Amath 482 HW 3

Zoheb Siddiqui

February 27, 2019

Abstract

This homework involves taking a video of a can captured from 3 different angles and using PCA to identify the motion in the system.

Introduction

We have a can set in motion. We have three cameras capturing this motion from 3 different locations. The goal is to use PCA to gain information about the motion of the can and learn more about how PCA works.
We extract the location of the can for each frame for each camera and get a matrix , M, representing the location of the can in each frame.
we perform SVD on this Matrix to get it's principle components [U, S, V].
Using the orthogonal matrix U, we find the projection of our original matrix M on U.
We now plot the projection vectors to give us an idea about the motion.

Theoretical Background

We are looking to analyze the motion of a spring system. In the first two cases the movement is only along one axis and thus we have only 1 degree of freedom.
In the next two systems the motion is along two axes and therefore we have 2 degrees of freedom.
Since we are using three cameras and in all 6 degrees of freedom, PCA should reduce the dimentionality of the system.
To find the dependence between the data we collect we must create the covariance matrix.
Suppose the data matrix is M, the covarience matrix $C = \frac{1}{n-1} MM^T$.
Higher number of non-zero elements in the covarience matrix indicate that the data is dependant and thus highly redundant. Since we are using three cameras to monitor the same system, we expect this to happen.
Our next goal is to remove this redundancy by diagonalizing the covarience matrix. This can be accomplished in two ways.

- We can perform eigenvalue decomposition on C to obtain [V_1, λ].
- We can perform SVD on M to obtain [U, S, V_2].

The Eigenvalue decomposition and SVD are related such that $U = V_1$ and $S = \sqrt{\lambda}$.
[U,S,V_2] are matrices such that U, V_2^T are orthogonal matrices that are responsible for rotation operations whereas S is a diagonal matrix consisting of the singular values of M. It is responsible for stretching operations.
Also, note that $M = USV^T$.
Now, we project M onto the U matrix as $projection = U^T M$.
In the first 2 cases we shall ideally only require the projection vector corresponding to the largest singular value as there is only 1 degree of freedom.
Similarly, in the next 2 cases we shall require 2 projection vectors corresponding to the two largest singular values.
The projection vectors chosen are then super imposed on one another till we have 1 vector which is the sum of all desired projection vectors.
Plotting this vector will yield us the motion of the object as analyzed by PCA.

Algorithm

First we make each video uniform by making number of frames equal.

For each frame:
    For all videos:
        Make each frame RGB
        Compress each frame from 460 x 640 to 120 x 160.

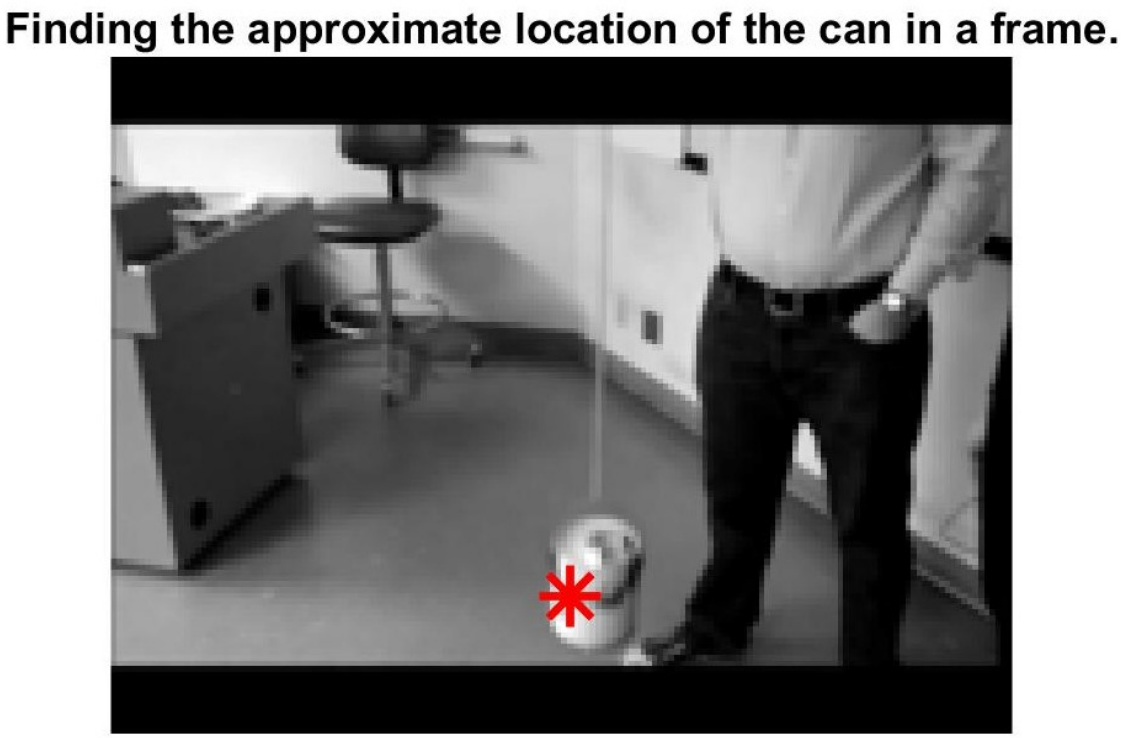
Create an empty matrix M $\subseteq R^{6,frames}$ to store the location of the can.
For each video: For each frame:
    For each video:
        Find the (x, y) location of the center of the can by finding average brightest point.
        Add the location to M.
De mean M by subtraction the mean of each row from the components of that row.

Perform SVD on M to obtain [U,S,V]. Note that we only need 6 singular values.

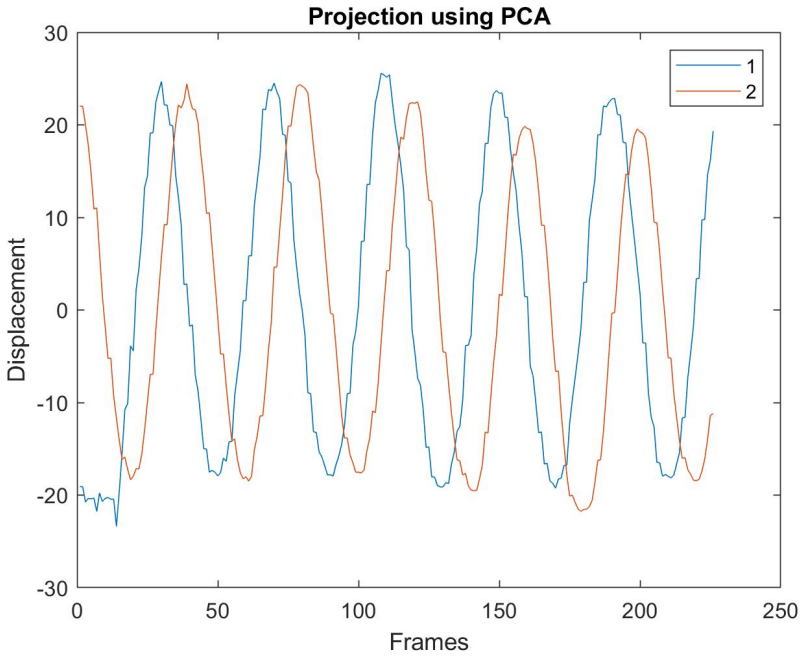
Project the matrix M on to U as $U^T M$. The projection will be a $R^{6,frames}$ matrix.
Look at diag(S).
If the first x values of S correspond to more than 80% of the information:
    Plot the x projection corresponding to the singualr values.

Computational Results

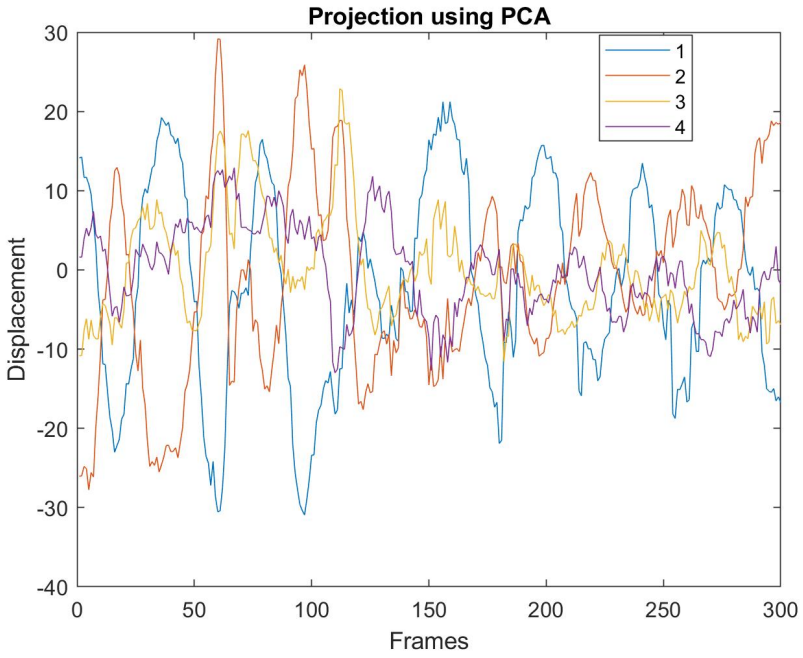
First we find the location of the can in each frame. The image with the location looks like:



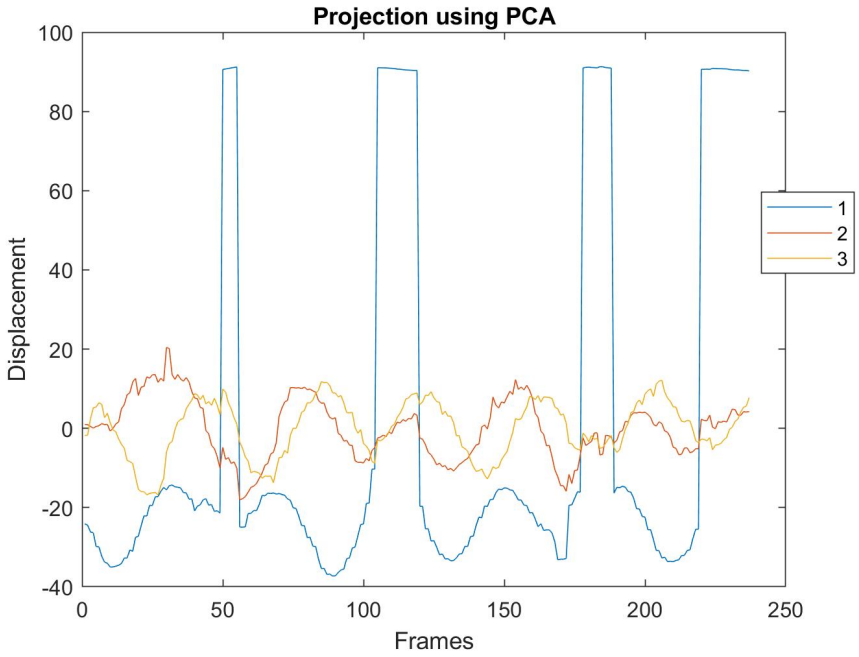
Part 1: The singular values are: [231.6368, 222.1034, 31.7308, 18.0200, 14.5307, 8.8178]. The first 2 values account for more than 85% of the motion. Thus, we only plot the 1st 2 projection vectors.
The plot for the projection looks like:



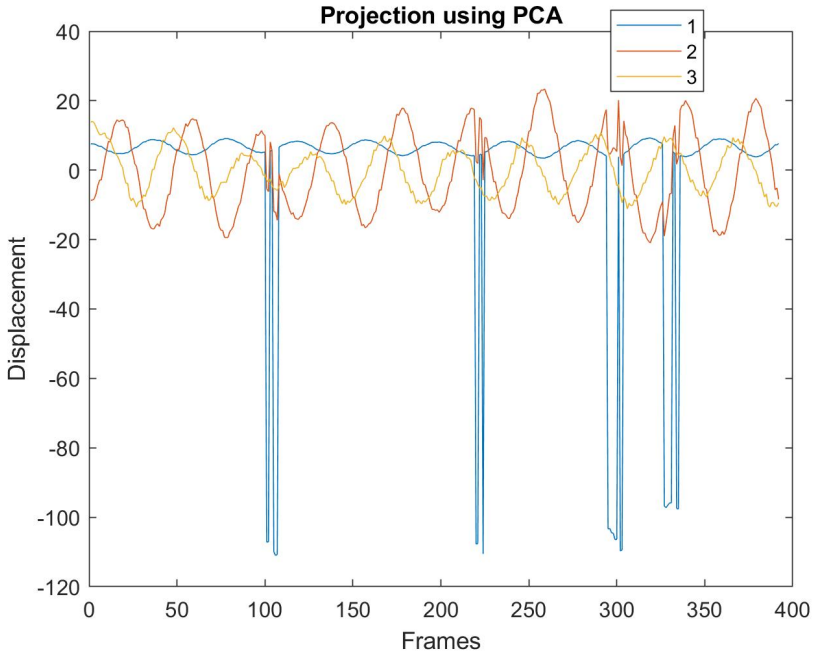
Part 2: The singular values are: [338.8041, 252.8970, 115.3860, 101.0712, 58.6748, 50.7360]. The first 4 values account for more than 88% of the motion. Thus, we only plot the 1st 4 projection vectors.
The plot for the projection looks like:



Part 3: The singular values are: [728.8733, 117.9349, 109.7198, 65.0510, 37.3166, 19.0856]. The first 3 values account for more than 88% of the motion. Thus, we only plot the 1st 3 projection vectors.
The plot for the projection looks like:



Part 4: The singular values are: [516.6206, 233.1386, 124.6126, 85.2393, 27.7876, 20.2342]. The first 3 values account for more than 86% of the motion. Thus, we only plot the 1st 3 projection vectors.
The plot for the projection looks like:



Conclusion

Part 1: Note that two singular values account for nearly all of the motion. Theoretically only one of them should have however the way we extract the location of the can is somewhat noisy.
The graph however does correspond to simple harmonic motion which is what we expected it to.

Part 2: In this case we have excessive noise due to camera shake added to the noisy data extraction process. Thus, more singular values are required to account for nearly all of the motion.
Even after using 4 singular values and their corresponding projection vectors we can see that the graph produced is fairly noisy. It does show traits of simple harmonic motion however it is tough to make out.
This leads me to conclude that PCA does not perform very well under the influence of excessive noise.

Part 3 and 4: In this case theoretically we would require 2 singular values however we end up using 3 to get accurate results. From the graph we can see that while there is simple harmonic motion, there is another form of motion mixed into it as well. However simply by looking at the graph we cannot say in which axis the motion occurs as we lose that information when we perform the projection.
In **part 4** specifically, we fail to identify rotation as we are using only one point to mark the can. If we were using multiple points and keeping track of the amount of time it takes for them to resurface, we would have identified the rotation motion as well.

Appendices

Matlab Functions Used

load(): Loads an image into matlab.

zeros(): creates a matrix filled with zeros.

imresize(): resizes an image to the desired pixel range.

rgb2gray(): converts an image to gray scale.

double(): converts a number to the double data type.

uint8(): converts a number to the integer data type.

imshow(): used to plot an image given pixel matrix of uint8 type.

mean(): Finds the mean of a matrix or a vector.

SVD(): Given a matrix, M, it breaks it down into 3 components [U,S,V] s.t $USV^T = M$

Code

Part 1

```
clear all; close all; clc;

%loading data

frames_1 = load('Data/cam1_1.mat');
frames_1 = frames_1.('vidFrames1_1');

frames_2 = load('Data/cam2_1.mat');
frames_2 = frames_2.('vidFrames2_1');

frames_3 = load('Data/cam3_1.mat');
frames_3 = frames_3.('vidFrames3_1');

%making number of frames uniform.
min_frames = 226;

frames_1 = frames_1(1:480,1:640, 1:3, 1:min_frames);
frames_2 = frames_2(1:480,1:640, 1:3, 1:min_frames);
frames_3 = frames_3(1:480,1:640, 1:3, 1:min_frames);

%%
%converting to rgb
%extracting the location of the can
M = zeros(6,min_frames);

for i = 1:min_frames

    frame_1_to_add = imresize(rgb2gray(frames_1(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_2_to_add = imresize(rgb2gray(frames_2(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_3_to_add = imresize(rgb2gray(frames_3(1:480,1:640, 1:3, i)...
        ), [120,160]);

    frame_1_to_add = double(frame_1_to_add);
    frame_2_to_add = double(frame_2_to_add);
    frame_3_to_add = double(frame_3_to_add);
```

```
x_vals_1 = 0;
y_vals_1 = 0;
count_1 = 0;

x_vals_2 = 0;
y_vals_2 = 0;
count_2 = 0;

x_vals_3 = 0;
y_vals_3 = 0;
count_3 = 0;

for j = 1:120
    for k = 1:160
        if frame_1_to_add(j,k) >= 240 && ...
                k >= 70 && k <= 100
            x_vals_1 = x_vals_1 + j;
            y_vals_1 = y_vals_1 + k;
            count_1 = count_1 + 1;
        end
        if frame_2_to_add(j,k) >= 240 && ...
                k >= 60 && k <= 90
            x_vals_2 = x_vals_2 + j;
            y_vals_2 = y_vals_2 + k;
            count_2 = count_2 + 1;
        end
        if frame_3_to_add(j,k) >= 240 && ...
                j >= 60 && j <= 80 && k >= 60 && k <= 131
            x_vals_3 = x_vals_3 + j;
            y_vals_3 = y_vals_3 + k;
            count_3 = count_3 + 1;
        end
    end
end

x_vals_1 = x_vals_1/count_1;
y_vals_1 = y_vals_1/count_1;

x_vals_2 = x_vals_2/count_2;
y_vals_2 = y_vals_2/count_2;

x_vals_3 = x_vals_3/count_3;
y_vals_3 = y_vals_3/count_3;

M(1,i) = x_vals_1;
```

```
    M(2,i) = y_vals_1;

    M(3,i) = x_vals_2;
    M(4,i) = y_vals_2;

    M(5,i) = x_vals_3;
    M(6,i) = y_vals_3;
end

figure(1)
imshow(uint8(frame_1_to_add));
hold on
plot(y_vals_1, x_vals_1, 'r*', 'LineWidth', ...
    2, 'MarkerSize', 15);
title('Finding the approximate location of the can in a frame.')
drawnow
%%
%De meaning data
mean_data = zeros(6);
for i = 1:6
    mean_data(i) = mean(M(i,:));
    M(i,:) = M(i,:) - mean(M(i,:));

end
%%
%SVD
[U,S,V] = svd(M,'econ');
disp(S);
%%
%projection
proj = U.'*M;
%%
%plotting projection
figure(2)
plot(proj(1,:) + proj(2, :))
title('Projection using PCA')
xlabel('Frames')
ylabel('Displacement')
hold on
```

Part 2

```
clear all; close all; clc;

%loading data
```

```
frames_1 = load('Data/cam1_2.mat');
frames_1 = frames_1.('vidFrames1_2');

frames_2 = load('Data/cam2_2.mat');
frames_2 = frames_2.('vidFrames2_2');

frames_3 = load('Data/cam3_2.mat');
frames_3 = frames_3.('vidFrames3_2');

%%
%making frames uniform
min_frames = 314;

frames_1 = frames_1(1:480,1:640, 1:3, 1:min_frames);
frames_2 = frames_2(1:480,1:640, 1:3, 1:min_frames);
frames_3 = frames_3(1:480,1:640, 1:3, 1:min_frames);

%%
%converting to rgb
%extracting the location of the can
M = zeros(6,min_frames);

for i = 1:min_frames

    frame_1_to_add = imresize(rgb2gray(frames_1(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_2_to_add = imresize(rgb2gray(frames_2(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_3_to_add = imresize(rgb2gray(frames_3(1:480,1:640, 1:3, i)...
        ), [120,160]);

    frame_1_to_add = double(frame_1_to_add);
    frame_2_to_add = double(frame_2_to_add);
    frame_3_to_add = double(frame_3_to_add);

    x_vals_1 = 0;
    y_vals_1 = 0;
    count_1 = 0;

    x_vals_2 = 0;
    y_vals_2 = 0;
    count_2 = 0;

    x_vals_3 = 0;
```

```
    y_vals_3 = 0;
    count_3 = 0;

    for j = 1:120
        for k = 1:160
            if frame_1_to_add(j,k) >= 240 && ...
                    k >= 70 && k <= 100
                x_vals_1 = x_vals_1 + j;
                y_vals_1 = y_vals_1 + k;
                count_1 = count_1 + 1;
            end
            if frame_2_to_add(j,k) >= 240 && ...
                    k >= 60 && k <= 90
                x_vals_2 = x_vals_2 + j;
                y_vals_2 = y_vals_2 + k;
                count_2 = count_2 + 1;
            end
            if frame_3_to_add(j,k) >= 240 && ...
                    j >= 60 && j <= 80 && k >= 60 && k <= 131
                x_vals_3 = x_vals_3 + j;
                y_vals_3 = y_vals_3 + k;
                count_3 = count_3 + 1;
            end
        end
    end

    x_vals_1 = x_vals_1/count_1;
    y_vals_1 = y_vals_1/count_1;

    x_vals_2 = x_vals_2/count_2;
    y_vals_2 = y_vals_2/count_2;

    x_vals_3 = x_vals_3/count_3;
    y_vals_3 = y_vals_3/count_3;

    M(1,i) = x_vals_1;
    M(2,i) = y_vals_1;

    M(3,i) = x_vals_2;
    M(4,i) = y_vals_2;

    M(5,i) = x_vals_3;
    M(6,i) = y_vals_3;
end
%%
```

```
%De meaning data
M(isnan(M))=0;
mean_data = zeros(6);
for i = 1:6
    mean_data(i) = mean(M(i,:));
    M(i,:) = M(i,:) - mean(M(i,:));

end
%%
%SVD
[U,S,V] = svd(M,'econ');
disp(S);
%%
%projection
proj = U.'*M;
%%
%plotting projection
figure(1)
plot(proj(1,:) + proj(2,:) + proj(3,:) + proj(4,:))
title('Projection using PCA')
xlabel('Frames')
ylabel('Displacement')
hold on
```

Part 3

```
clear all; close all; clc;

%loading data

frames_1 = load('Data/cam1_3.mat');
frames_1 = frames_1.('vidFrames1_3');

frames_2 = load('Data/cam2_3.mat');
frames_2 = frames_2.('vidFrames2_3');

frames_3 = load('Data/cam3_3.mat');
frames_3 = frames_3.('vidFrames3_3');
%%
min_frames = 237;
frames_1 = frames_1(1:480,1:640, 1:3, 1:min_frames);
frames_2 = frames_2(1:480,1:640, 1:3, 1:min_frames);
frames_3 = frames_3(1:480,1:640, 1:3, 1:min_frames);

%%
```

```
%converting to rgb
%extracting the location of the can
M = zeros(6,min_frames);

for i = 1:min_frames

    frame_1_to_add = imresize(rgb2gray(frames_1(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_2_to_add = imresize(rgb2gray(frames_2(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_3_to_add = imresize(rgb2gray(frames_3(1:480,1:640, 1:3, i)...
        ), [120,160]);

    frame_1_to_add = double(frame_1_to_add);
    frame_2_to_add = double(frame_2_to_add);
    frame_3_to_add = double(frame_3_to_add);

    x_vals_1 = 0;
    y_vals_1 = 0;
    count_1 = 0;

    x_vals_2 = 0;
    y_vals_2 = 0;
    count_2 = 0;

    x_vals_3 = 0;
    y_vals_3 = 0;
    count_3 = 0;

    for j = 1:120
        for k = 1:160
            if frame_1_to_add(j,k) >= 240 ...
                    && k >= 70 && k <= 100
                x_vals_1 = x_vals_1 + j;
                y_vals_1 = y_vals_1 + k;
                count_1 = count_1 + 1;
            end
            if frame_2_to_add(j,k) >= 240 ...
                    && k >= 60 && k <= 90
                x_vals_2 = x_vals_2 + j;
                y_vals_2 = y_vals_2 + k;
                count_2 = count_2 + 1;
            end
            if frame_3_to_add(j,k) >= 240 ...
                    && j >= 60 && j <= 80 && k >= 60 && k <= 131
```

```
                x_vals_3 = x_vals_3 + j;
                y_vals_3 = y_vals_3 + k;
                count_3 = count_3 + 1;
            end
        end
    end

    x_vals_1 = x_vals_1/count_1;
    y_vals_1 = y_vals_1/count_1;

    x_vals_2 = x_vals_2/count_2;
    y_vals_2 = y_vals_2/count_2;

    x_vals_3 = x_vals_3/count_3;
    y_vals_3 = y_vals_3/count_3;

    M(1,i) = x_vals_1;
    M(2,i) = y_vals_1;

    M(3,i) = x_vals_2;
    M(4,i) = y_vals_2;

    M(5,i) = x_vals_3;
    M(6,i) = y_vals_3;

end
%%
%De meaning data
mean_data = zeros(6);
M(isnan(M))=0;
for i = 1:6
    mean_data(i) = mean(M(i,:));
    M(i,:) = M(i,:) - mean(M(i,:));

end
%%
%SVD
M(isnan(M))=0;
[U,S,V] = svd(M,'econ');
disp(S);
%%
%projection
proj = U.'*M;
%%
%plotting projection
```

```
figure(1)
plot(proj(1,:) + proj(2, :) + proj(3, :))
title('Projection using PCA')
xlabel('Frames')
ylabel('Displacement')
hold on
```

Part 4

```
clear all; close all; clc;

%loading data

frames_1 = load('Data/cam1_4.mat');
frames_1 = frames_1.('vidFrames1_4');

frames_2 = load('Data/cam2_4.mat');
frames_2 = frames_2.('vidFrames2_4');

frames_3 = load('Data/cam3_4.mat');
frames_3 = frames_3.('vidFrames3_4');
%%
min_frames = min(length(frames_3(1,1,1,:)),...
    min(length(frames_1(1,1,1,:)), length(frames_2(1,1,1,:))));
frames_1 = frames_1(1:480,1:640, 1:3, 1:min_frames);
frames_2 = frames_2(1:480,1:640, 1:3, 1:min_frames);
frames_3 = frames_3(1:480,1:640, 1:3, 1:min_frames);

%%
%converting to rgb
%extracting the location of the can
M = zeros(6,min_frames);

for i = 1:min_frames

    frame_1_to_add = imresize(rgb2gray(frames_1(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_2_to_add = imresize(rgb2gray(frames_2(1:480,1:640, 1:3, i)...
        ), [120,160]);
    frame_3_to_add = imresize(rgb2gray(frames_3(1:480,1:640, 1:3, i)...
        ), [120,160]);

    frame_1_to_add = double(frame_1_to_add);
    frame_2_to_add = double(frame_2_to_add);
    frame_3_to_add = double(frame_3_to_add);
```

```
x_vals_1 = 0;
y_vals_1 = 0;
count_1 = 0;

x_vals_2 = 0;
y_vals_2 = 0;
count_2 = 0;

x_vals_3 = 0;
y_vals_3 = 0;
count_3 = 0;

for j = 1:120
    for k = 1:160
        if frame_1_to_add(j,k) >= 240 && k >= 70 && k <= 100
            x_vals_1 = x_vals_1 + j;
            y_vals_1 = y_vals_1 + k;
            count_1 = count_1 + 1;
        end
        if frame_2_to_add(j,k) >= 240 && k >= 60 && k <= 90
            x_vals_2 = x_vals_2 + j;
            y_vals_2 = y_vals_2 + k;
            count_2 = count_2 + 1;
        end
        if frame_3_to_add(j,k) >= 240 && j >= 60 ...
                && j <= 80 && k >= 60 && k <= 131
            x_vals_3 = x_vals_3 + j;
            y_vals_3 = y_vals_3 + k;
            count_3 = count_3 + 1;
        end
    end
end

x_vals_1 = x_vals_1/count_1;
y_vals_1 = y_vals_1/count_1;

x_vals_2 = x_vals_2/count_2;
y_vals_2 = y_vals_2/count_2;

x_vals_3 = x_vals_3/count_3;
y_vals_3 = y_vals_3/count_3;

M(1,i) = x_vals_1;
M(2,i) = y_vals_1;
```

```
    M(3,i) = x_vals_2;
    M(4,i) = y_vals_2;

    M(5,i) = x_vals_3;
    M(6,i) = y_vals_3;

end
%%
%De meaning data
M(isnan(M))=0;
mean_data = zeros(6);
for i = 1:6
    mean_data(i) = mean(M(i,:));
    M(i,:) = M(i,:) - mean(M(i,:));

end
%%
%SVD
M(isnan(M))=0;
[U,S,V] = svd(M,'econ');
disp(S);
%%
%projection
proj = U.'*M;
%%
%plotting projection
figure(1)
plot(proj(1,:) + proj(2, :) + proj(3,:))
title('Projection using PCA')
xlabel('Frames')
ylabel('Displacement')
hold on
```