1}}\mathbf{YY^T} = \frac{\mathbf{1}}{\mathbf{n-1}}\mathbf{U^T X X^T U} = \mathbf{U^T A A^T U} = \mathbf{U^T U \Sigma^2 U^T U} = \mathbf{\Sigma^2} \tag{4}$$

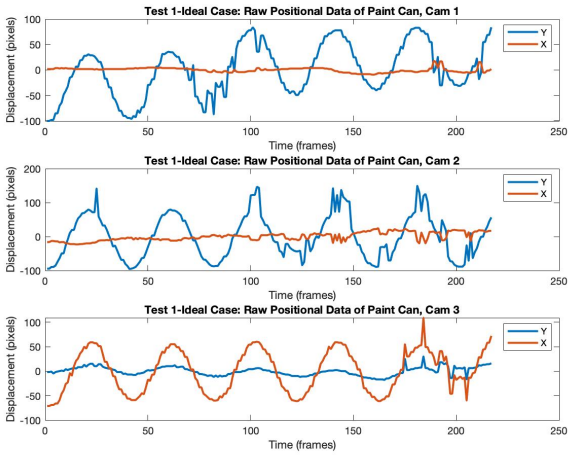
And the variables in $\mathbf{Y}$ are uncorrelated as off-diagonal elements of Σ are zero.

3 Algorithm Implementation and Development

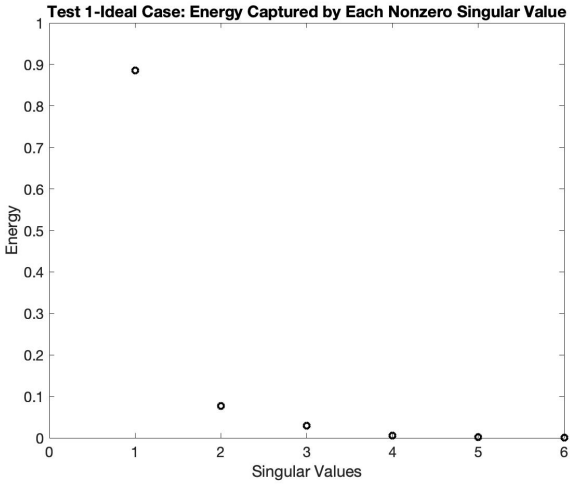
- Start from Test 1 - Ideal case, first load **cam1_1.mat**, and convert the movie to grey scale using *rgb2gray* and we view the video by creating a loop and displaying each frame and change the data to double precision for mathematical processing.
- After playing the video, we know that there is a bright spot that is on the top of the paint can, and we track the (x,y) coordinates of the bright spot in each video frame for each time to identify the motion of the paint can. This gives data x_1_1(t), y_1_1(t). The tracking is done using the idea of the Shannon Filter to crop the video frame down to only the region of the paint can moving trajectory through set other region's pixel values to 0. In terms of the video after cropping, it will display the grey scale of the target region, which is the region of mass moving, and other regions are blacked out.
- Then we use *max* function to find out the coordinates of bright spot in each time frame as the bright spot will have the highest pixel value. We repeat the same process for **cam2_1.mat** and **cam3_1.mat**, which are the ideal cases from Cam 2 and 3, and we get the data x_2_1(t), y_2_1(t),x_3_1(t), y_3_1(t).
- After getting all three set of (x,y) coordinates of the bright spot, for later Principal Component Analysis, we make all 3 cams start at same relative position by setting them staring near the highest point in first 25 frames and trim each movie clip down to the same size. Then we subtract the mean off from the data. After that, we aggregate these six vectors into a matrix called XY_total. We plot the raw data positional data of paint can from three cameras
- Now we do the Principal Component Analysis, we first take the svd of the data matrix XY_total and get the scaled singular values using *diag* function on S and then we find the energy captured by each nonzero singular value by dividing each squared singular value by the sum of all squared singular value.
- After plotting the energy captured by each nonzero singular value, we determine important singular values which would give us most of the information of the system and the other singular values that capture only little amount of the energy can be treated as noise. Then we plot the graph of V' and time frames as V' gives us information on the behavior of the movement along the principle components in time series on each principal components.
- We repeat the same steps for Test 2, Test 3, and Test 4. We use singular values that reaches 95% cumulative energy of the system as the cutoff to plot for the principal components.

4 Computational Results

Test 1: Ideal Case: Based on the above algorithms in section 3, we created the plot of raw positional data of the paint can in all three cameras, as shown in Figure 1(a). In the graph, the x and y coordinates are in intrinsic coordinates, which corresponds to the pixel indices. The X here relates to x-y plane in our world coordinates, and Y here representing z direction movements. We can tell that in the ideal case, the paint can is mainly moving in the z direction in a periodic motion as the shape of first components indicates. We are observing motion that is similar to simple harmonic motion. From figure1(b), we know that we have the first principle component which is significant in capturing the majority of the energy (88.52%), and the other principle components capture only little amount of the energy, and they can be treated as noise due to no video recording or data collecting is perfect.



(a) Raw Positional Data of Paint Can



(b) Energy Captured by Nonzero Singular Values

Figure 1: Test 1/Ideal Case: Raw Positional Data and Energy Captured

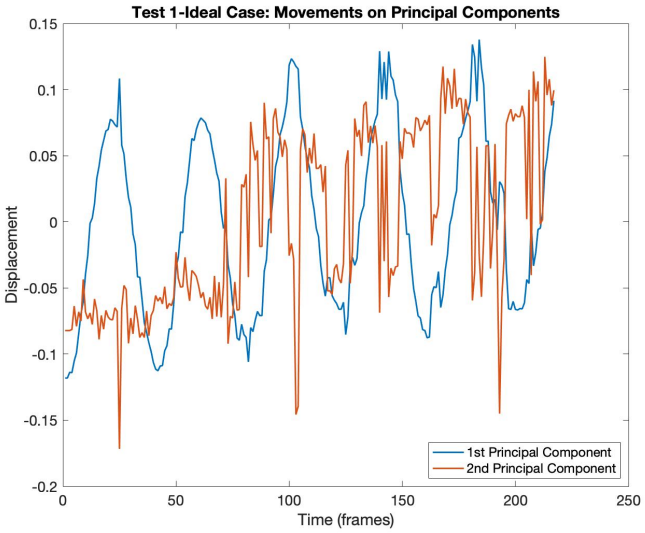
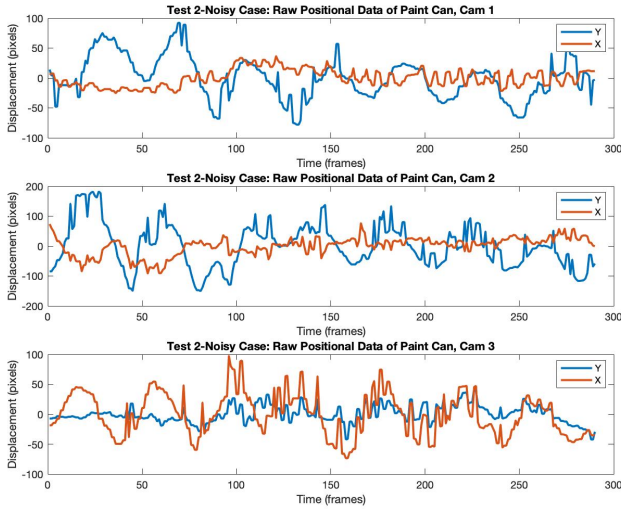


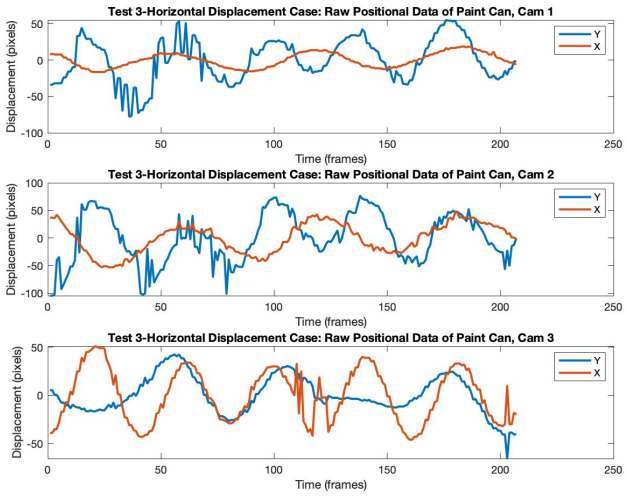
Figure 2: Test 1/Ideal Case: Movements on Principal Components

Test 2: Noisy Case: We follow the same procedures from Test 1 to create the plot of raw positional data of the paint can in all three cameras, as shown in Figure 3(a). Compared to the ideal case, based on figure 3(c), we now see three principle components that have significant meaning to our system if we use 95% cumulative energy of the system as the cutoff for giving information about the system motion. This is primarily because in this case the camera is greatly shaking while filming, which adds a significant amount of noise to our data despite the motion of the paint can itself didn't change much compared to test 1. The noise resulting from consistent shaking acts like a periodic behavior based on figure 3(e), but we still can observe that the paint can mainly move in the z-direction in a periodic motion as the shape of the first components indicates its similarity to test 1.

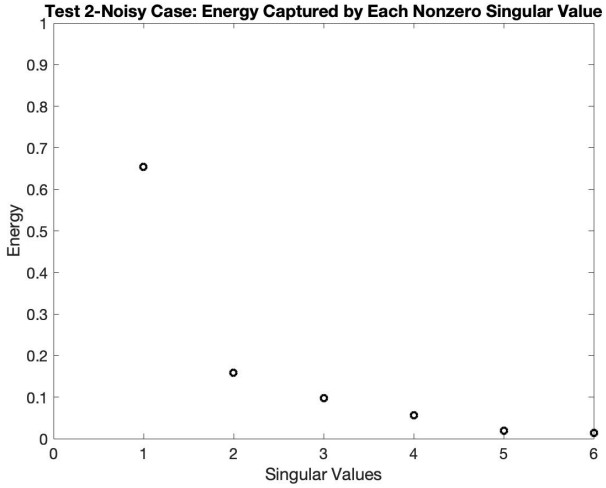
Test 3: Horizontal Displacement: The raw positional data of the paint can in all three cameras is shown in Figure 3(b). Based on figure 3(d), we now see four principal components that have significant meaning to our system if we use 95% cumulative energy of the system as the cutoff for giving information about the system motion. This is due to the off-center release, which makes the mass move in the x-y plane. However, we can still see the dominant up-down motion of the mass based on the first principle component in Figure3(f). Now we have a second relative large singular value describe the periodic motion in the x-y plane, and it is actual motion in the x-y plane compared to the noise from the shaking camera in the x-y plane from test 2.



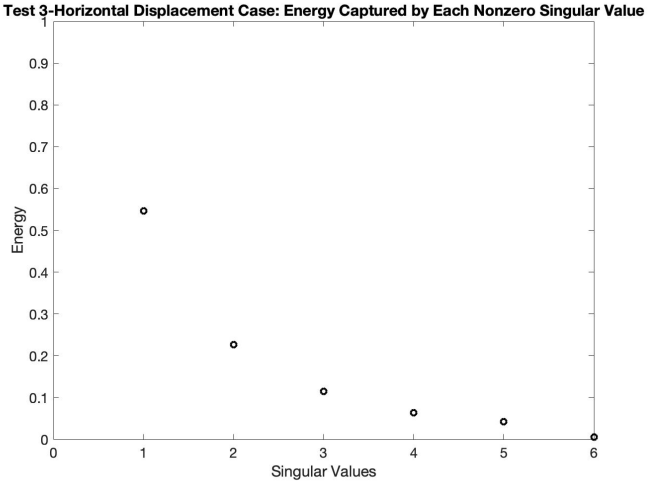
(a) Test 2: Raw Positional Data of Paint Can



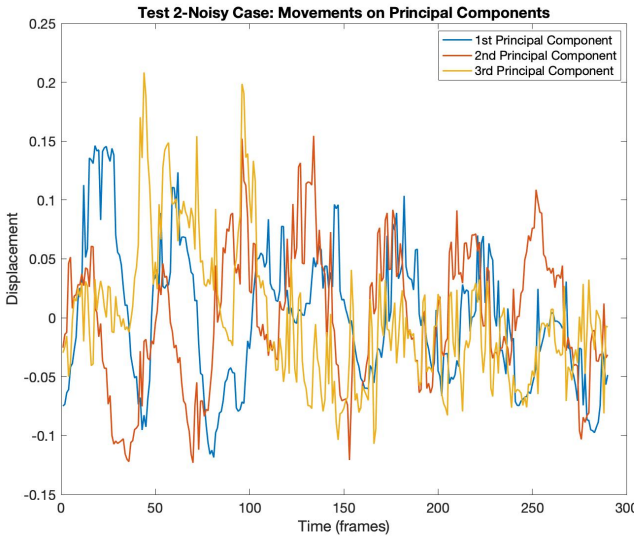
(b) Test 3: Raw Positional Data of Paint Can



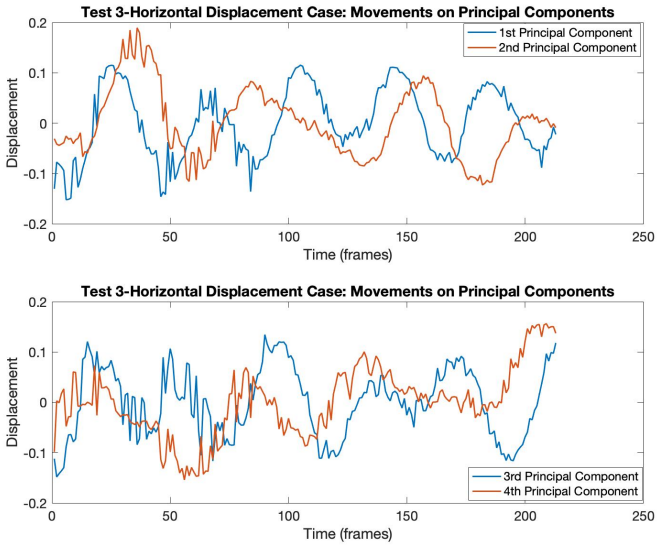
(c) Test 2: Energy Captured by Nonzero Singular Values



(d) Test 3: Energy Captured by Nonzero Singular Values



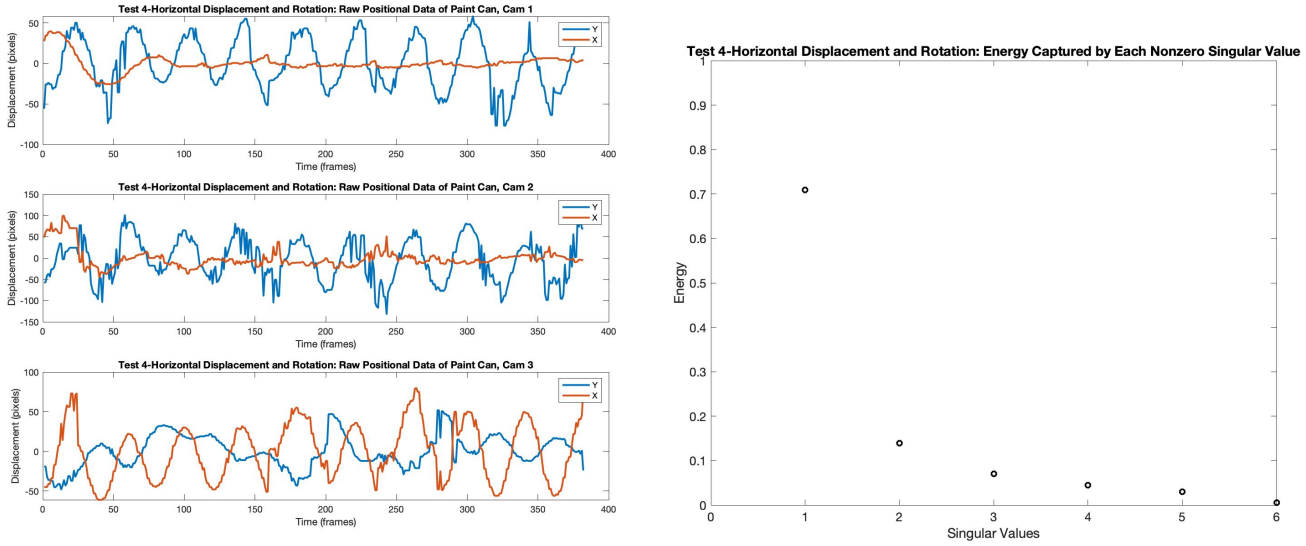
(e) Test 2: Movements on Principal Components



(f) Test 3: Movements on Principal Components

Figure 3: Test 2 & 3

Test 4: Horizontal Displacement and Rotation: The raw positional data of the paint can in all three cameras is shown in Figure 4(a), and we can tell that after adding rotation to the mass, the x direction movement of the mass is not as periodic as it was in test 3. Based on figure 4(b), we now see four principal components that have significant meaning to our system if we use 95% cumulative energy of the system as the cutoff for giving information about the system motion, this is similar to test 3 as the motion is get more complex compared to the ideal case in test 1 and noisy case in test 2. This is due to the off-center release and rotation motion, which moves in the xy plane. However, we can still see the dominant up-down motion of the mass based on the first principle component in Figure3(f).



(a) Raw Positional Data of Paint Can

(b) Energy Captured by Nonzero Singular Value

Figure 4: Test 4: Positional Data and Energy Captured

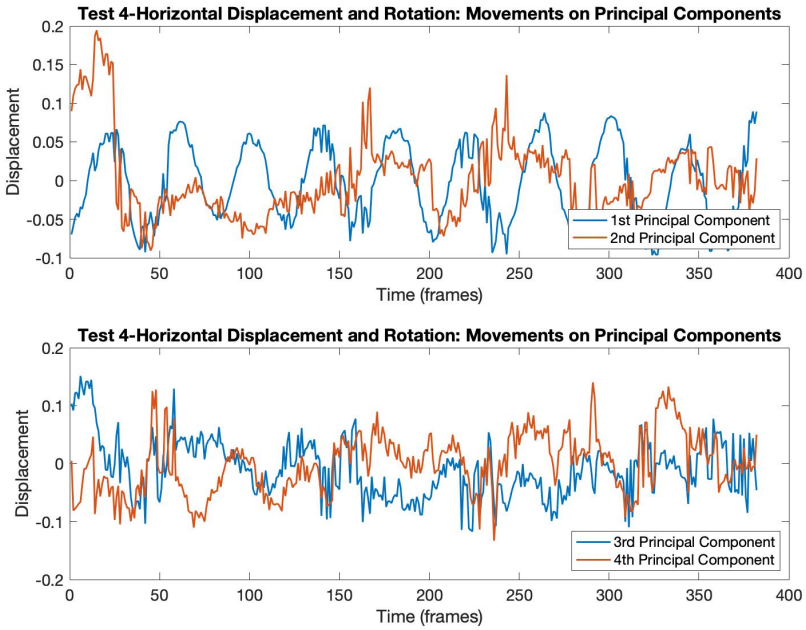


Figure 5: Test 4: Movements on Principal Components

5 Summary and Conclusions

In this problem, in order to apply Principal Component Analysis on different datasets, we first extract positions of the paint can from the video frames by tracking the bright spot on the top of the paint can and making a matrix containing six dimensions of data. We use SVD to eliminate the redundancies and utilize PCA to explore which directions are most important for different tests by looking at the principal components that capture the majority of the energy in the spring-mass system. We are also able to produce the plot for each test on how much energy of the system that each singular value captures. The PCA method is very useful when trying to eliminate redundancy and determining the dominant motion of the system and its characteristics.

Appendix A MATLAB Functions

- `I = rgb2gray(RGB)` converts the truecolor image RGB to the grayscale image I.
- `[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.
- `[U,S,V] = svd(A,'econ')` 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'

Appendix B MATLAB Code

```
clear all; close all; clc;
% Test 1: Ideal Case
load cam1_1.mat;  % -- Load Movie cam1_1 (Ideal Case from Cam 1)
[height1_1 width1_1 rgb num_frames1_1] = size(vidFrames1_1);
x_1_1 = [];
y_1_1 = [];
for j = 1:num_frames1_1  % -- Watch Movie
X = rgb2gray(vidFrames1_1(:,:,:,j));  % convert to grey scale
%imshow(X); drawnow
X1 = double(X);  % converted to double precision for mathematical processing
X1(:,1:300) = 0;  % crop the video to the target region
X1(:,400:end) = 0;
X1(1:200,:) = 0;
X1(450:end,:) = 0;
[M,I] = max(X1(:));
[y,x] = ind2sub(size(X1),I);
x_1_1 = [x_1_1 x];  % record the bright spot position in each frame
y_1_1 = [y_1_1 y];
end
load cam2_1.mat;  % -- Load Movie cam2_1 (Ideal Case from Cam 2)
[height2_1 width2_1 rgb num_frames2_1] = size(vidFrames2_1);
x_2_1 = [];
y_2_1 = [];
for j=1:num_frames2_1  % -- Watch Movie
X = rgb2gray(vidFrames2_1(:,:,:,j));  % convert to grey scale
%imshow(X); drawnow
X2 = double(X);  % converted to double precision for mathematical processing
X2(:,1:250) = 0;  % crop the video to the target region
X2(:,350:end) = 0;
X2(1:100,:) = 0;
X2(380:end,:) = 0;
[M,I] = max(X2(:));
[y,x] = ind2sub(size(X2),I);
x_2_1 = [x_2_1 x];  % record the bright spot position in each frame
y_2_1 = [y_2_1 y];
end
load cam3_1.mat;  % -- Load Movie cam3_1 (Ideal Case from Cam 3)
[height3_1 width3_1 rgb num_frames3_1] = size(vidFrames3_1);
x_3_1 = [];
y_3_1 = [];
for j=1:num_frames3_1  % -- Watch Movie
X = rgb2gray(vidFrames3_1(:,:,:,j));  % convert to grey scale
%imshow(X); drawnow
X3 = double(X);  % converted to double precision for mathematical processing
```

```
X3(:,1:270) = 0;  % crop the video to the target region
X3(:,480:end) = 0;
X3(1:230,:) = 0;
X3(340:end,:) = 0;
[M,I] = max(X3(:));
[y,x] = ind2sub(size(X3),I);
x_3_1 = [x_3_1 x];  % record the bright spot position in each frame
y_3_1 = [y_3_1 y];
end

% make all 3 cams start at same relative position
[Min,Ind] = min(y_1_1(1:25));
x_1_1 = x_1_1(Ind:end);
y_1_1 = y_1_1(Ind:end);
[Min,Ind] = min(y_2_1(1:25));
x_2_1 = x_2_1(Ind:end);
y_2_1 = y_2_1(Ind:end);
[Min,Ind] = min(x_3_1(1:25));
x_3_1 = x_3_1(Ind:end);
y_3_1 = y_3_1(Ind:end);
% trim each movie clip down to the same size
frames_min = min([length(x_1_1),length(x_2_1),length(x_3_1)]);
x_1_1 = x_1_1(1:frames_min);
x_1_1 = x_1_1 - mean(x_1_1);
y_1_1 = y_1_1(1:frames_min);
y_1_1 = y_1_1 - mean(y_1_1);
x_2_1 = x_2_1(1:frames_min);
x_2_1 = x_2_1 - mean(x_2_1);
y_2_1 = y_2_1(1:frames_min);
y_2_1 = y_2_1 - mean(y_2_1);
x_3_1 = x_3_1(1:frames_min);
x_3_1 = x_3_1 - mean(x_3_1);
y_3_1 = y_3_1(1:frames_min);
y_3_1 = y_3_1 - mean(y_3_1);
XY_total = [x_1_1; y_1_1; x_2_1; y_2_1; x_3_1; y_3_1];

figure(1)  % Plot the Raw Data Positional Data of Paint Can
subplot(3,1,1)
plot(1:frames_min, XY_total(2,:),1:frames_min, XY_total(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 1-Ideal Case: Raw Positional Data of Paint Can, Cam 1");
legend("Y", "X")
subplot(3,1,2)
plot(1:frames_min, XY_total(4,:),1:frames_min, XY_total(3,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 1-Ideal Case: Raw Positional Data of Paint Can, Cam 2");
legend("Y", "X")
subplot(3,1,3)
plot(1:frames_min, XY_total(6,:),1:frames_min, XY_total(5,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 1-Ideal Case: Raw Positional Data of Paint Can, Cam 3")
legend("Y", "X")

% PCA on test 1
[U,S,V] = svd(XY_total/sqrt(frames_min-1), 'econ');
sig = diag(S);
```

```
figure(2)
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title("Test 1-Ideal Case: Energy Captured by Each Nonzero Singular Value");
xlabel("Singular Values"); ylabel("Energy");
axis([0 6 0 1])
set(gca,'Fontsize',14,'Xtick',0:1:6)
% Create Projection on first three Principal Components
figure(3)
XY_proj = V';
plot(1:frames_min, XY_proj(1:2,:),'Linewidth',1.5)
title("Test 1-Ideal Case: Movements on Principal Components");
ylabel("Displacement"); xlabel("Time (frames)");
legend("1st Principal Component", "2nd Principal Component", ...
       "3rd Principal Component",'location','southeast');
set(gca,'Fontsize',14)

%% Test 2: Noisy Case.
clear all; close all; clc;
load cam1_2.mat;  % -- Load Movie cam1_2 (Noisy Case from Cam 1)
[height1_2 width1_2 rgb num_frames1_2] = size(vidFrames1_2);
x_1_2 = [];
y_1_2 = [];
for j = 1:num_frames1_2  % -- Watch Movie
X = rgb2gray(vidFrames1_2(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X1 = double(X);  % converted to double precision for mathematical processing
X1(:,1:300) = 0;  % crop the video to the target region
X1(:,410:end) = 0;
X1(1:220,:) = 0;
X1(450:end,:) = 0;
[M,I] = max(X1(:));
[y,x] = ind2sub(size(X1),I);
x_1_2 = [x_1_2 x];  % record the bright spot position in each frame
y_1_2 = [y_1_2 y];
end
load cam2_2.mat;  % -- Load Movie cam2_2 (Noisy Case from Cam 2)
[height2_2 width2_2 rgb num_frames2_2] = size(vidFrames2_2);
x_2_2 = [];
y_2_2 = [];
for j=1:num_frames2_2  % -- Watch Movie
X = rgb2gray(vidFrames2_2(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X2 = double(X);  % converted to double precision for mathematical processing
X2(:,1:200) = 0;  % crop the video to the target region
X2(:,400:end) = 0;
X2(1:50,:) = 0;
X2(450:end,:) = 0;
[M,I] = max(X2(:));
[y,x] = ind2sub(size(X2),I);
x_2_2 = [x_2_2 x];  % record the bright spot position in each frame
y_2_2 = [y_2_2 y];
end
load cam3_2.mat;  % -- Load Movie cam3_2 (Noisy Case from Cam 3)
[height3_2 width3_2 rgb num_frames3_2] = size(vidFrames3_2);
x_3_2 = [];
y_3_2 = [];
```

```
for j=1:num_frames3_2  % -- Watch Movie
X = rgb2gray(vidFrames3_2(:,:,:,j));  % convert to grey scale
%imshow(X); drawnow
X3 = double(X);  % converted to double precision for mathematical processing
X3(:,1:270) = 0;  % crop the video to the target region
X3(:,480:end) = 0;
X3(1:200,:) = 0;
X3(340:end,:) = 0;
[M,I] = max(X3(:));
[y,x] = ind2sub(size(X3),I);
x_3_2 = [x_3_2 x];  % record the bright spot position in each frame
y_3_2 = [y_3_2 y];
end

% make all 3 cams start at same relative position
[Min,Ind] = min(y_1_2(1:25));
x_1_2 = x_1_2(Ind:end);
y_1_2 = y_1_2(Ind:end);
[Min,Ind] = min(y_2_2(1:25));
x_2_2 = x_2_2(Ind:end);
y_2_2 = y_2_2(Ind:end);
[Min,Ind] = min(x_3_2(1:25));
x_3_2 = x_3_2(Ind:end);
y_3_2 = y_3_2(Ind:end);
% trim each movie clip down to the same size
frames_min = min([length(x_1_2),length(x_2_2),length(x_3_2)]);
x_1_2 = x_1_2(1:frames_min);
x_1_2 = x_1_2 - mean(x_1_2);
y_1_2 = y_1_2(1:frames_min);
y_1_2 = y_1_2 - mean(y_1_2);
x_2_2 = x_2_2(1:frames_min);
x_2_2 = x_2_2 - mean(x_2_2);
y_2_2 = y_2_2(1:frames_min);
y_2_2 = y_2_2 - mean(y_2_2);
x_3_2 = x_3_2(1:frames_min);
x_3_2 = x_3_2 - mean(x_3_2);
y_3_2 = y_3_2(1:frames_min);
y_3_2 = y_3_2 - mean(y_3_2);
XY_total = [x_1_2; y_1_2; x_2_2; y_2_2; x_3_2; y_3_2];

figure(1)  % Plot the Raw Data Positional Data of Paint Can
subplot(3,1,1)
plot(1:frames_min, XY_total(2,:),1:frames_min, XY_total(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 2-Noisy Case: Raw Positional Data of Paint Can, Cam 1");
legend("Y", "X")
subplot(3,1,2)
plot(1:frames_min, XY_total(4,:),1:frames_min, XY_total(3,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 2-Noisy Case: Raw Positional Data of Paint Can, Cam 2");
legend("Y", "X")
subplot(3,1,3)
plot(1:frames_min, XY_total(6,:),1:frames_min, XY_total(5,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 2-Noisy Case: Raw Positional Data of Paint Can, Cam 3")
legend("Y", "X")
```

```
% PCA on test 2
[U,S,V] = svd(XY_total/sqrt(frames_min-1), 'econ');
sig = diag(S);  % Calculating the energy of the truncations
figure(2)
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title("Test 2-Noisy Case: Energy Captured by Each Nonzero Singular Value");
xlabel("Singular Values"); ylabel("Energy");
axis([0 6 0 1])
set(gca,'Fontsize',14,'Xtick',0:1:6)
% Create Projection on first three Principal Components
figure(3)
XY_proj = V';
plot(1:frames_min, XY_proj(1:3,:),'Linewidth',1.5)
title("Test 2-Noisy Case: Movements on Principal Components");
ylabel("Displacement"); xlabel("Time (frames)");
legend("1st Principal Component", "2nd Principal Component", ...
       "3rd Principal Component",'location','southeast');
set(gca,'Fontsize',14)

%% Test 3: Horizontal Displacement Case
clear all; close all; clc;
load cam1_3.mat;  % -- Load Movie cam1_3 (Horizontal Displacement Case from Cam 1)
[height1_3 width1_3 rgb num_frames1_3] = size(vidFrames1_3);
x_1_3 = [];
y_1_3 = [];
for j = 1:num_frames1_3  % -- Watch Movie
X = rgb2gray(vidFrames1_3(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X1 = double(X);  % converted to double precision for mathematical processing
X1(:,1:280) = 0;  % crop the video to the target region
X1(:,400:end) = 0;
X1(1:225,:) = 0;
X1(380:end,:) = 0;
[M,I] = max(X1(:));
[y,x] = ind2sub(size(X1),I);
x_1_3 = [x_1_3 x];  % record the bright spot position in each frame
y_1_3 = [y_1_3 y];
end
load cam2_3.mat;  % -- Load Movie cam2_3 (Horizontal Displacement Case from Cam 2)
[height2_3 width2_3 rgb num_frames2_3] = size(vidFrames2_3);
x_2_3 = [];
y_2_3 = [];
for j=1:num_frames2_3  % -- Watch Movie
X = rgb2gray(vidFrames2_3(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X2 = double(X);  % converted to double precision for mathematical processing
X2(:,1:220) = 0;  % crop the video to the target region
X2(:,400:end) = 0;
X2(1:180,:) = 0;
X2(380:end,:) = 0;
[M,I] = max(X2(:));
[y,x] = ind2sub(size(X2),I);
x_2_3 = [x_2_3 x];  % record the bright spot position in each frame
y_2_3 = [y_2_3 y];
end
```

```
load cam3_3.mat;  % -- Load Movie cam3_3 (Horizontal Displacement Case from Cam 3)
[height3_3 width3_3 rgb num_frames3_3] = size(vidFrames3_3);
x_3_3 = [];
y_3_3 = [];
for j=1:num_frames3_3  % -- Watch Movie
X = rgb2gray(vidFrames3_3(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X3 = double(X);  % converted to double precision for mathematical processing
X3(:,1:270) = 0;  % crop the video to the target region
X3(:,480:end) = 0;
X3(1:170,:) = 0;
X3(320:end,:) = 0;
[M,I] = max(X3(:));
[y,x] = ind2sub(size(X3),I);
x_3_3 = [x_3_3 x];  % record the bright spot position in each frame
y_3_3 = [y_3_3 y];
end

% make all 3 cams start at same relative position
[Min,Ind] = min(y_1_3(1:25));
x_1_3 = x_1_3(Ind:end);
y_1_3 = y_1_3(Ind:end);
[Min,Ind] = min(y_2_3(1:25));
x_2_3 = x_2_3(Ind:end);
y_2_3 = y_2_3(Ind:end);
[Min,Ind] = min(x_3_3(1:25));
x_3_3 = x_3_3(Ind:end);
y_3_3 = y_3_3(Ind:end);
% trim each movie clip down to the same size
frames_min = min([length(x_1_3),length(x_2_3),length(x_3_3)]);
x_1_3 = x_1_3(1:frames_min);
x_1_3 = x_1_3 - mean(x_1_3);
y_1_3 = y_1_3(1:frames_min);
y_1_3 = y_1_3 - mean(y_1_3);
x_2_3 = x_2_3(1:frames_min);
x_2_3 = x_2_3 - mean(x_2_3);
y_2_3 = y_2_3(1:frames_min);
y_2_3 = y_2_3 - mean(y_2_3);
x_3_3 = x_3_3(1:frames_min);
x_3_3 = x_3_3 - mean(x_3_3);
y_3_3 = y_3_3(1:frames_min);
y_3_3 = y_3_3 - mean(y_3_3);
XY_total = [x_1_3; y_1_3; x_2_3; y_2_3; x_3_3; y_3_3];

figure(1)  % Plot the Raw Data Positional Data of Paint Can
subplot(3,1,1)
plot(1:frames_min, XY_total(2,:),1:frames_min, XY_total(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 3-Horizontal Displacement Case: Raw Positional Data of Paint Can, Cam 1");
legend("Y", "X")
subplot(3,1,2)
plot(1:frames_min, XY_total(4,:),1:frames_min, XY_total(3,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 3-Horizontal Displacement Case: Raw Positional Data of Paint Can, Cam 2");
legend("Y", "X")
subplot(3,1,3)
```

```
plot(1:frames_min, XY_total(6,:),1:frames_min, XY_total(5,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 3-Horizontal Displacement Case: Raw Positional Data of Paint Can, Cam 3")
legend("Y", "X")

% PCA on test 3
[U,S,V] = svd(XY_total/sqrt(frames_min-1), 'econ');
sig = diag(S);  % Calculating the energy of the truncations
figure(2)
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title("Test 3-Horizontal Displacement Case: Energy Captured by Each Nonzero Singular Value");
xlabel("Singular Values"); ylabel("Energy");
axis([0 6 0 1])
set(gca,'Fontsize',14,'Xtick',0:1:6)
% Create Projection on first three Principal Components
figure(3)
subplot(1,2,1)
XY_proj = V';
plot(1:frames_min, XY_proj(1:2,:),'Linewidth',1.5)
title("Test 3-Horizontal Displacement Case: Movements on Principal Components");
ylabel("Displacement"); xlabel("Time (frames)");
legend("1st Principal Component", "2nd Principal Component",'location','southeast');
set(gca,'Fontsize',14)
subplot(1,2,2)
plot(1:frames_min, XY_proj(3:4,:),'Linewidth',1.5)
title("Test 3-Horizontal Displacement Case: Movements on Principal Components");
ylabel("Displacement"); xlabel("Time (frames)");
legend("3rd Principal Component","4th Principal Component",'location','southeast');
set(gca,'Fontsize',14)
%% Test 4: Horizontal Displacement and Rotation.
clear all; close all; clc;
load cam1_4.mat;  % -- Load Movie cam1_4 (Horizontal Displacement and Rotation from Cam 1)
[height1_4 width1_4 rgb num_frames1_4] = size(vidFrames1_4);
x_1_4 = [];
y_1_4 = [];
for j = 1:num_frames1_4  % -- Watch Movie
X = rgb2gray(vidFrames1_4(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X1 = double(X);  % converted to double precision for mathematical processing
X1(:,1:320) = 0;  % crop the video to the target region
X1(:,460:end) = 0;
X1(1:225,:) = 0;
X1(400:end,:) = 0;
[M,I] = max(X1(:));
[y,x] = ind2sub(size(X1),I);
x_1_4 = [x_1_4 x];  % record the bright spot position in each frame
y_1_4 = [y_1_4 y];
end
load cam2_4.mat;  % -- Load Movie cam2_4 (Horizontal Displacement and Rotation from Cam 2)
[height2_4 width2_4 rgb num_frames2_4] = size(vidFrames2_4);
x_2_4 = [];
y_2_4 = [];
for j=1:num_frames2_4  % -- Watch Movie
X = rgb2gray(vidFrames2_4(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X2 = double(X);  % converted to double precision for mathematical processing
```

```
X(:,1:220) = 0;  % crop the video to the target region
X(:,400:end) = 0;
X(1:80,:) = 0;
X(380:end,:) = 0;
[M,I] = max(X2(:));
[y,x] = ind2sub(size(X2),I);
x_2_4 = [x_2_4 x];  % record the bright spot position in each frame
y_2_4 = [y_2_4 y];
end
load cam3_4.mat;  % -- Load Movie cam3_4 (Horizontal Displacement and Rotation from Cam 3)
[height3_4 width3_4 rgb num_frames3_4] = size(vidFrames3_4);
x_3_4 = [];
y_3_4 = [];
for j=1:num_frames3_4  % -- Watch Movie
X = rgb2gray(vidFrames3_4(:,:,:,j));  % convert to grey scale
% imshow(X); drawnow
X3 = double(X);  % converted to double precision for mathematical processing
X3(:,1:300) = 0;  % crop the video to the target region
X3(:,480:end) = 0;
X3(1:140,:) = 0;
X3(280:end,:) = 0;
[M,I] = max(X3(:));
[y,x] = ind2sub(size(X3),I);
x_3_4 = [x_3_4 x];  % record the bright spot position in each frame
y_3_4 = [y_3_4 y];
end
% make all 3 cams start at same relative position
[Min,Ind] = min(y_1_4(1:25));
x_1_4 = x_1_4(Ind:end);
y_1_4 = y_1_4(Ind:end);
[Min,Ind] = min(y_2_4(1:25));
x_2_4 = x_2_4(Ind:end);
y_2_4 = y_2_4(Ind:end);
[Min,Ind] = min(x_3_4(1:25));
x_3_4 = x_3_4(Ind:end);
y_3_4 = y_3_4(Ind:end);

% trim each movie clip down to the same size
frames_min = min([length(x_1_4),length(x_2_4),length(x_3_4)]);
x_1_4 = x_1_4(1:frames_min);
x_1_4 = x_1_4 - mean(x_1_4);
y_1_4 = y_1_4(1:frames_min);
y_1_4 = y_1_4 - mean(y_1_4);
x_2_4 = x_2_4(1:frames_min);
x_2_4 = x_2_4 - mean(x_2_4);
y_2_4 = y_2_4(1:frames_min);
y_2_4 = y_2_4 - mean(y_2_4);
x_3_4 = x_3_4(1:frames_min);
x_3_4 = x_3_4 - mean(x_3_4);
y_3_4 = y_3_4(1:frames_min);
y_3_4 = y_3_4 - mean(y_3_4);
% Deal with deal with outliers in cam2 -the frames bright light not appearing in some frames
for i  = 2:frames_min
    if abs(x_2_4(1,i) - x_2_4(1,i-1)) > 50
        x_2_4(1,i) = x_2_4(1,i-1);
        y_2_4(1,i) = y_2_4(1,i-1);
```

```
    end
end
XY_total = [x_1_4; y_1_4; x_2_4; y_2_4; x_3_4; y_3_4];
figure(1)  % Plot the Raw Data Positional Data of Paint Can
subplot(3,1,1)
plot(1:frames_min, XY_total(2,:),1:frames_min, XY_total(1,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 4-Horizontal Displacement and Rotation: Raw Positional Data of Paint Can, Cam 1");
legend("Y", "X")
subplot(3,1,2)
plot(1:frames_min, XY_total(4,:),1:frames_min, XY_total(3,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 4-Horizontal Displacement and Rotation: Raw Positional Data of Paint Can, Cam 2");
legend("Y", "X")
subplot(3,1,3)
plot(1:frames_min, XY_total(6,:),1:frames_min, XY_total(5,:), 'Linewidth', 2)
ylabel("Displacement (pixels)"); xlabel("Time (frames)");
title("Test 4-Horizontal Displacement and Rotation: Raw Positional Data of Paint Can, Cam 3")
legend("Y", "X")

% PCA on test 3
[U,S,V] = svd(XY_total/sqrt(frames_min-1), 'econ');
sig = diag(S);  % Calculating the energy of the truncations
figure(2)
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title("Test 4-Horizontal Displacement and Rotation: Energy Captured by Each Nonzero Singular Value");
xlabel("Singular Values"); ylabel("Energy");
axis([0 6 0 1])
set(gca,'Fontsize',14,'Xtick',0:1:6)
% Create Projection on first three Principal Components
figure(3)
subplot(2,1,1)
XY_proj = V';
plot(1:frames_min, XY_proj(1:2,:),'Linewidth',1.5)
title("Test 4-Horizontal Displacement and Rotation: Movements on Principal Components");
ylabel("Displacement"); xlabel("Time (frames)");
legend("1st Principal Component", "2nd Principal Component", ...
       "3rd Principal Component",'location','southeast');
set(gca,'Fontsize',14)
subplot(2,1,2)
plot(1:frames_min, XY_proj(3:4,:),'Linewidth',1.5)
title("Test 4-Horizontal Displacement and Rotation: Movements on Principal Components");
ylabel("Displacement"); xlabel("Time (frames)");
legend("3rd Principal Component","4th Principal Component",'location','southeast');
set(gca,'Fontsize',14)
```