AMATH 482 Homework 3

Biyao Li

February 20, 2020

Abstract

In this assignment, we are given 4 sets of three video data where each set describes a oscillatory motion of a paint can. The three videos in each data set capture the motion using three cameras at different locations. The goal of this assignment is to use Principal Component Analysis(PCA) on each of the data set to analyze the connection between the principal components and the motion of the can. Moreover, we will compare the results between different data set to see the how different oscillatory motion impact the principal components.

1 Introduction and Overview

In order to analyze the oscillatory motion of the paint can, we must capture the locations of the paint can in each frame of the video. Since each frame is a 2-D plane, we can store the estimated location as a set of x, y coordinates such that for each of the 4 data set, we will have a 6 by number of frame matrix that captures the location of the paint can from three cameras:

$$\begin{bmatrix} x_{cam1} \\ y_{cam1} \\ x_{cam2} \\ y_{cam2} \\ x_{cam3} \\ y_{cam3} \end{bmatrix} \tag{1}$$

Recall that from a simply spring mass system, we will see a one dimensional motion but in this case, we have a 6 dimensional data that describe a one dimensional motion which means that some of the data are redundant. We can remove such redundancy by projecting our data in X onto the principal components such that we will get a new set of coordinates that will tell us more about the motion of the paint can. If we compute the covariance matrix of the new coordinates, we will get a diagonal matrix such that the variables are uncorrelated which implies that there are no redundancy. Furthermore, we can use the principal components to approximate the motion of the paint can and analyze which of principal components are significant to describe the general motion of the paint can.

2 Theoretical Background

2.1 Singular Value Decomposition(SVD)

For any matrix $A \in C^{m\times n}$, we can apply the Singular Value Decomposition to A such that

$$A = U\Sigma V^* \tag{2}$$

Where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary matrices, and $\Sigma \in \mathbb{R}^{m\times n}$ diagonal matrix. The columns of U are the left singular vectors of A and the columns of V are the right singular vectors of A. The values on the diagonal of Σ are the singular values of A and if we let σ_1 be the first diagonal value and so on, the singular values of A are always ordered from the largest to smallest such that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$. Moreover, since U and V are unitary matrix, it is true that $UU^* = I$ and $VV^* = I$ where I is the identity matrix.

2.2 Energy in each approximation

Recall that the Σ matrix from the svd of A contains m singular values on the diagonal. Then we define the energy contained in the rank-N approximation as:

$$energy = \frac{\sigma_1^2 + \sigma_2^2 + ... + \sigma_n^2}{\sigma_1^2 + \sigma_2^2 + ... + \sigma_m^2} \tag{3}$$

In the case of this assignment, the energies describe the significant direction of the paint can's movement. For example, if we have a paint can that is perfectly moving up and down, we should see the energy of the first two approximations should contain all of the energies in the system. The reason that it is not the first approximation that contains all of the energy is because spring-mass system is a second ordered differential equation problem which has two solutions such that we must use the rank-2 approximation to describe the up down motion.

2.3 Covariance Matrix

For any matrix X that contains the data for a set of variables like the matrix in equation 1, we can compute the covariance matrix of X by

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

Where n is the number of variables. The covariance matrix computes the variance and covariance between each row of X. The non-diagonal values of C_X at the i-th row and j-th column represent the covariance between the i-th and j-th rows of X. A larger covariance means that the data of two rows are highly correlated which implied redundancy whereas a covariance of zero means two rows are statistically independent. For the diagonal value on the i-th row of the covariance matrix, it is the variance of the data on the i-th row of X. Recall that if we removed the redundency from the data, the covariance matrix should be a diagonal matrix which makes sense since all covariances would be zero which implies that the rows of the data are uncorrelated.

Furthermore, we can diagonalize the covariance matrix as

$$C_X = V \Lambda V^{-1} \tag{5}$$

where V contains the eigenvectors which are the principal component directions and Λ contains the eigenvalues on its diagonal.

2.4 Principal Component Analysis(PCA)

With the data matrix X, let $A = \frac{1}{\sqrt{n-1}} X$ such that the covariance matrix of X will equal to

$$C_X = A A^T \tag{6}$$

Recall if we apply SVD on A, $A = U \Sigma V^*$ such that

$$C_X = U \Sigma V^* V \Sigma U^T \tag{7}$$

By the unitary property of V and diagonal property of Σ, the covariance matrix can be simplified as

$$C_X = U \Sigma^2 U^T \tag{8}$$

Furthermore, based on equation 4, the eigenvectors in U is the principal component direction of the data such that the rows of $U^T X$ are the principal components. Then let $Y = U^T X$ and if we compute the covariance matrix of Y, we will get the following:

$$C_Y = \frac{1}{n-1} Y Y^T = \frac{1}{n-1} U^T X X^T U = U^T A A^T U = U^T U \Sigma V^* V \Sigma U^T U = \Sigma^2 \tag{9}$$

This means that the covariance matrix of Y is a diagonal matrix which implies that project data have no redundancy which is exactly what we want. Then we can use the principal components to analyze the motion of the paint can and this is the entire process of the PCA.

3 Algorithm Implementation and Development

3.1 Loading and visualizing the video data

For each of the 4 datasets, we have three **.mat** files each representing the video data captured by the three cameras. By using the **load** command on each .**mat** file, we will get each frame of the video stack up together as a matrix. If we iterate through the matrix, we will be able to access each frame of the video which is a 480 by 640 matrix. Furthermore, we can call the **imshow** command on the matrix of each frame which will allow us to visualize the video in MATLAB. Lines 4 to 12 of the code in appendix B is a example of the above procedure and these lines of code are used whenever we need to iterate through each frame of the video.

3.2 Locating the paint can

In order to perform PCA, we first need to store the location of the paint can from three cameras in the format of the matrix in equation 1. If we take a look at any video composed by the given data, we can see that there is a white light on top of the paint can. This is helpful for locating the paint can since if we transform each frame of the video into a grayscale image, the maximum values in the matrices of grayscale images are the locations of the white light which provide estimated locations of the paint can. We treat the location of the paint can as x,y values which can be found by using **max** and **ind2sub** command. A potential problem we might encounter is that there might be other object in the video that is also emitting white light such that the maximum value might not given us the location of the white light on top of the paint can. This problem can be resolved by the following procedure:

- For the first frame, apply a Shannon filter so that the image will appear black except for the part that is along the motion of the paint can.
- After we collect the location of the paint can in the first frame, we will apply a new Shannon filter to the next frame so that we only keep the pixels that is around the location of the paint can in the previous frame. Repeat this step until we went through every frame of the video.

The lines of code of this part for each of the 4 data set are:

- First data set(ideal case): 17 - 67
- Second data set(Noisy case): 98 - 152
- Third data set(Horizontal displacement): 194 - 243
- Fourth data set(Horizontal displacement and rotation): 284 - 333

From now on, I will refer each data set to the case that they are describing.

3.3 SVD

For each of the 4 data set, after we collected the locations of the paint can from the video frames of the three cameras, we will have three 2 by the number of frames matrices. Since the video recorded from the cameras have different number of frames, we need to trim the matrices such that they have the same number of columns. This will allow us to stack the three matrices into a matrix like X in equation 1. After we subtract the mean from each row of the data matrix, we can define matrix A as $\frac{1}{\sqrt{n-1}}X$ and we will apply SVD on A by using the **svd** command. Note that in MATLAB, the V matrix returned from the **svd** command isn't the V^* in equation 2 such that to get back to A, we need to take the transpose of V matrix. This part of the code in Appendix B are 68 to 78 for the ideal case, 153 to 165 for the noisy case, 245 to 256 for horizontal displacement case, and 335 to 346 for the horizontal displacement and rotation case.

3.4 Energy plots and projections

Following how we define the energy at each approximation in section 2.2, we can plot the energies contained in each approximation which can be used to visualize which rank approximations captures the majority of the energies in the system. This part of the code in Appendix B are 80 to 85 for the ideal case, 167 to 172 for the noisy case, 258 to 263 for horizontal displacement case, and 348 to 353 for the horizontal displacement and rotation case.

Moreover, the energy plot can tell us which of the principal components are significant. Suppose if the energy plot shows that the first two approximations contains all of the energy, then only the first two principal components are significant. This is because the approximations that contains no energy imply that the corresponding principal components are irrelevant in describing the general motion of the paint can.

Recall from section 2.4, by projecting the location matrix of the paint can onto the principal component direction which is the transpose of U matrix(from the svd of A), we will get a matrix of principal components. From the energy plot, we will know which principal components are significant and we can plot these principal components to visualize the motion of the paint can described by the location data. This part of the code in Appendix B are 87 to 95 for the ideal case, 173 to 190 for the noisy case, 264 to 280 for horizontal displacement case, and 355 to 370 for the horizontal displacement and rotation case.

4 Computational Results

4.1 First data set - ideal case

For this case, the paint can is constantly moving up and down without any significant motion to the right and left and this is well supported by the plots in figure 1. As we see in the energy plot, the energy in the system is captured by the first two approximations which means that there is only one significant motion in the system. Recall that the spring mass system is a second order differential equation problem which have a general solution of $x(t) = c_1 cos(wt) + c_2 sin(wt)$. This implies that we need to have a cosine and sine component to completely describe the vertical movement of the paint can. This is exactly what we have as the first principal component display a sine plot and the second principal component display a cosine plot. Moreover, we can see that the first principal component is at its maximum whenever the second principal component is at its minimum which implies the second principal component is the derivative of the first principal component. This make sense since the derivative of sine is cosine.

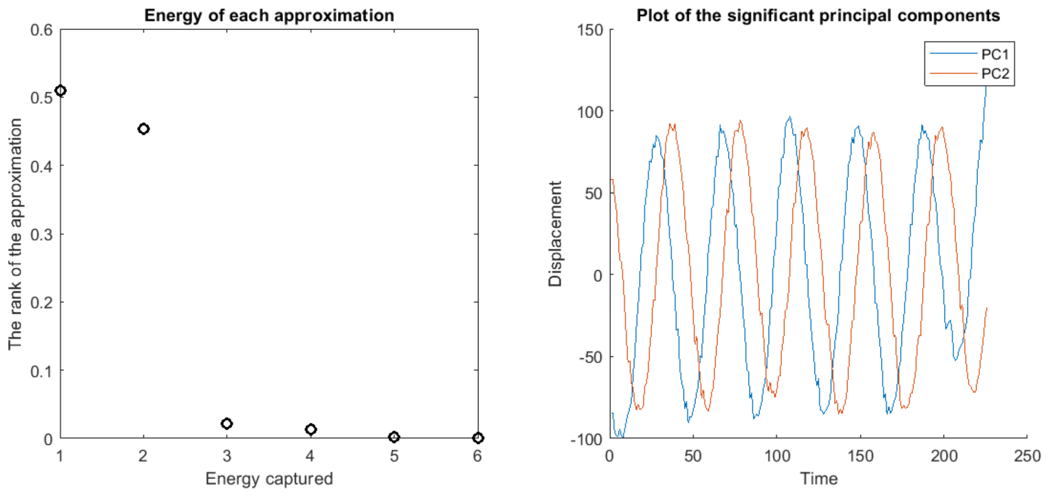

Figure 1: Energy plot and the plot of significant principal components for case 1

4.2 Second data set - noisy case

In this case, we introduce camera shakes into the video such that we should expect some noise in the principal components for the vertical motion. Comparing to previous case, we see that the energy level of the third and fourth approximation get larger so I decided to plot the corresponding principal components to see if it can give us any information. For plot for the first two principal components in figure 2, we can still see the cosine and sine wave that describe the vertical motion but there are a lot of noise in the second principal component. Recall that I extract the coordinate of the paint can with the help of the paint can location from the previous frame. This is problematic when there is shaking in the video since the location of the paint could potential move far apart from the previous location but this explains why the plots of the first two principal components looks really noisy comparing to the near perfect plots for the first case. For plot of the third and fourth principal components in figure 2, it seems like everything appears to be noise which makes sense since there is only vertical motion of the paint can in the first two cases.

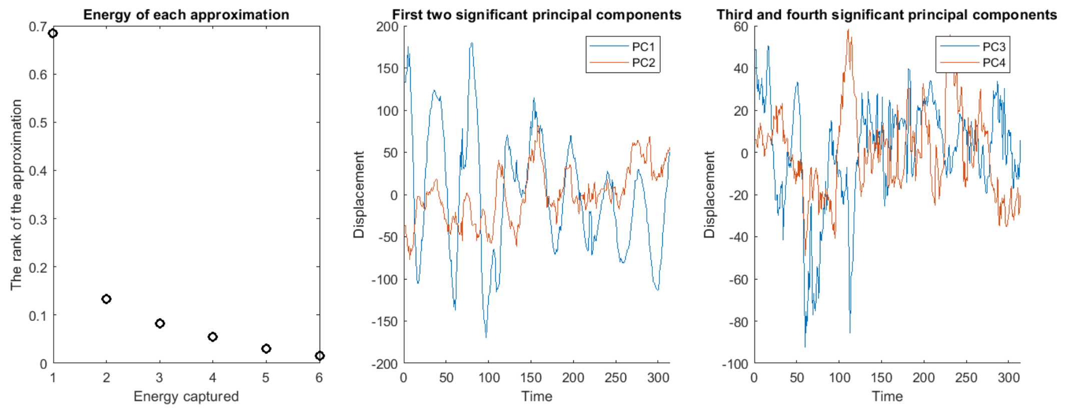

Figure 2: Energy plot and the plot of significant principal components for case 2

4.3 Third data set - horizontal displacement

In this case, we released the paint can off-centre such that the paint will have both horizontal and vertical displacement. This is supported by the energy plot in figure 3 where the energies in the third approximation become much more significant comparing to previous cases. The plot of the first two principal components in figure 3 looks similar to what we have for the first case and we start to see some periodic patterns in the plot of the third principal component in figure 3 which is caused by the horizontal motion that was added.

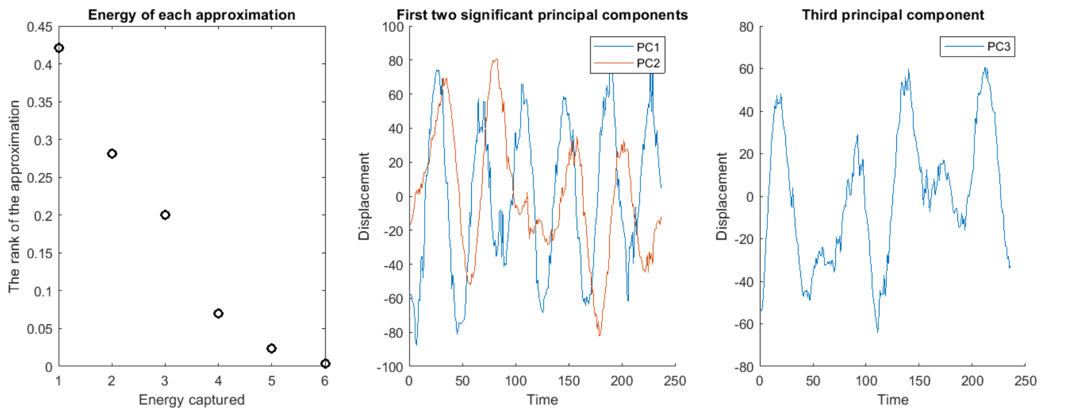

Figure 3: Energy plot and the plot of significant principal components for case 3

4.4 Fourth data set - horizontal displacement and rotation

In this case, we have rotation of the paint can with vertical and horizontal motion. From the energy plot in figure 4, we see that the energy contained in the third approximation decreased by a half comparing to the third case. Recall that we locate the position of the paint can in each frame by locating the bright light on top of it. When there is rotation in the system, we are unable to correctly locate the paint can as in some frame, we only see the back of the bright light source. I think this inaccuracy of data is what caused the change in the energy captured in the third approximation and we continue to see the impact of rotation in the plot of the third principal component in figure 4 as the plot get more noisy comparing to the plot of the third principal component in the third case. Note that for both case 3 and 4, I didn't include the plot of the fourth principal component because the corresponding energy captured isn't significant so that I think the plot would be noisy and not provide any additional information.

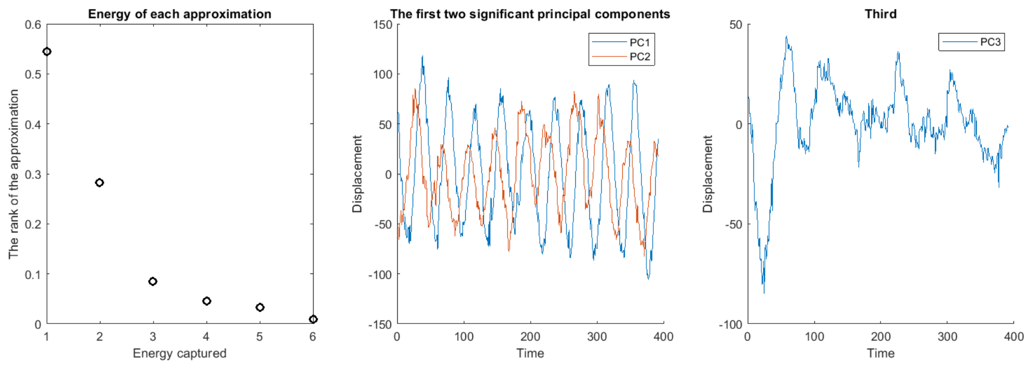

Figure 4: Energy plot and the plot of significant principal components for case 4

5 Summary and Conclusions

In this assignment, we analyzed the video files for four cases of paint can oscillation by performing the Principal Component Analysis on the data. From PCA, we are able to compute the principal components of the data which describe the general motion of the paint can in each direction. Moreover, we plotted the amount of the energy contained in each approximations which give us the information regarding the significance of the principal components. This is a really enjoyable assignment because it is a great way to learn how to apply PCA and SVD with real life data.

Appendix A MATLAB Functions

Add your important MATLAB functions here with a brief implementation explanation. This is how to make an **unordered** list:

- `load(filename)` loads variables in the MAT-File into the MATLAB workspace
- `[M,I] = max(A)` return a maximum value M in the matrix A and the corresponding index of M.
- `[I1,I2,...,In] = ind2sub(sz,ind)` returns n arrays I1,I2,...,In containing the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz.
- `I = rgb2gray(RGB)` converts the truecolor image RGB to the grayscale image I
- `[U,S,V] = svd(A)` performs a singular value decomposition of matrix A, such that A = U*S*V'.
- `D = diag(v)` returns a square diagonal matrix with the elements of vector v on the main diagonal.
- `M = mean(A,dim)` returns the mean along dimension dim.

- `X = zeros(n)` returns an n-by-n matrix of zeros.
- `imshow(I)` displays the grayscale image I in a figure.

Appendix B MATLAB Code

```
%%
clear ; close all; clc
%% Watch movie
load('cam1_4.mat');
filter = zeros(480,640);
filter(:, :) = 1;
numFrames = size(vidFrames1_4,4);
for j = 1:numFrames
    X = vidFrames1_4(:,:,:,j);
    imshow(X);
    drawnow
end
%% Method 1
clear ; close all; clc
%% Camera 1
% The range of the bucket is from 310 to 385
filter = zeros(480,640);
filter(47:433,310:385) = 1;
coord1 = [];
load('cam1_1.mat');
numFrames = size(vidFrames1_1,4);
for j = 1:numFrames
    X = vidFrames1_1(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd.*filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-20:x+20,y-20:y+20) = 1;
    coord1 = [coord1, [y;x]];
end
%% Camera 2
load('cam2_1.mat');
coord2 = [];
filter = zeros(480,640);
filter(100:433,250:330) = 1;
numFrames = size(vidFrames2_1,4);
for j = 1:numFrames
    X = vidFrames2_1(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd.*filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-20:x+20,y-20:y+20) = 1;
    coord2 = [coord2, [y;x]];
```

```
end
%% Camera 3
load('cam3_1.mat');j
coord3 = [];
% Motion is in X direction so change filter
filter = zeros(480,640);
filter(250:330,220:end) = 1;
numFrames = size(vidFrames3_1,4);
for j = 1:numFrames
    X = vidFrames3_1(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd.*filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-20:x+20,y-20:y+20) = 1;
    coord3 = [coord3, [y;x]];
end
%% SVD
trim = size(coord1);
coord2 = coord2(:,1:trim(2));
coord3 = coord3(:,1:trim(2));
X = [coord1;coord2;coord3];
avgs = mean(X,2);
for j = 1:6
    X(j,:) = X(j,:) - avgs(j);
end
A = X/sqrt(trim(2)-1);
[U,S,V] = svd(A,'econ');
%% Ploting the energy curve
subplot(1,2,1);
sig = diag(S);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title('Energy of each approximation');
xlabel('Energy captured');
ylabel('The rank of the approximation');
%% Principal Component projection
p = U' * X;
subplot(1,2,2)
hold on
plot(1:trim(2),p(1,:))
plot(1:trim(2),p(2,:))
title('Plot of the significant principal components')
ylabel('Displacement');
xlabel('Time');
legend('PC1','PC2');
%% Method 2
clear ; close all; clc
%% Camera 1
load('cam1_2.mat');
filter = zeros(480,640);
filter(1:320,300:430) = 1;
coord1 = [];
```

```
numFrames = size(vidFrames1_2,4);
for j = 1:numFrames
    X = vidFrames1_2(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-20:x+12,y-20:y+20) = 1;
    coord1 = [coord1, [y;x]];
end
%% Camera 2
load('cam2_2.mat');
numFrames = size(vidFrames2_2,4);
filter = zeros(480,640);
filter(1:370,210:380) = 1;
coord2 = [];
for j = 1:numFrames
    X = vidFrames2_2(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd.*filter;
%     imshow(uint8(Xgrayf));
%     drawnow
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-50:x+45,y-40:y+40) = 1;
    coord2 = [coord2, [y;x]];
end
%% Camera 3
load('cam3_2.mat');
filter = zeros(480,640);
filter(160:330,:) = 1;
coord3 = [];
numFrames = size(vidFrames3_2,4);
for j = 1:numFrames
    X = vidFrames3_2(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
%     imshow(uint8(Xgrayf));
%     drawnow
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-30:x+30,y-30:y+30) = 1;
    coord3 = [coord3, [y;x]];
end
%% SVD
trim = size(coord1);
coord2 = coord2(:,1:trim(2));
coord3 = coord3(:,1:trim(2));
```

```
X = [coord1;coord2;coord3];
avgs = mean(X,2);
for j = 1:6
    X(j,:) = X(j,:) - avgs(j);
end
% let A = X/sqrt(n-1)
A = X/sqrt(trim(2)-1);
[U,S,V] = svd(A,'econ');
sig = diag(S);
%% Ploting the energy curve
subplot(1,3,1);
sig = diag(S);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title('Energy of each approximation');
xlabel('Energy captured');
ylabel('The rank of the approximation');
%% Principal Component projection
p = U' * X;
subplot(1,3,2)
hold on
plot(1:trim(2),p(1,:))
plot(1:trim(2),p(2,:))
title('First two significant principal components')
ylabel('Displacement');
xlabel('Time');
legend('PC1','PC2');
subplot(1,3,3);
hold on
plot(1:trim(2),p(3,:))
plot(1:trim(2),p(4,:))
title('Third and fourth significant principal components')
ylabel('Displacement');
xlabel('Time');
legend('PC3','PC4');
%% Method 3
clear ; close all; clc
%% Camera 1
load('cam1_3.mat');
filter = zeros(480,640);
filter(:,250:400) = 1;
coord1 = [];
numFrames = size(vidFrames1_3,4);
for j = 1:numFrames
    X = vidFrames1_3(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-20:x+20,y-20:y+20) = 1;
    coord1 = [coord1, [y;x]];
end
%% Camera 2
```

```
load('cam2_3.mat');
filter = zeros(480,640);
filter(:,200:420) = 1;
coord2 = [];
numFrames = size(vidFrames2_3,4);
for j = 1:numFrames
    X = vidFrames2_3(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-40:x+40,y-40:y+40) = 1;
    coord2 = [coord2, [y;x]];
end
%% Camera 3
load('cam3_3.mat');
filter = zeros(480,640);
filter(180:340,:) = 1;
numFrames = size(vidFrames3_3,4);
coord3 = [];
for j = 1:numFrames
    X = vidFrames3_3(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-30:x+30,y-30:y+30) = 1;
    coord3 = [coord3, [y;x]];
end
%% SVD
trim = size(coord3);
coord1 = coord1(:,1:trim(2));
coord2 = coord2(:,1:trim(2));
X = [coord1;coord2;coord3];
avgs = mean(X,2);
for j = 1:6
    X(j,:) = X(j,:) - avgs(j);
end
% let A = X/sqrt(n-1)
A = X/sqrt(trim(2)-1);
[U,S,V] = svd(A,'econ');
sig = diag(S);
%% Ploting the energy curve
subplot(1,3,1);
sig = diag(S);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title('Energy of each approximation');
xlabel('Energy captured');
ylabel('The rank of the approximation');
%% Principal Component projection
```

```
p = U' * X;
subplot(1,3,2)
hold on
plot(1:trim(2),p(1,:))
plot(1:trim(2),p(2,:))
title('First two significant principal components')
ylabel('Displacement');
xlabel('Time');
legend('PC1','PC2','PC3','PC4');
subplot(1,3,3)
hold on
plot(1:trim(2),p(3,:))
title('Third principal component')
ylabel('Displacement');
xlabel('Time');
legend('PC3');
%% Method 4
clear ; close all; clc
%% Camera 1
load('cam1_4.mat');
filter = zeros(480,640);
filter(:,300:480) = 1;
numFrames = size(vidFrames1_4,4);
coord1 = [];
for j = 1:numFrames
    X = vidFrames1_4(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-30:x+30,y-30:y+30) = 1;
    coord1 = [coord1, [y;x]];
end
%% Camera 2
load('cam2_4.mat');
filter = zeros(480,640);
filter(:,200:400) = 1;
numFrames = size(vidFrames2_4,4);
coord2 = [];
for j = 1:numFrames
    X = vidFrames2_4(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-30:x+30,y-30:y+30) = 1;
    coord2 = [coord2, [y;x]];
end
%% Camera 3
load('cam3_4.mat');
```

```
filter = zeros(480,640);
filter(140:280,250:end) = 1;
numFrames = size(vidFrames3_4,4);
coord3 = [];
for j = 1:numFrames
    X = vidFrames3_4(:,:,:,j);
    Xgray = rgb2gray(X);
    Xgrayd = double(Xgray);
    Xgrayf = Xgrayd .* filter;
    [mv,idx] = max(Xgrayf(:));
    [x,y] = ind2sub(size(Xgrayf), idx);
    filter = zeros(480,640);
    filter(x-30:x+30,y-30:y+30) = 1;
    coord3 = [coord3, [y;x]];
end
%% SVD
trim = size(coord1);
coord2 = coord2(:,1:trim(2));
coord3 = coord3(:,1:trim(2));
X = [coord1;coord2;coord3];
avgs = mean(X,2);
for j = 1:6
    X(j,:) = X(j,:) - avgs(j);
end
% let A = X/sqrt(n-1)
A = X/sqrt(trim(2)-1);
[U,S,V] = svd(A,'econ');
sig = diag(S);
%% Ploting the energy curve
subplot(1,3,1);
sig = diag(S);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2)
title('Energy of each approximation');
xlabel('Energy captured');
ylabel('The rank of the approximation');
%% Principal Component projection
p = U' * X;
subplot(1,3,2)
hold on
plot(1:trim(2),p(0,:))
plot(1:trim(2),p(2,:))
title('The first two significant principal components')
ylabel('Displacement');
xlabel('Time');
legend('PC1','PC2');
subplot(1,3,3)
hold on
plot(1:trim(2),p(3,:))
title('Third')
ylabel('Displacement');
xlabel('Time');
legend('PC3');
```