Amath482 – Winter 2020

PCA

Chenxi Di (1663044)

February 21, 2020

Abstract

This study aims to study the physical behaviour of an oscillating hanging mass in different situations. Three cameras was used to capture its movement from three directions. By the use of Principle Component Analysis (PCA), we could extract the ideal or simplified behaviour even with the shaken cameras, added horizontal displacement or introduced rotation.

I. Introduction and Overview

The Principle Component Analysis (PCA) is a powerful tool to perform matrix dimensionality reduction. Based on the Singular Value Decomposition (SVD), PCA would extract the simplified behaviour through noisy and redundant data, and even transform the data into the optimal viewpoint for analysis. This project aims to study the physical behaviour of an oscillating hanging mass in different four scenarios. Three cameras was used to capture its movement from three directions. The first test is an ideal case where the mass, simply released downward, performed simple harmonic motion in z direction. But in the second test, we introduced camera shake into the video recording which generated more noise. In the third test, the mass was released off-center so there is both a pendulum motion and simple harmonic oscillations. In the 4th test, the mass was released off-center and rotated so as to produce motion in the x, y plane, rotation, and motion in the z direction. (Cameras are still in the last two tests) From here, we employed PCA to study the systems.

II. Theoretical Background

A singular value decomposition (SVD) is a factorization of a matrix into a some constitutive components all of which have useful meaning to the application. The full SVD decomposition of matrix A is:

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

where $\mathbf{U} \in C^{m\times m}$ is unitary, $\mathbf{V} \in C^{n\times n}$ is unitary and $\Sigma \in R^{m\times n}$ is diagonal.
To remove redundancy and identify the signals with maximal variance, we could diagonalize the covariance matrix, which is what the SVD does to reduce dimension. Note that the covariance matrix is

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^\mathbf{T} \tag{2}$$

where $\mathbf{X} \in C^{m\times n}$ where m are the measuring positions, and n is the number of data points at each position. The diagonal terms of $\mathbf{C_X}$ are the variances for measurements while the off-diagonal terms are the covariance. Large variances refers to dynamics of interest while low variances are considered to be non-interesting dynamics. By identifying the large variance, we could identify the most fluctuated points and therefore capture the important components.
To diagonalize the covariance matrix, consider the transformed variable

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

where U is the unitary transformation associated with the SVD.
Now Y has covariance matrix

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^\mathbf{T} \tag{4}$$

$$= \frac{1}{n-1}\mathbf{U}^*\mathbf{U}\Sigma^2\mathbf{U}\mathbf{U}^* \tag{5}$$

$$= \frac{1}{n-1}\Sigma^2 \tag{6}$$

From here, we can have diagonal variances.

III. Algorithm Implementation and Development

For each tests, I first loaded the data recorded by three cameras and save each video to a 4D tensor. From here, I converted it to double for each frame in order to perform analysis. By the use of image processing tool box, I carefully observe how the mass (a light

yellow point) moved. To more precisely measure its behaviour, I selected x range and y range (pixel) where the movement occurs. Then I extracted that portion of image during each frame. Since the exact movement is always accompanied with maximum value, I performed maximum search at that frame and saved its indices through ind2sub(). After obtaining the exact coordinate of the mass position, I moved to next frame and performed same algorithm. Ultimately, I would have an X vector and Y vector storing all the coordinate information about the mass for each camera, and I am able to plot the x and y displacement.

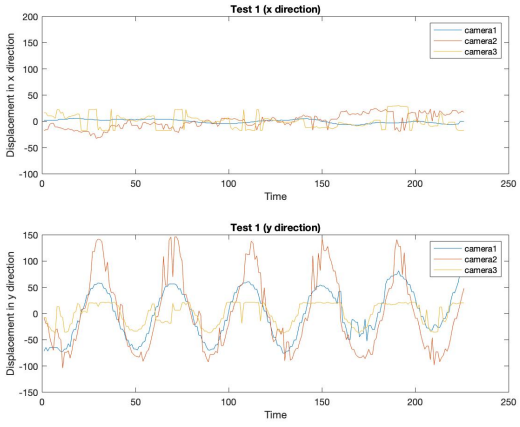
From here, I will construct a new matrix X with all 3 X1 and 3 Y1 vectors in each test and subtract the mean from each row, and employ SVD to get the U, V, Σ. From here, I could compute the variances for each component and plot the variances divided by variance sum to see which components are significant.

Note that the videos has different number of frame, to construct the matrix X (which will be used for SVD), I need to first adjust their lengths. So for each test, I recorded the minimum number of frames and cut the extra frames in other videos.

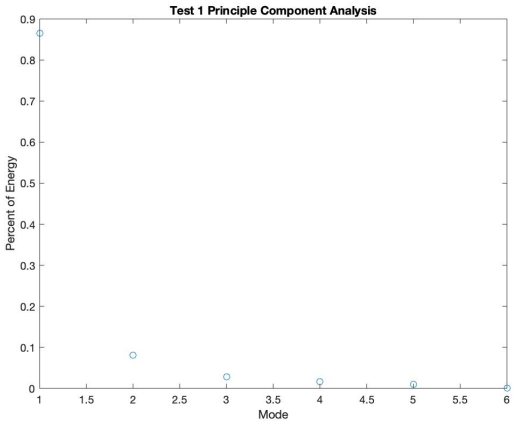
(Also notice that the third camera has reversed coordinates compared to other 2 cameras. So I flipped its vectors to construct the position plot.)

IV. Computational Results

Test 1 Results



(a) Mass Position in x and y direction



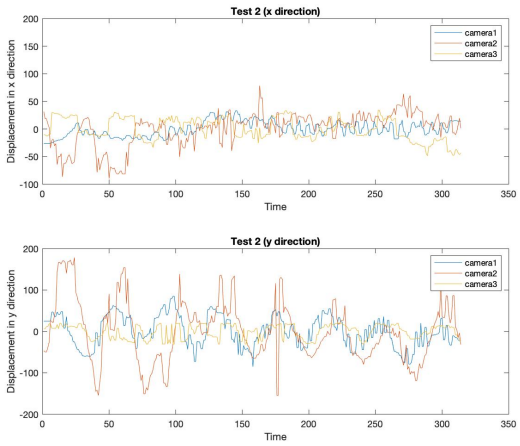
(b) Principal Components and Variances Percentage

Figure 1: Test 1

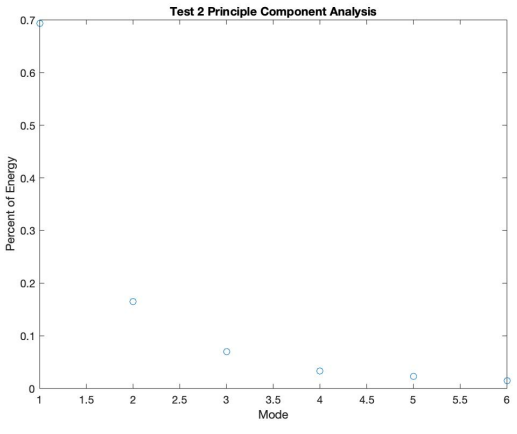
The blue line in the Position plot corresponds to the data from camera1, the red line corresponds to the data from camera2, the yellow line corresponds to the data from camera3. From the Figure1a, we see the mass oscillating along the y direction in all 3 cameras while stay almost at same position in x direction. This make sense since the 1 test is the ideal case where the mass is simply oscillating up and down. From the Principal Component and Variance plot (Figure1b), the first mode's variance makes up almost 90% among the total variances (or energy), which means we have one principal component of interest. And the rest of components have relatively low energy. This is plausible since the mass mainly perform z direction oscillation in the first test.

Test 2 Results

Compared to the Figure1 graph, we can see more noise in the Figure2a. This makes sense since in the case, all three cameras are shaken. However, the Figure2a still shows a static x direction and oscillation in y direction. The overall physical behaviour of the mass is similar to the first test. This makes sense since in test 2, the mass is also simply oscillating up and down. From the Principal Component and Variance plot (Figure2b), the first mode's variance makes up almost 70% among the total variances (or energy), which is less than the amount in first test. The second mode's variance makes up 20% among the total variances (or energy). Therefore, there may be two relatively significant principal components of interest.



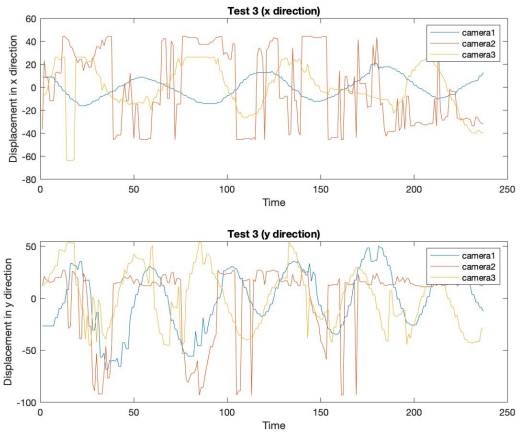
(a) Mass Position in x and y direction



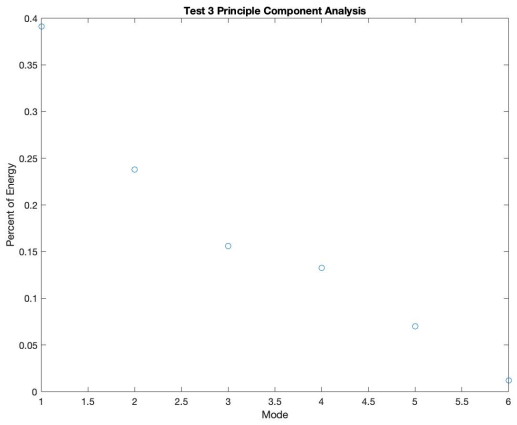
(b) Principal Components and Variances Percentage

Figure 2: Test 2

Test 3 Results



(a) Mass Position in x and y direction



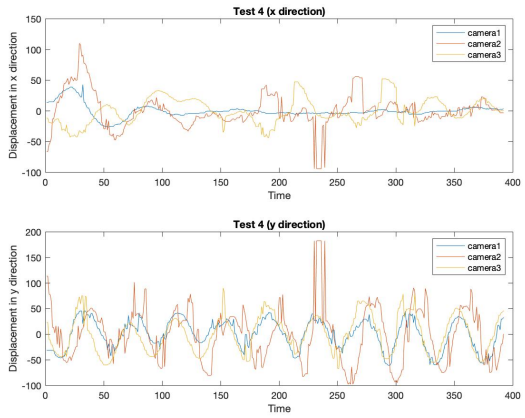
(b) Principal Components and Variances Percentage

Figure 3: Test 3

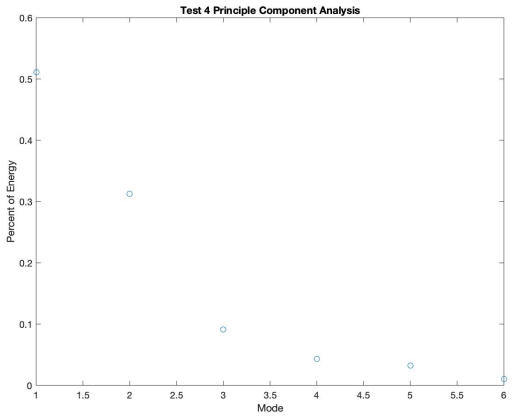
Different from first 2 plots, the Figure3a shows the mass is oscillating in both x and y direction. Indeed, the mass is released off center in the third test. So there will be x direction displacements. From Figure 3b, we can see the first mode's variance makes up 40% among the total variances (or energy) while the second mode takes 25%, the third mode takes 15%. Although they are less than the initial two tests, they are closed to each other. Therefore, the first three modes can be used to extract the main behaviour.

Test 4 Results

The Figure4a shows there is x and y direction oscillation in three cameras. Indeed, the mass is released off center and designed to rotate, so there will be x and y displacements. From Figure4b, the first mode's variance makes up 50% among the total variances (or energy) while the second mode takes 33%, the third mode takes 10%. These percentages are closed to each other. Therefore, the first three modes can be used to extract the main behaviour.



(a) Mass Position in x and y direction



(b) Principal Components and Variances Percentage

Figure 4: Test 4

V. Summary and Conclusions

Over the course of the study, we extracted the data from three cameras in four tests to take a glimpse of the mass movement. We plotted the mass displacement in both x and y direction. Furthermore, we performed PCA on the datasets to explore how many significant principal component exists in each scenarios. Principal Component Analysis is a powerful tool to reduce high dimensionality and help to extract the simplified behaviour.

Appendix A MATLAB functions used and brief implementation explanation

`ind2sub`: `ind2sub(size(img2), I)` returns a 2-array [newx, newy] containing the equivalent multidimensional subscripts corresponding to the linear indices index for the img2 object

`frame2im(F)`: returns the indexed image data X and associated colormap map from the single movie frame F.

`imshow()`: visualize images

`imtool()`: activate the image processing tool, help to later extract pixel indices

`svd()`: perform a singular value decomposition of matrix X/sqrt(n-1)

`repmat(mn,1, n)`: returns an array containing n copies of mn in 1 dimension

Appendix B MATLAB codes

```
clear all; close all; clc;
%% test 1
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');
numFrames1 = size(vidFrames1_1,4);
numFrames2 = size(vidFrames2_1,4);
numFrames3 = size(vidFrames3_1,4);

for k = 1 : numFrames1
    mov(k).cdata = vidFrames1_1(:,:,:,k);
    mov(k).colormap = [];
end

% draw each frame
%for j = 1:numFrames1
 %   img = frame2im(mov(j));
 %   imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

% first camera
X1 = [];
Y1 = [];
for j=1:numFrames1
    img = frame2im(mov(j));
    img_2 = double(img(:,300:420));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X1 = [X1 newx];
```

```
    Y1 = [Y1 newy];
end

% second camera
for k = 1 : numFrames2
    mov(k).cdata = vidFrames2_1(:,:,:,k);
    mov(k).colormap = [];
end
% draw
%for j = 1:numFrames2
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

X2 = [];
Y2 = [];

for j=1:numFrames2
    img = frame2im(mov(j));
    img_2 = double(img(:,250:340));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X2 = [X2 newx];
    Y2 = [Y2 newy];
end

% third camera
for k = 1 : numFrames3
    mov(k).cdata = vidFrames3_1(:,:,:,k);
    mov(k).colormap = [];
end
% draw
%for j = 1:numFrames3
```

```
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
img = frame2im(mov(3));
imtool(img)

X3 = [];
Y3 = [];

for j=1:numFrames3
    img = frame2im(mov(j));
    img_2 = double(img(240:320,290:350));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X3 = [X3 newx];
    Y3 = [Y3 newy];
end

minSize = 226; % 226, first camera size
X2 = X2(10:minSize+9); % shift the data vector by 10 frame
Y2 = Y2(10:minSize+9);
X3 = X3(1:minSize);
Y3 = Y3(1:minSize);

figure(1)
subplot(2,1,1);
plot(X1-mean(X1)), hold on
plot(X2-mean(X2))
plot(Y3-mean(Y3))
xlabel('Time')
ylabel('Displacement in x direction')
ylim([-100, 200])
title('Test 1 (x direction)')
legend('camera1','camera2','camera3');
subplot(2,1,2);
```

```
plot(Y1-mean(Y1)), hold on
plot(Y2-mean(Y2))
plot(X3-mean(X3))
xlabel('Time')
ylabel('Displacement in y direction')
title('Test 1 (y direction)')
legend('camera1','camera2','camera3');
hold off

% produce the principal components of interest using the SVD
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n] = size(X); % compute data size
mn = mean(X,2);  % compute mean for each row
X = X-repmat(mn,1,n); % subtract mean

[u,s,v]=svd(X'/sqrt(n-1)); % perform svd
lambda=diag(s).^2;   % diagnal variances
figure(2)
plot(lambda/sum(lambda),'o');  % percentage of energy in each mode
xlabel('Mode')
ylabel('Percent of Energy')
title('Test 1 Principle Component Analysis')

%% test 2
clear all; close all; clc;

load('cam1_2.mat');
load('cam2_2.mat');
load('cam3_2.mat');
numFrames1 = size(vidFrames1_2,4);
numFrames2 = size(vidFrames2_2,4);
numFrames3 = size(vidFrames3_2,4);

for k = 1 : numFrames1
    mov1(k).cdata = vidFrames1_2(:,:,:,k);
```

```
    mov1(k).colormap = [];
end

% draw each frame
% for j = 1:numFrames1
  %img = frame2im(mov(j));
  %imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

% first camera
X1 = [];
Y1 = [];
for j=1:numFrames1
    img = frame2im(mov1(j));
    img_2 = double(img(200:end,280:400));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X1 = [X1 newx];
    Y1 = [Y1 newy];
end

% second camera
for k = 1 : numFrames2
    mov2(k).cdata = vidFrames2_2(:,:,:,k);
    mov2(k).colormap = [];
end
% draw
%for j = 1:numFrames2
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)
```

```
X2 = [];
Y2 = [];

for j=1:numFrames2
    img = frame2im(mov2(j));
    img_2 = double(img(:,200:400));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X2 = [X2 newx];
    Y2 = [Y2 newy];
end

% third camera
for k = 1 : numFrames3
    mov3(k).cdata = vidFrames3_2(:,:,:,k);
    mov3(k).colormap = [];
end
% draw
%for j = 1:numFrames3
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

X3 = [];
Y3 = [];

for j=1:numFrames3
    img = frame2im(mov3(j));
    img_2 = double(img(210:320,310:360));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X3 = [X3 newx];
```

```
    Y3 = [Y3 newy];
end

minSize = 314; % 314, first camera size
X2 = X2(20:minSize + 19); % shift the data vector by 20 frame
Y2 = Y2(20:minSize + 19);
X3 = X3(1:minSize);
Y3 = Y3(1:minSize);

figure(1)
subplot(2,1,1);
plot(X1-mean(X1)), hold on
plot(X2-mean(X2))
plot(Y3-mean(Y3))
xlabel('Time')
ylabel('Displacement in x direction')
ylim([-100, 200])
title('Test 2 (x direction)')
legend('camera1','camera2','camera3');
subplot(2,1,2);
plot(Y1-mean(Y1)), hold on
plot(Y2-mean(Y2))
plot(X3-mean(X3))
xlabel('Time')
ylabel('Displacement in y direction')
title('Test 2 (y direction)')
legend('camera1','camera2','camera3');
hold off

% produce the principal components of interest using the SVD
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n] = size(X); % compute data size
mn = mean(X,2);  % compute mean for each row
X = X-repmat(mn,1,n); % subtract mean
```

```
[u,s,v]=svd(X'/sqrt(n-1)); % perform svd
lambda=diag(s).^2;   % diagnal variances
figure(2)
plot(lambda/sum(lambda),'o');  % percentage of energy in each mode
xlabel('Mode')
ylabel('Percent of Energy')
title('Test 2 Principle Component Analysis')
```

```
%% test 3
clear all; close all; clc;

load('cam1_3.mat');
load('cam2_3.mat');
load('cam3_3.mat');
numFrames1 = size(vidFrames1_3,4);
numFrames2 = size(vidFrames2_3,4);
numFrames3 = size(vidFrames3_3,4);

for k = 1 : numFrames1
    mov1(k).cdata = vidFrames1_3(:,:,:,k);
    mov1(k).colormap = [];
end

% draw each frame
% for j = 1:numFrames1
  %img = frame2im(mov(j));
  %imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

% first camera
```

```
X1 = [];
Y1 = [];
for j=1:numFrames1
    img = frame2im(mov1(j));
    img_2 = double(img(220:370,280:370));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X1 = [X1 newx];
    Y1 = [Y1 newy];
end

% second camera
for k = 1 : numFrames2
    mov2(k).cdata = vidFrames2_3(:,:,:,k);
    mov2(k).colormap = [];
end
% draw
%for j = 1:numFrames2
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

X2 = [];
Y2 = [];

for j=1:numFrames2
    img = frame2im(mov2(j));
    img_2 = double(img(230:350,180:270));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X2 = [X2 newx];
    Y2 = [Y2 newy];
end
```

```
% third camera
for k = 1 : numFrames3
    mov3(k).cdata = vidFrames3_3(:,:,:,k);
    mov3(k).colormap = [];
end
% draw
%for j = 1:numFrames3
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

X3 = [];
Y3 = [];

for j=1:numFrames3
    img = frame2im(mov3(j));
    img_2 = double(img(180:270,300:400));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X3 = [X3 newx];
    Y3 = [Y3 newy];
end

minSize = 237; % 237, third camera size
X2 = X2(30:minSize + 29); % shift the data vector by 30 frame
Y2 = Y2(30:minSize + 29);
X1 = X1(1:minSize);
Y1 = Y1(1:minSize);

figure(1)
subplot(2,1,1);
plot(X1-mean(X1)), hold on
```

```
plot(X2-mean(X2))
plot(Y3-mean(Y3))
xlabel('Time')
ylabel('Displacement in x direction')
title('Test 3 (x direction)')
legend('camera1','camera2','camera3');
subplot(2,1,2);
plot(Y1-mean(Y1)), hold on
plot(Y2-mean(Y2))
plot(X3-mean(X3))
xlabel('Time')
ylabel('Displacement in y direction')
title('Test 3 (y direction)')
legend('camera1','camera2','camera3');
hold off

% produce the principal components of interest using the SVD
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n] = size(X); % compute data size
mn = mean(X,2);  % compute mean for each row
X = X-repmat(mn,1,n); % subtract mean

[u,s,v]=svd(X'/sqrt(n-1)); % perform svd
lambda=diag(s).^2;   % diagnal variances
figure(2)
plot(lambda/sum(lambda),'o');  % percentage of energy in each mode
xlabel('Mode')
ylabel('Percent of Energy')
title('Test 3 Principle Component Analysis')

%% test 4
clear all; close all; clc;

load('cam1_4.mat');
load('cam2_4.mat');
```

```
load('cam3_4.mat');
numFrames1 = size(vidFrames1_4,4);
numFrames2 = size(vidFrames2_4,4);
numFrames3 = size(vidFrames3_4,4);

for k = 1 : numFrames1
    mov1(k).cdata = vidFrames1_4(:,:,:,k);
    mov1(k).colormap = [];
end

% draw each frame
% for j = 1:numFrames1
  %img = frame2im(mov(j));
  %imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

% first camera
X1 = [];
Y1 = [];
for j=1:numFrames1
    img = frame2im(mov1(j));
    img_2 = double(img(200:360,300:480));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X1 = [X1 newx];
    Y1 = [Y1 newy];
end

% second camera
for k = 1 : numFrames2
    mov2(k).cdata = vidFrames2_4(:,:,:,k);
    mov2(k).colormap = [];
end
```

```
% draw
%for j = 1:numFrames2
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

X2 = [];
Y2 = [];

for j=1:numFrames2
    img = frame2im(mov2(j));
    img_2 = double(img(110:390,140:400));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X2 = [X2 newx];
    Y2 = [Y2 newy];
end

% third camera
for k = 1 : numFrames3
    mov3(k).cdata = vidFrames3_4(:,:,:,k);
    mov3(k).colormap = [];
end
% draw
%for j = 1:numFrames3
% img = frame2im(mov(j));
% imshow(img); drawnow
%end
%img = frame2im(mov(3));
%imtool(img)

X3 = [];
Y3 = [];
```

```
for j=1:numFrames3
    img = frame2im(mov3(j));
    img_2 = double(img(150:300,300:500));
    [~,I]=max(img_2(:));
    [newy,newx]=ind2sub(size(img_2),I);
    X3 = [X3 newx];
    Y3 = [Y3 newy];
end

minSize = 392; % 392, first camera size
X2 = X2(1:minSize);
Y2 = Y2(1:minSize);
X3 = X3(1:minSize);
Y3 = Y3(1:minSize);

figure(1)
subplot(2,1,1);
plot(X1-mean(X1)), hold on
plot(X2-mean(X2))
plot(Y3-mean(Y3))
xlabel('Time')
ylabel('Displacement in x direction')
title('Test 4 (x direction)')
legend('camera1','camera2','camera3');
subplot(2,1,2);
plot(Y1-mean(Y1)), hold on
plot(Y2-mean(Y2))
plot(X3-mean(X3))
xlabel('Time')
ylabel('Displacement in y direction')
title('Test 4 (y direction)')
legend('camera1','camera2','camera3');
hold off
```

```
% produce the principal components of interest using the SVD
X = [X1;Y1;X2;Y2;X3;Y3];
[m,n] = size(X); % compute data size
mn = mean(X,2);  % compute mean for each row
X = X-repmat(mn,1,n); % subtract mean

[u,s,v]=svd(X'/sqrt(n-1)); % perform svd
lambda=diag(s).^2;   % diagnal variances
figure(2)
plot(lambda/sum(lambda),'o');  % percentage of energy in each mode
xlabel('Mode')
ylabel('Percent of Energy')
title('Test 4 Principle Component Analysis')
```

Reference

MathWorks: https://www.mathworks.com/help/matlab/ref/

J. Nathan Kutz. (2013) Data-Driven Modeling and Scientific Computation: Methods for Complex Systems and Big Data. Oxford University Press. ISBN:978-0-19-966034-6