Chingpo Lin
Amath482
HW3

Abstract

There are 4 kinds of system of spring for me to explore, they are in ideal, noisy, horizontal displacement, and horizontal displacement with rotation. I will explore the PCA (Principle Component Analysis) of these four systems. Before using PCA, it is necessary to get the mass position from the system in the video. In using the PCA, I will know its practical use and the effect of noise on that algorithm.

Introduction and Overview

This time, I was given 4 separate Video to analyze the motion of spring-mass system. By using Shannon filter I can filter out useless part in the video, and SVD allows me to see more clear into the system. All of these will allow to me describe different aspect of the PCA in these four systems. In PCA, we will subtract the mean by its column, which is important because we make the graph centralized.

Theoretical Background

This time I will explore the spring system with PCA. The spring system, which has a formula of:

$$z(t) = Acos(\omega t + \phi) \text{ (1)}$$

With A, and ϕ is the initial displacement and velocity of the mass, and ω is the frequency where remains constant in different system. There is also a useful formulation which is singular value decomposition (SVD), which decomposition the matrix A into 3 Important parts:

$$A = U\sum V^* \quad (2)$$

By seeing the rank n approximation, we can know the energy, and the energy can also get from formulation:

$$energy_N = \frac{\sigma_1^2 + \ldots + \sigma_N^2}{\sigma_1^2 + \ldots + \sigma_r^2} \quad (3)$$

Where we can easily know the percentage of initial image or a moment in a video which is useful. Also, the principle component analysis can connect the SVD of AA^T by:

$$C_X = \frac{1}{n-1} XX^T = AA^T \quad (4)$$

Because we transform our data by change the basis to work in the basic of principle component, and the data change into Y by:

$$Y = U^T X \quad (5)$$

We have to convert C_X into C_Y, and the covariance for Y is:

$$C_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1} U^T XX^T U = U^T AA^T U$$

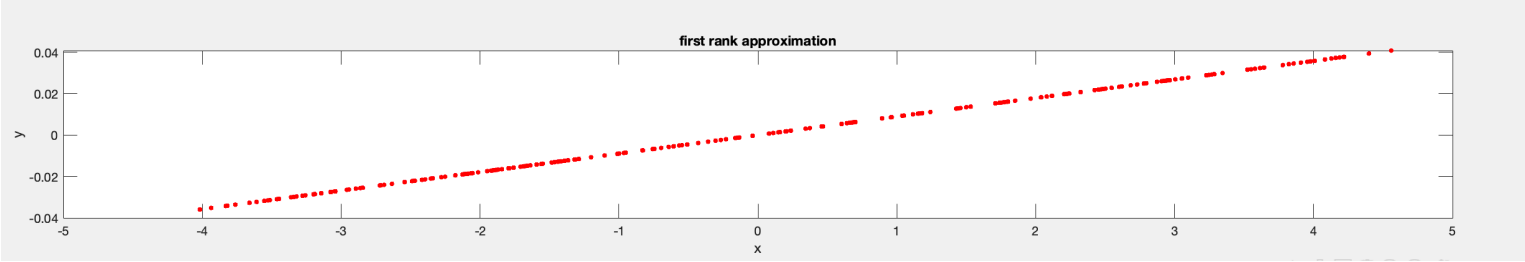
$$= U^T U(\sum)^2 U^T U = (\sum)^2$$

So, the covariance for our transformed data is just the square of the middle variable in SVD.

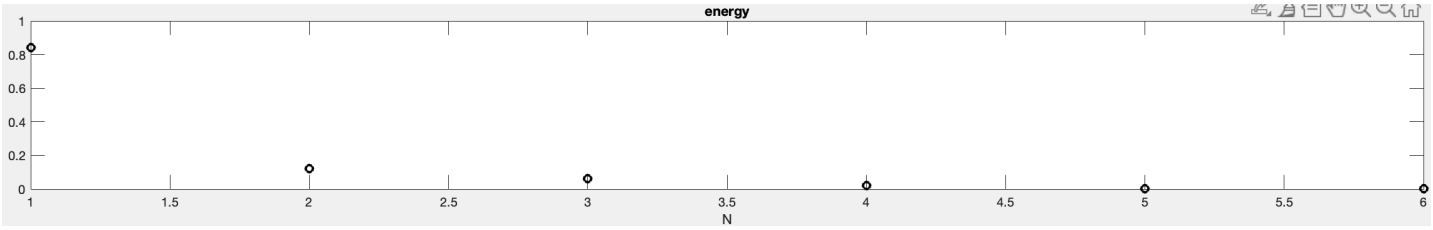
Algorithm Implementation and Development

First, we read all of the Video, and then we filter out the parts we don't want cause the only part we want is the spring-mass system. Shannon filter allow us to realize it. By taking 1 on the pixel we want and 0 we don't, we can get rid of useless factors in this video. I see clearly on the Video and data, and then find out the range of filter for each different N in different cases. Then, for each of the frame of each of video, I remove the color, filter the useless part, and turn it into unit8 form. Also, because the white has a color rgb of 255, I filter out the color has great different from this spring-mass system using find(). Then, I find the corresponding x-y coordinate in 480x640. Then for the result, I collect all of the coordinate and subtract the mean by its column, which is important in PCA due to centralization. Then, I plot the first rank approximation graph in 2d, the change of energy with increasing N by the equation above, displacement of the system spring in 2D with x-y plane and z, and the PC1 of each video.

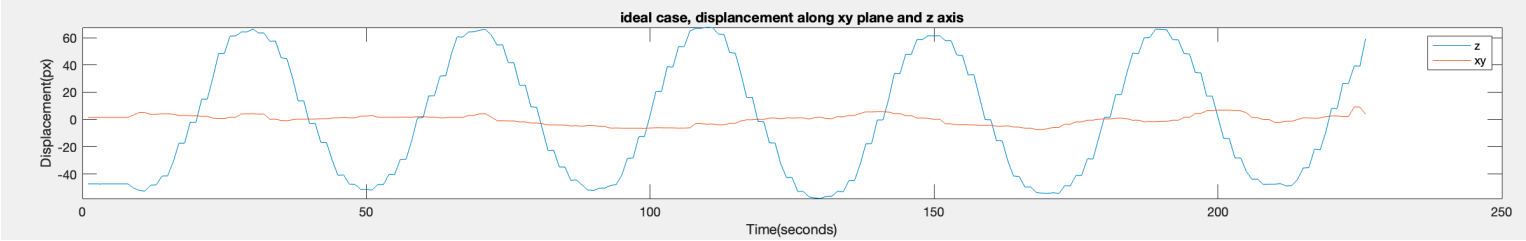
Computational Result



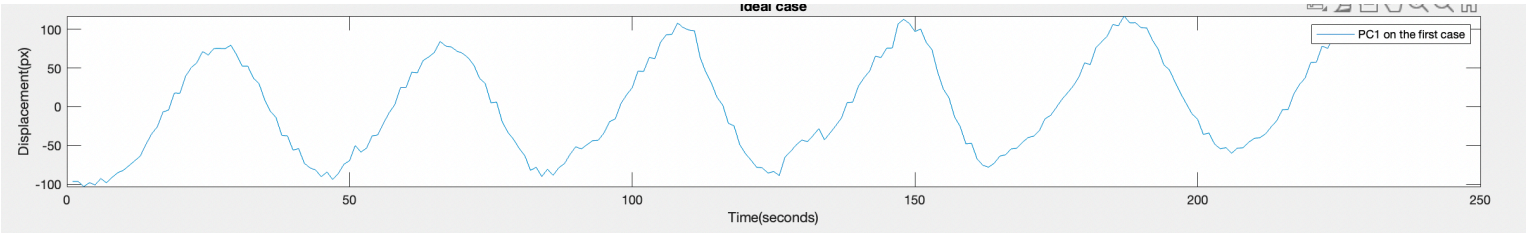
figure(1): rank1 approximation graph of test1



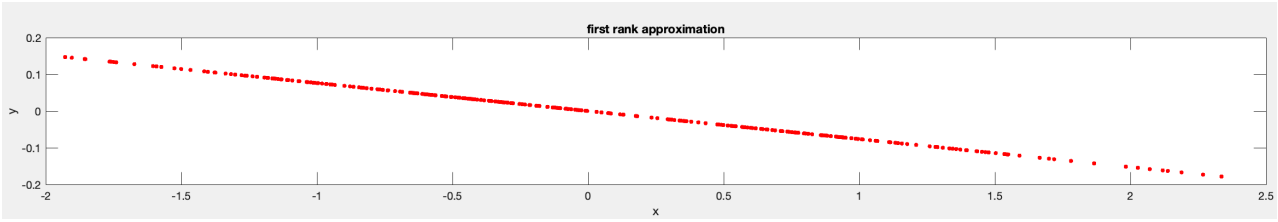
figure(2): energy change with rank N of test1



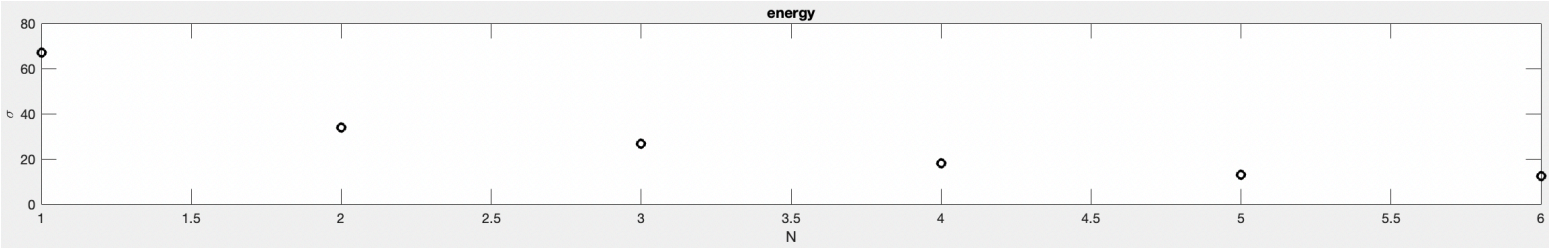
figure(3): displacement of xy-plane and z in 2D of test1



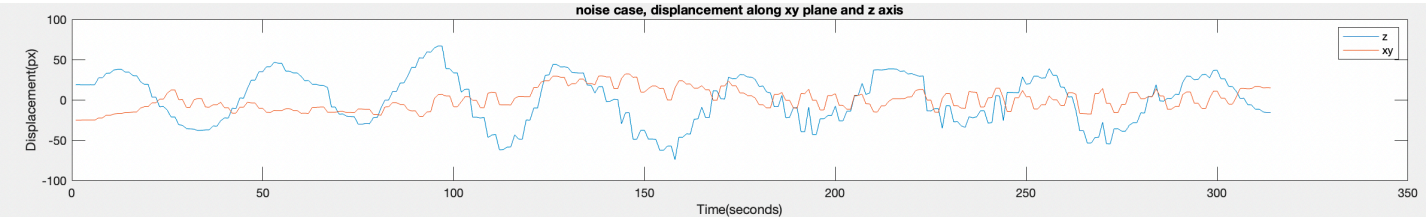
figure(4): PC1 on the test1



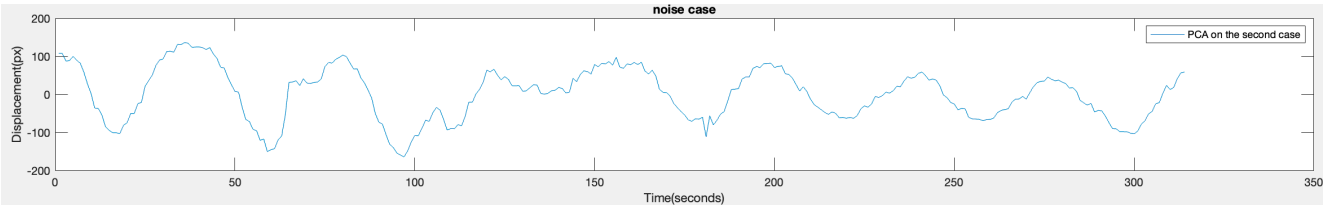
figure(5): rank1 approximation graph of test2



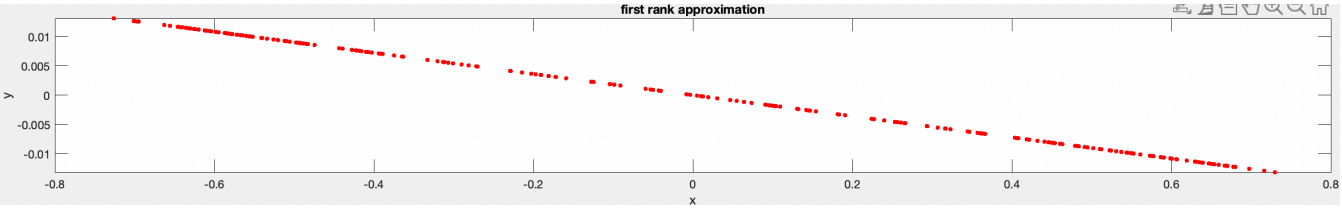
figure(6): energy change with rank N of test2



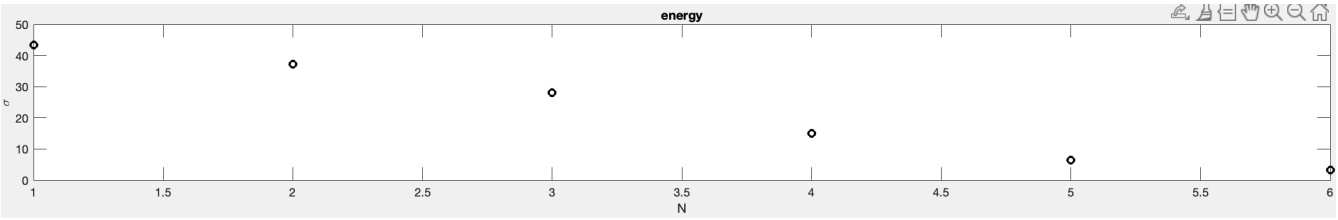
figure(7): displacement of xy-plane and z in 2D of test2



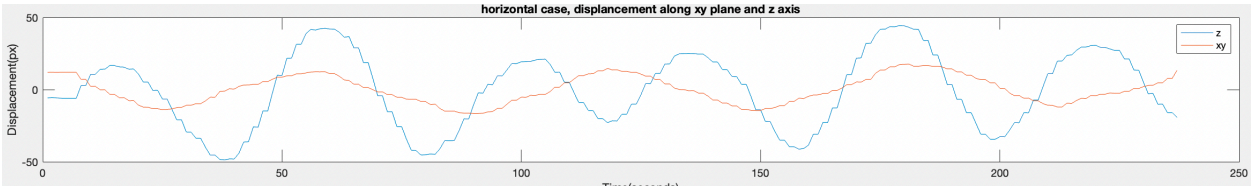
figure(8): PC1 on the test2



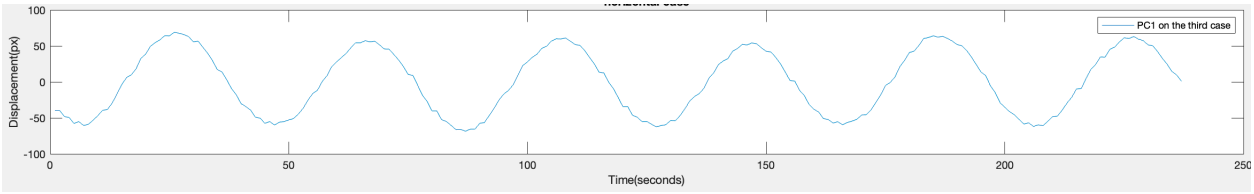
figure(9): rank1 approximation graph of test3



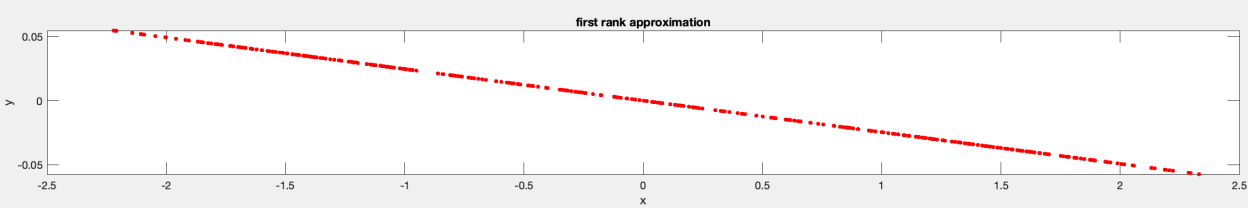
figure(10): energy change with rank N of test3



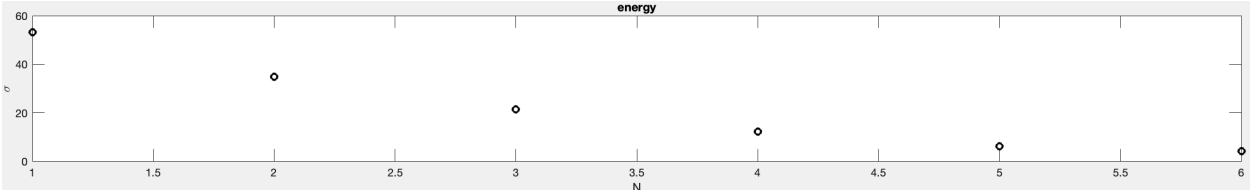
figure(11): displacement of xy-plane and z in 2D of test3



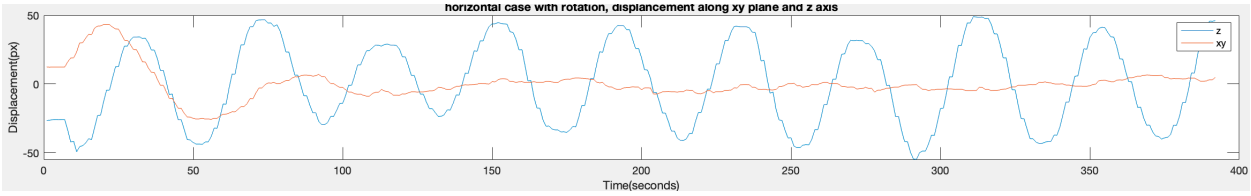
figure(12): PC1 on the test3



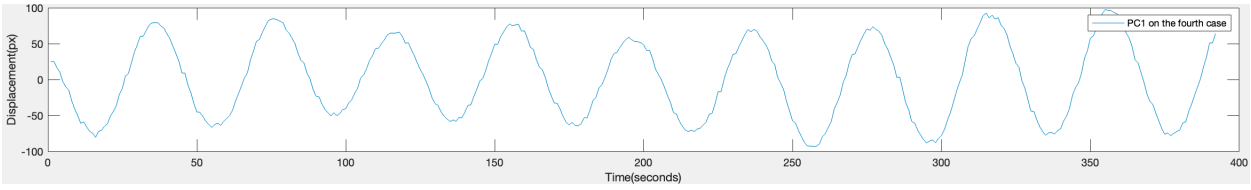
figure(13): rank1 approximation graph of test4



figure(14): energy change with rank N of test4



figure(15): displacement of xy-plane and z in 2D of test4



figure(16): PC1 on the test4

As, we see except the first case, all of the rest has a negative slope of the first approximation. For the second graph of energy of each case, we see that most of

energy in the first three approximation of idea case. However, in the noise case, even the sixth rank has about 10 percent energy which is large. This means noise can have a great impact on the performance of SVD. also we see the displacement graph, all of ideal and horizontal has clear position change, but noise is weird also in the PC1 plot.

Summary and Conclusions

This means noise can have great impact on PCA algorithms, and for other case, PCA algorithm can successfully get the path plot and even have an ideal of energy distribution. When we see the displacement of ideal, we find that xy keeps constant and z change which is same as what we saw in video. But for the noise case, because the camera keep moving, the algorithms are hard to detect if this is a movement of system. And for horizontal case, we can see that it xy change and z keep going between changes and stay constant. In conclusion, noise indeed affect our PCA algorithms.

Appendix A - function in code

zeros: create a vector or matrix with all 0 in given dimension
rgb2gray: remove the rgb color of image.
Im2double: convert all value to double
Im2unit8: convert image into 8-bit integer
mean: get the mean of given array or matrix

Appendix B - Matlab code

```
%% load ideal case camN_1

% Clean workspace

clear all; close all; clc
```

```
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
%implay(vidFrames1_1)
%implay(vidFrames2_1)
%implay(vidFrames3_1)

%% load noisy case camN_2
% Clean workspace
clear all; close all; clc
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
%implay(vidFrames1_2)
%implay(vidFrames2_2)
%implay(vidFrames3_2)

%% load Horizontal Displacement case camN_3
% Clean workspace
clear all; close all; clc
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
%implay(vidFrames1_3)
%implay(vidFrames2_3)
%implay(vidFrames3_3)

%% load Horizontal Displacement and Rotation case camN_4
% Clean workspace
clear all; close all; clc
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
```

```
%implay(vidFrames1_4)
%implay(vidFrames2_4)
%implay(vidFrames3_4)

%% constant
Nx = 480;
Ny = 640;
wx = 200;
wy = 200;

%% N = 1 case1

numFrames1 = size(vidFrames1_1,4);
filter = zeros(Nx, Ny);
filter(Nx/4 + wx/4:Nx/2 + wx, Ny/2 - wy/10:Ny/2 + wy/2) = 1;
plotx1 = zeros(1, numFrames1);
ploty1 = zeros(1, numFrames1);

for j = 1:numFrames1
    I = vidFrames1_1(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx1(j) = mean(place1);
    ploty1(j) = mean(place2);
end

%% N = 2 case1
numFrames2 = size(vidFrames2_1,4);
filter = zeros(Nx, Ny);
filter(Nx/4 - wx/10:Nx/2 + wx, Ny/2 - wy/2:Ny/2 + wy/5) = 1;
```

```
plotx2 = zeros(1, numFrames2);
ploty2 = zeros(1, numFrames2);

for j = 1:numFrames2
    I = vidFrames2_1(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx2(j) = mean(place1);
    ploty2(j) = mean(place2);
end

%% N = 3 case1
numFrames3 = size(vidFrames3_1,4);
filter = zeros(Nx, Ny);
filter(Nx/2 - wx/10:Nx/2 + wx/2, Ny/4 + wy/2 - wy/10:Ny/2 + wy - wy/10) = 1;
plotx3 = zeros(1, numFrames3);
ploty3 = zeros(1, numFrames3);

for j = 1:numFrames3
    I = vidFrames3_1(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx3(j) = mean(place1);
    ploty3(j) = mean(place2);
end

%% result of first case
```

```
minlength = min([numFrames1, numFrames2, numFrames3]);

plotx1 = plotx1(1:minlength);

ploty1 = ploty1(1:minlength);

plotx2 = plotx2(1:minlength);

ploty2 = ploty2(1:minlength);

plotx3 = plotx3(1:minlength);

ploty3 = ploty3(1:minlength);


all = [plotx1; ploty1; plotx2; ploty2; plotx3; ploty3];

center = all - mean(all, 2);


[U,S,V] = svd(center / sqrt(minlength) - 1, 'econ');

disp = U' * center;


figure(1)

X_rank1 = S(1,1)*U(:,1)*V(:,1)';

subplot(4,1,1)

plot(X_rank1(1,:),X_rank1(2,:),'r.','MarkerSize',10);

xlabel('x');

ylabel('y');

title("first rank approximation");

sig = diag(S).^2;

subplot(4,1,2)

plot(sig/sum(sig),'ko','Linewidth',2)

ylabel('\sigma');

xlabel('N');

title("energy");

subplot(4,1,3)

plot(1:minlength, center(1,:), 1:minlength, center(2,:))

ylabel("Displacement(px)"); xlabel("Time(seconds)");

title("noise case, displancement along xy plane and z axis");

legend("z", "xy")

subplot(4,1,4)
```

```
plot(1:minlength, disp(1,:))
ylabel("Displacement(px)"); xlabel("Time(seconds)");
title("noise case");
legend("PCA on the second case")

%% N = 1 case2
numFrames1 = size(vidFrames1_2,4);
filter = zeros(Nx, Ny);
filter(Nx/4 - wx/4:Nx/2 + wx, Ny/2 - wy/10:Ny/2 + wy/2) = 1;
plotx1 = zeros(1, numFrames1);
ploty1 = zeros(1, numFrames1);

for j = 1:numFrames1
    I = vidFrames1_2(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx1(j) = mean(place1);
    ploty1(j) = mean(place2);
end

%% N = 2 case2
numFrames2 = size(vidFrames2_2,4);
filter = zeros(Nx, Ny);
filter(Nx/8 - wx/20:Nx - wx/4, Ny/4 + wy/10:Ny/2 + wy/2) = 1;
plotx2 = zeros(1, numFrames2);
ploty2 = zeros(1, numFrames2);

for j = 1:numFrames2
    I = vidFrames2_2(:,:,:,j);
    I = rgb2gray(I);
```

```
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx2(j) = mean(place1);
    ploty2(j) = mean(place2);
end

%% N = 3 case2
numFrames3 = size(vidFrames3_2,4);
filter = zeros(Nx, Ny);
filter(Nx/2 - wx/4:Nx/2 + wx/2, Ny/4 + wy/2:Ny/2 + wy/2 + wy/4) = 1;
plotx3 = zeros(1, numFrames3);
ploty3 = zeros(1, numFrames3);

for j = 1:numFrames3
    I = vidFrames3_2(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx3(j) = mean(place1);
    ploty3(j) = mean(place2);
end

%% result of second case
minlength = min([numFrames1, numFrames2, numFrames3]);
plotx1 = plotx1(1:minlength);
ploty1 = ploty1(1:minlength);
plotx2 = plotx2(1:minlength);
ploty2 = ploty2(1:minlength);
plotx3 = plotx3(1:minlength);
```

```
ploty3 = ploty3(1:minlength);


all = [plotx1; ploty1; plotx2; ploty2; plotx3; ploty3];
center = all - mean(all, 2);


[U,S,V] = svd(center / sqrt(minlength), 'econ');
disp = U' * center;


figure(1)
X_rank1 = S(1,1)*U(:,1)*V(:,1)';
subplot(4,1,1)
plot(X_rank1(1,:),X_rank1(2,:),'r.','MarkerSize',10);
xlabel('x');
ylabel('y');
title("first rank approximation");
sig = diag(S).^2;
subplot(4,1,2)
plot(sig/sum(sig),'ko','Linewidth',2)
ylabel('\sigma');
xlabel('N');
title("energy");
subplot(4,1,3)
plot(1:minlength, center(1,:), 1:minlength, center(2,:))
ylabel("Displacement(px)"); xlabel("Time(seconds)");
title("ideal case, displancement along xy plane and z axis");
legend("z", "xy")
subplot(4,1,4)
plot(1:minlength, disp(1,:))
ylabel("Displacement(px)"); xlabel("Time(seconds)");
title("ideal case");
legend("PC1 on the first case")


%% N = 1 case3
```

```
numFrames1 = size(vidFrames1_3,4);
filter = zeros(Nx, Ny);
filter(Nx/2:Nx - wx/4, Ny/2 - wy/4:Ny/2 + wy/4) = 1;
plotx1 = zeros(1, numFrames1);
ploty1 = zeros(1, numFrames1);

for j = 1:numFrames1
    I = vidFrames1_3(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx1(j) = mean(place1);
    ploty1(j) = mean(place2);
end

%% N = 2 case3
numFrames2 = size(vidFrames2_3,4);
filter = zeros(Nx, Ny);
filter(Nx/3:Nx - wx/2, Ny/4 + wy/4:Ny/2 + wy/4) = 1;
plotx2 = zeros(1, numFrames2);
ploty2 = zeros(1, numFrames2);

for j = 1:numFrames2
    I = vidFrames2_3(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx2(j) = mean(place1);
    ploty2(j) = mean(place2);
```

```
end

%% N = 3 case3
numFrames3 = size(vidFrames3_3,4);
filter = zeros(Nx, Ny);
filter(Nx/2 - wx/4:Nx/2 + wx/2, Ny/4 + wy/4:Ny/2 + wy/2 + wy/4) = 1;
plotx3 = zeros(1, numFrames3);
ploty3 = zeros(1, numFrames3);

for j = 1:numFrames3
    I = vidFrames3_3(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx3(j) = mean(place1);
    ploty3(j) = mean(place2);
end

%% result of third case
minlength = min([numFrames1, numFrames2, numFrames3]);
plotx1 = plotx1(1:minlength);
ploty1 = ploty1(1:minlength);
plotx2 = plotx2(1:minlength);
ploty2 = ploty2(1:minlength);
plotx3 = plotx3(1:minlength);
ploty3 = ploty3(1:minlength);

all = [plotx1; ploty1; plotx2; ploty2; plotx3; ploty3];
center = all - mean(all, 2);

[U,S,V] = svd(center / sqrt(minlength), 'econ');
```

```
disp = U' * center;


figure(1)
X_rank1 = S(1,1)*U(:,1)*V(:,1)';
subplot(4,1,1)
plot(X_rank1(1,:),X_rank1(2,:),'r.','MarkerSize',10);
xlabel('x');
ylabel('y');
title("first rank approximation");
sig = diag(S).^2;
subplot(4,1,2)
plot(sig/sum(sig),'ko','Linewidth',2)
ylabel('\sigma');
xlabel('N');
title("energy");
subplot(4,1,3)
plot(1:minlength, center(1,:), 1:minlength, center(2,:))
ylabel("Displacement(px)"); xlabel("Time(seconds)");
title("horizontal case, displancement along xy plane and z axis");
legend("z", "xy")
subplot(4,1,4)
plot(1:minlength, disp(1,:))
ylabel("Displacement(px)"); xlabel("Time(seconds)");
title("horizontal case");
legend("PC1 on the third case")


%% N = 1 case4
numFrames1 = size(vidFrames1_4,4);
filter = zeros(Nx, Ny);
filter(Nx/2:Nx - wx/5, Ny/2:Ny/2 + wy - wy/4) = 1;
plotx1 = zeros(1, numFrames1);
ploty1 = zeros(1, numFrames1);
```

```
for j = 1:numFrames1
    I = vidFrames1_4(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx1(j) = mean(place1);
    ploty1(j) = mean(place2);
end

%% N = 2 case4
numFrames2 = size(vidFrames2_4,4);
filter = zeros(Nx, Ny);
filter(Nx/5:Nx - wx/2, Ny/4 + wy/4:Ny/2 + wy/2) = 1;
plotx2 = zeros(1, numFrames2);
ploty2 = zeros(1, numFrames2);

for j = 1:numFrames2
    I = vidFrames2_4(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 245);
    plotx2(j) = mean(place1);
    ploty2(j) = mean(place2);
end

%% N = 3 case4
numFrames3 = size(vidFrames3_4,4);
filter = zeros(Nx, Ny);
filter(Nx/4:Nx/2 + wx/5, Ny/2:Ny - wy/2) = 1;
```

```
plotx3 = zeros(1, numFrames3);
ploty3 = zeros(1, numFrames3);

for j = 1:numFrames3
    I = vidFrames3_4(:,:,:,j);
    I = rgb2gray(I);
    I = im2double(I);
    I = I.*filter;
    I = im2uint8(I); % set to unit8 form
    [place1, place2] = find(I > 225);
    plotx3(j) = mean(place1);
    ploty3(j) = mean(place2);
end

%% result of fourth case
minlength = min([numFrames1, numFrames2, numFrames3]);
plotx1 = plotx1(1:minlength);
ploty1 = ploty1(1:minlength);
plotx2 = plotx2(1:minlength);
ploty2 = ploty2(1:minlength);
plotx3 = plotx3(1:minlength);
ploty3 = ploty3(1:minlength);

all = [plotx1; ploty1; plotx2; ploty2; plotx3; ploty3];
center = all - mean(all, 2);

[U,S,V] = svd(center / sqrt(minlength), 'econ');
disp = U' * center;

figure(1)
X_rank1 = S(1,1)*U(:,1)*V(:,1)';
subplot(4,1,1)
plot(X_rank1(1,:),X_rank1(2,:),'r.','MarkerSize',10);
```

```
xlabel('x');

ylabel('y');

title("first rank approximation");

sig = diag(S).^2;

subplot(4,1,2)

plot(sig/sum(sig),'ko','Linewidth',2)

ylabel('\sigma');

xlabel('N');

title("energy");

subplot(4,1,3)

plot(1:minlength, center(1,:), 1:minlength, center(2,:))

ylabel("Displacement(px)"); xlabel("Time(seconds)");

title("horizontal case with rotation, displancement along xy plane and z axis");

legend("z", "xy")

subplot(4,1,4)

plot(1:minlength, disp(1,:))

ylabel("Displacement(px)"); xlabel("Time(seconds)");

title("horizontal case with rotation");

legend("PC1 on the fourth case")
```