AMATH482
HW3 PCA and a Spring-Mass System
Dean Wang

Abstract
Given three cameras located at three different points to record the moving of a spring-mass system. However, there are some noise happened on the camera. We need to solve ideal case, noise case, horizontal displacement case, and horizontal displacement and rotation case. What we are going to do is to analysis the data from three cameras to avoid the infection.

Introduction and Overview
Even though we three videos about the spring-mass system moving from different positions, we want to use principal component analysis to analysis the data. We will apply SVD in this project to get a number of constitutive components. The bucket is only moved along z direction in the ideal case; the video is noised by the shake made by people; the bucket is moved off center and moving in xy direction together with z direction in the horizontal displacement case; the bucket is moved off center and rotated and moving in in xy direction together with z direction in the horizontal displacement and rotation case.

Theoretical Background
We are given:

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

where we know that U and V are unitary matrices that simply lead to rotation and ∑ scales an image as prescribed by the singular values. In addition, U and V are rotational matrix, and ∑ is stretching matrix. By using PAC method, we can decompose A (the prove below) and project the origin graph.
The decomposition:

$$A^TA = \left(U\Sigma V^*\right)^T\left(U\Sigma V^*\right) = V\Sigma^2V^* \tag{2}$$

$$AA^T = \left(U\Sigma V^*\right)\left(U\Sigma V^*\right)^T = U\Sigma^2U^* \tag{3}$$

Furthermore, covariance is also important in the analysis since it will show all the correlation among variables, thus it is very easy to tell which variable is redundant by knowing the statistically dependent value. The covariance function:

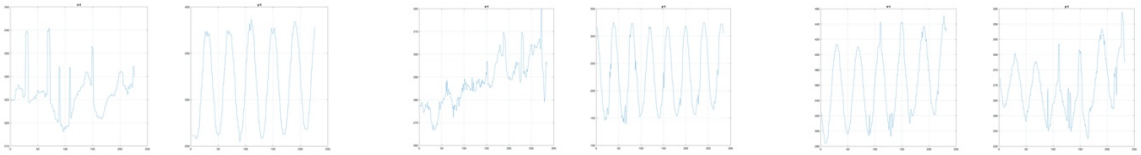
$$C_X = \frac{1}{n-1}XX^T \tag{4}$$

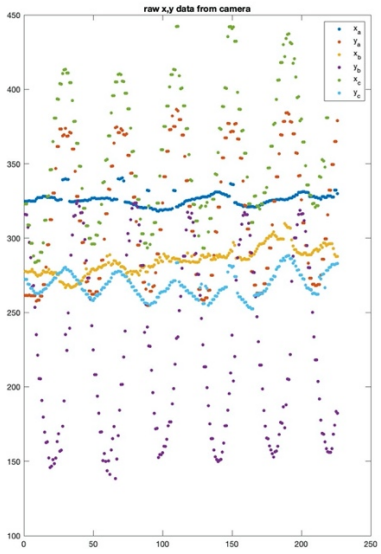
Algorithm Implementation and Development
The purpose of the assignment is to process three sets of videos in four different test settings to obtain the xy coordinates mass in each video. Finally, SVD extract the motion information from the six dimensions. The switch of different test sets the variable called test at the beginning of the code, giving {1,2,3,4} the corresponding video file reading test1, test2, test3, and test4.

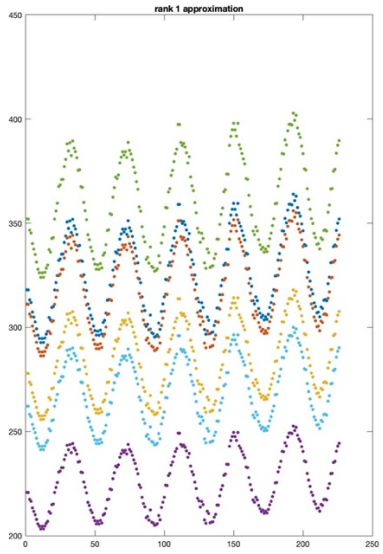
Since all four cases will use the same code, they are in the same logic. First of all, I use for loop to change the different video of 3 camera and set selected region of certain camera. Then I import a function called findmloc.m to read a video message and locate the bucket. In this section, I choose most light area to transfer the grayscale image to binary image where light part is 1 and dark part is 0. Some morphology processing here to gathering highlight area. Then, I plot here for visual illustration and label all the highlight region. Finally, I set find the lowest light area in frame and print the procedure. Moreover, let's go back to the main file. I plot xy and process the data. Since 3 cameras have different recording lengths truncate them to same length, I use the code to solve the problem. In the end I get the four set of graphs.

Computational Result

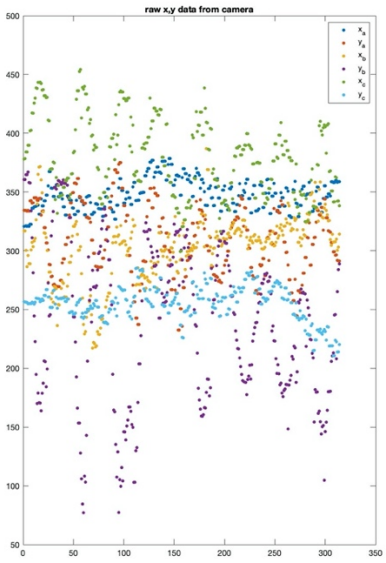
In the case of ideal one, we get the mass location detection from the selected original video from three directions. I print the correlation of x-t and y-t on the graphs and generate the comparison of raw x y data from camera and the rank1 approximation. The graphs from left to right is vidFrames1, vidFrames2, vidFrames3.

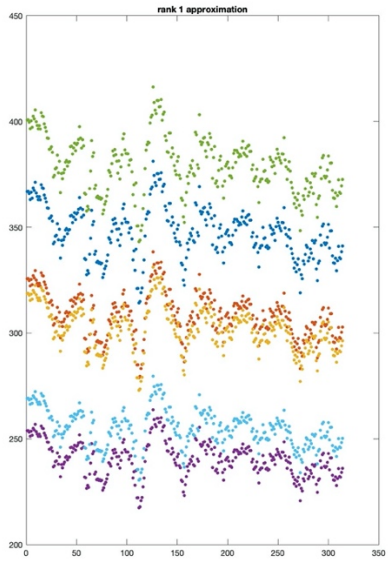




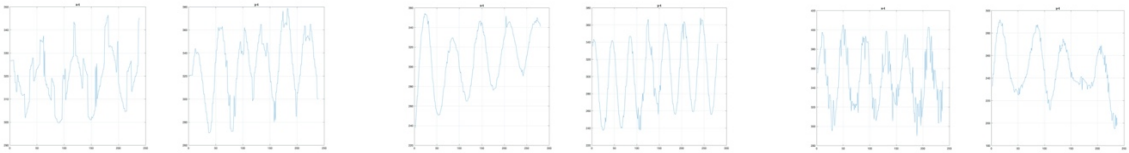


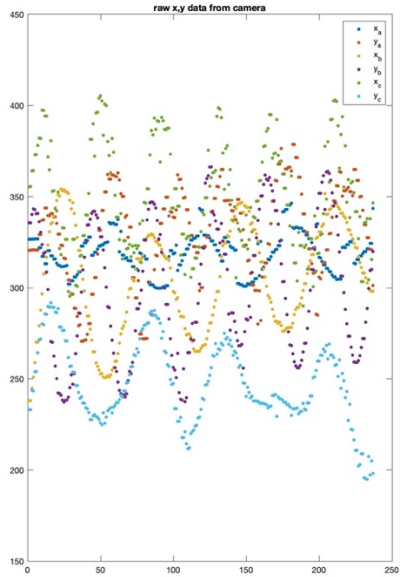
In the case of noise one, we get the mass location detection from the selected original video from three directions. I print the correlation of x-t and y-t on the graphs and generate the comparison of raw x y data from camera and the rank1 approximation.

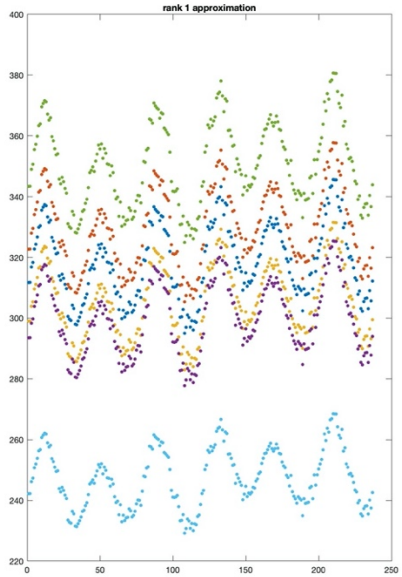




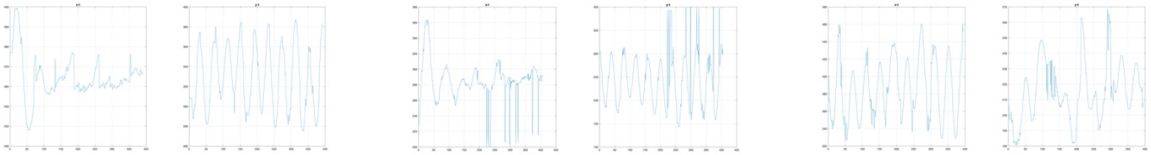
In the case of horizontal displacement one, we get the mass location detection from the selected original video from three directions. I print the correlation of x-t and y-t on the graphs and generate the comparison of raw x y data from camera and the rank1 approximation. The graphs from left to right is vidFrames1, vidFrames2, vidFrames3.

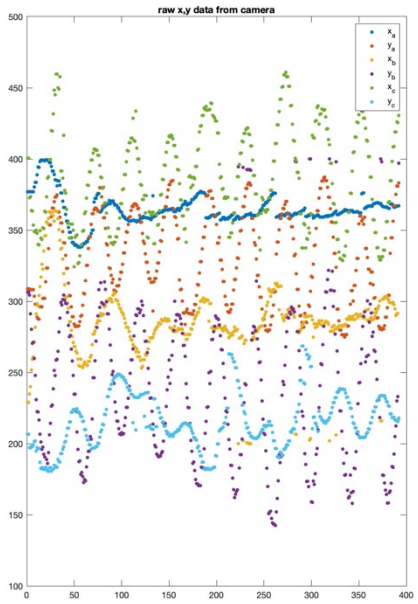


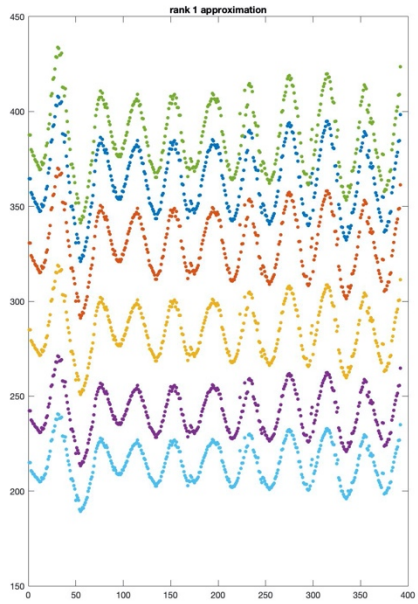




In the case of horizontal displacement and rotation one, we get the mass location detection from the selected original video from three directions. I print the correlation of x-t and y-t on the graphs and generate the comparison of raw x y data from camera and the rank1 approximation. The graphs from left to right is vidFrames1, vidFrames2, vidFrames3.







Summary and Conclusions

After analyzing the mass moving in four cases, I print the graphs of raw data and approximation. It's goes without saying that PCA method will produce an organized oscillation and a better result. With this method, we not only decrease the redundant but also save the storage in the code processing, thus it can help us save the time.

Appendix

amath482_HW3.m

```
clc
clear
close all
testnum=1;
%change the test number here

for cameranum=1:3
%change the different video of 3 camera here

regionselect=zeros(480,640);
if cameranum==1
regionselect(:,280:400)=1;
end
if cameranum==2
regionselect(1:400,200:450)=1;
end
if cameranum==3
regionselect(180:320, 250:640)=1;
end
%setting selected region of certain camera

tol=10;
if testnum==1||testnum==2
tol=4;
end
if testnum==3||testnum==4
tol=3;
end

[x,y]=findmloc(testnum,cameranum,regionselect,tol);
%plot x y
figure('Position', [10 10 1800 600])
subplot(1,2,1)
plot(1:length(y),x)
grid on
title('x-t')
subplot(1,2,2)
plot(1:length(y),y)
grid on
title('y-t')

if cameranum==1
   x_a=x;
   y_a=y;
end
if cameranum==2
   x_b=x;
   y_b=y;
end
if cameranum==3
   x_c=x;
   y_c=y;
end
end
```

```
%%
%data processing
tmin=min([length(x_a),length(x_b),length(x_c)]);
x_a=x_a(1:tmin);
x_b=x_b(1:tmin);
x_c=x_c(1:tmin);
y_a=y_a(1:tmin);
y_b=y_b(1:tmin);
y_c=y_c(1:tmin);
%since 3 cameras have different recording lengths
%truncate them to same length

XY=[x_a;y_a;x_b;y_b;x_c;y_c];
[U,S,V] = svd(XY,'econ');
X_rank1 = S(1,1)*U(:,1)*V(:,1)';
disp(S)
XY_proj=U'*XY;

figure('Position', [100 200 1300 800])
subplot(1,2,1)
for i=1:6
   plot(1:tmin,XY(i,:),'.','MarkerSize',10)
   hold on
end
title('raw x,y data from camera')
legend('x_a','y_a','x_b','y_b','x_c','y_c')
subplot(1,2,2)
for i=1:6
   plot(1:tmin,X_rank1(i,:),'.','MarkerSize',10)
   hold on
end
title('rank 1 approximation')
```

findmloc.m

```
function [x,y]=findmloc(testnum,cameranum,regionselect,tol)


casenum=num2str(testnum);
vi=cameranum;

filename=[{['cam1_',casenum,'.mat']},{['cam2_',casenum,'.mat']},{['cam3_',cas
enum,'.mat']}];
framename=[{['vidFrames1_',casenum,'']},{['vidFrames2_',casenum,'']},{['vidFr
ames3_',casenum,'']}];
load(string(filename(vi)))
name=char(framename(vi));

% eval(['implay(',name,')'])
%%

eval(['numFrames = size(',name,',4)']);
figure('Position', [100 200 1600 800])
t=[];
x=[];
y=[];

for j = 1:1:numFrames

eval(['X = ',char(framename(vi)),'(:,:,:,j);'])

Xg=rgb2gray(X);

% imshow(Xg); drawnow
Xg=Xg.*uint8(regionselect);


        thresh=double(max(max(Xg))-tol)/255;
        %choose most light area
        I=im2bw(Xg,thresh);
        %transfer the grayscle image to binary image where light part is 1
        %and dark part is 0

        se=strel('disk',20);
        se2=strel('disk',40);
        I = imdilate(I,se);
        I = imerode(I,se);
        I = imdilate(I,se);
        I = imdilate(I,se);
        %some morphology processing here to gathering highlight area

        subplot(1,2,1)
        imshow(Xg); drawnow
        title('selected origin video')
        subplot(1,2,2)
        imshow(I); drawnow
        title('selected highlight')
        hold on
```

```
        %plot here for visual illustration

        [l,m] = bwlabel(I);
        %lable all the highlight region
        cxy=regionprops(l,'Centroid');
        if m>0
        xlist=[];
        ylist=[];
        for i=1:m
            xlist=[xlist cxy(i).Centroid(1)];
            ylist=[ylist cxy(i).Centroid(2)];
        end
        end

        index=find(ylist==max(ylist));
        %set find the lowest light area in frame
        selecx=cxy(index).Centroid(1);
        selecy=cxy(index).Centroid(2);
        x=[x selecx];
        y=[y selecy];
        disp([num2str(j),'/',num2str(numFrames),'compeleted'])
        %print precedure
        scatter(selecx,selecy,100,'rx')
end
title(['mass location detection of file:  ',name])
disp(j)
pause(0.001)

end
```