Jeremy Aguirre

AMATH 482

January 24, 2020

Assignment #3: Applying Principal Component Analysis to a Mass-Spring System

Abstract

In this assignment, we were provided with four sets of three videos in a .mat format. Each set of videos corresponded to a particular case, Case 1 being the ideal case in which a can is undergoing simple harmonic motion in the z direction, Case 2 being the noisy case in which behavior similar to Case 1 is observed through shaky cameras. Case 3 is the same as Case 1 with the addition of a pendulum-like motion of the can. Case 4 is the same as Case 3 with the addition of the can rotating. Principle component analysis is applied to the three videos in each case and the results of this analysis are interpreted through graphs depicting the energy and displacement contributions of each principle component.

I. Introduction and Overview

In this assignment, we were provided with four sets of three videos in a .mat format. Each set of videos corresponded to a particular case, Case 1 being the ideal case in which a can is undergoing simple harmonic motion in the z direction, Case 2 being the noisy case in which behavior similar to Case 1 is observed through shaky cameras. Case 3 is the same as Case 1 with the addition of a pendulum-like motion of the can. Case 4 is the same as Case 3 with the addition of the can rotating. Principle component analysis is applied to the three videos in each case and the results of this analysis are interpreted through graphs depicting the energy and displacement contributions of each principle component.

II. Theoretical Background

Singular Value Decomposition

The representation of the full SVD is as follows:

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

$$\mathbf{U} \in \mathbb{C}^{m\times m} \text{ is unitary}$$

$$\mathbf{V} \in \mathbb{C}^{n\times n} \text{ is unitary}$$

$$\mathbf{\Sigma} \in \mathbb{R}^{m\times n} \text{ is diagonal}$$

This decomposition tool is used to isolate constitutive elements from the matrix **A** which can be interpreted based on the application. A common example used to describe the SVD is the decomposition of a disk characterized by a 2x2 matrix. In this example, $\mathbf{V}^{*}$ is thought of as the component which rotates a unit disk. $\mathbf{\Sigma}$ is responsible for the stretching or compression of the disk for this case in two dimensions. The values along the diagonal of $\mathbf{\Sigma}$ are known as the singular values (σ_i). The matrix **U** then rotates the now ellipse again to yield an ellipse characterized by matrix **A.** The singular vale decomposition has various uses depending on the application.

Principal Component Analysis

One such application of the singular vale decomposition is in principle component analysis. PCA is used to remove noisy and redundant data in order to extract a simplified or ideal behavior of a system. In order to do this, data regarding a system is converted into the matrix **X** in order to compute the covariance matrix $\mathbf{C_x}$ as follows:

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

With the covariance matrix, SVD can be applied and $\mathbf{\Sigma}$ can be used to find the energy for each mode (principle component). Through this process, dominant principle components can be analyzed to understand the simplified behavior of a system.

III. Algorithm Implementation and Development

After loading the .mat files containing the video information to be analyzed, the command 'size' was used to extract the number of frames in each video. A for loop was used to extract grayscale data from each frame of the first video using the command 'rgb2gray.' The information was converted to grayscale in order to reduce the data size. The data was then converted to double precision using the command 'double.' At this point, the command 'pcolor' was used to plot the video information in a way which allowed for easy identification of the intervals of interest (as the can is in motion). The unnecessary data was removed and of the remaining data, the max values were found using the 'max' command. The indices of the max values beyond a particular threshold were then converted into x and y coordinates using the command 'find.' The average of these values was found for each pixel thus completing the pseudo normalization of the max values (corresponding to the position of the can). This process is repeated for the other two videos of the same case to find the relative x and y coordinates of the can. The resulting three sets of x, y coordinates are of different length and so the shortest video is determined using the 'min' command and the longer vectors of video information are cut to match. The three resulting vectors of the same length are put into a matrix. The size of this matrix is determined using the 'size' command, the average of each row in the matrix is determined using the command 'mean' and the command 'repmat' is used to subtract each mean from its respective row in the matrix. This yields a matrix of information (regarding the position of the can) which can be manipulated using the singular value decomposition command, 'svd.' This yields a matrix 's' of which each element in the diagonal is divided by the sum of all diagonal elements. This provides the fractional energy corresponding to each principle component. Then, matrix multiplication between 'v' and 's' is used to show the displacement relative to each principle components energy contribution. This entire process is repeated for the other three cases.

IV. Computational Results

Case 1

According to Figure 1 and Figure 2, there is only one dominant component for the ideal case. This is expected since the can is experiencing simple harmonic motion in the z direction only. The oscillation of the can's movement in the z direction can be seen clearly in Figure 1.

Case 2

According to Figure 3 and Figure 4, there is still only one dominant component for the noisy case. However, the other principle components have a larger contribution resulting from the shakiness of the camera. This also accounts for the component corresponding to the can's simple harmonic motion having a smaller fractional energy contribution.

Case 3

From Figure 6, it is clear that there are two dominant principle components for Case 3. This is what is expected as the can is experiencing simple harmonic oscillations in the z direction and pendulum motion in the x, y plane. These oscillatory movements can be seen in Figure 5. The most dominant principle component corresponds to movement in the z direction while the second dominant corresponds to movement in the x, y plane. This makes sense because the pendulum motion of the can contributes to both movement in the z direction and in the x, y plane, while the simple harmonic movement contributes to movement in the z direction.

Case 4

Here, we expect to see three dominant principle components, and there is a noticeable increase in the fractional energy contribution of the second and third components compared to the first case. However, this is not totally clear through PCA. This is likely due to the fact that rotation of the can does not contribute as much energy as its simple harmonic motion or pendulum motion. Also, this rotation detracts from the energy contribution of the pendulum motion.

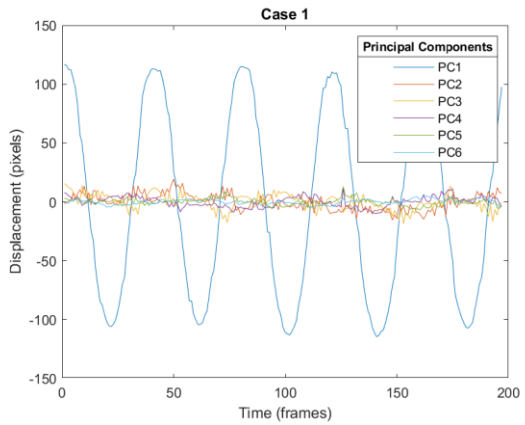

Figure 1. The displacement corresponding to each principle component over time for Case 1

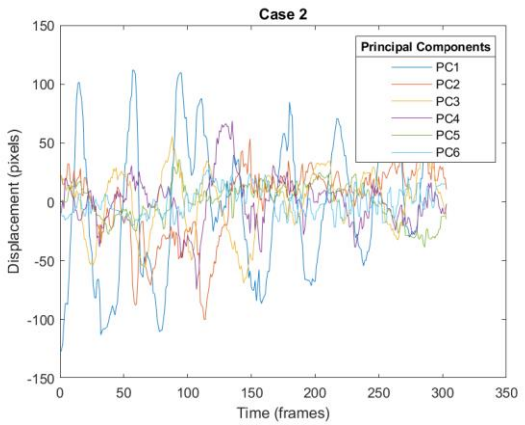

Figure 3. The displacement corresponding to each principle component over time for Case 2

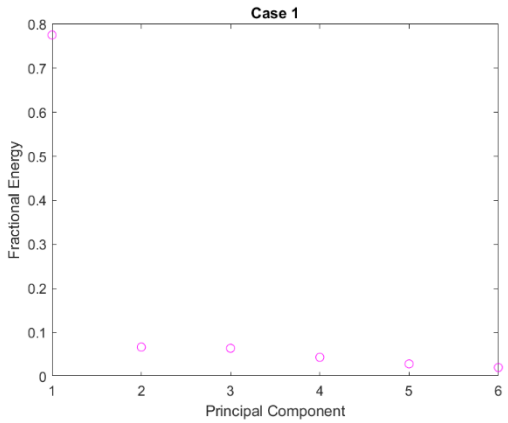

Figure 2. Energy contribution of each principle component for Case 1

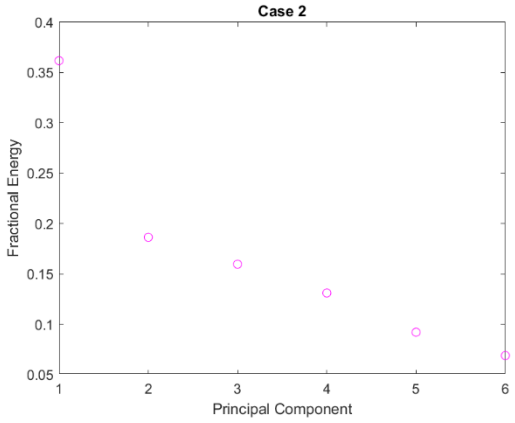

Figure 4. Energy contribution of each principle component for Case 2

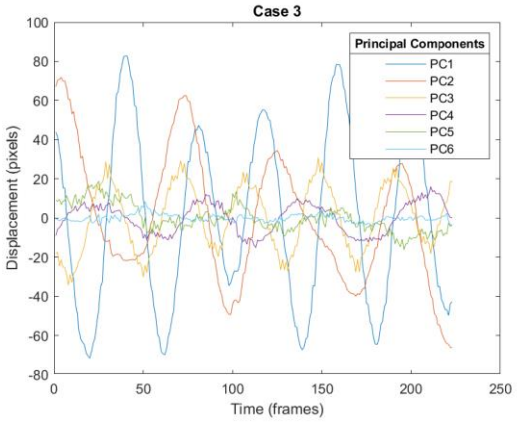

Figure 5. The displacement corresponding to each principle component over time for Case 3

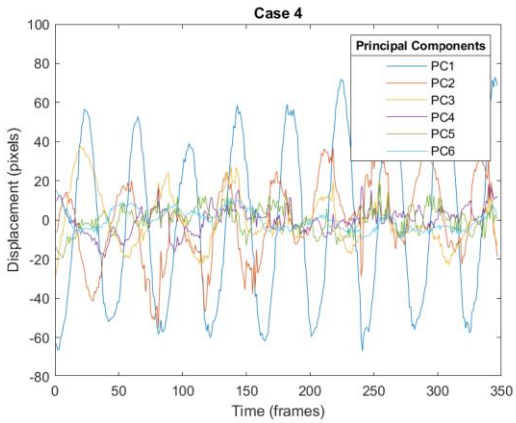

Figure 7. The displacement corresponding to each principle component over time for Case 4

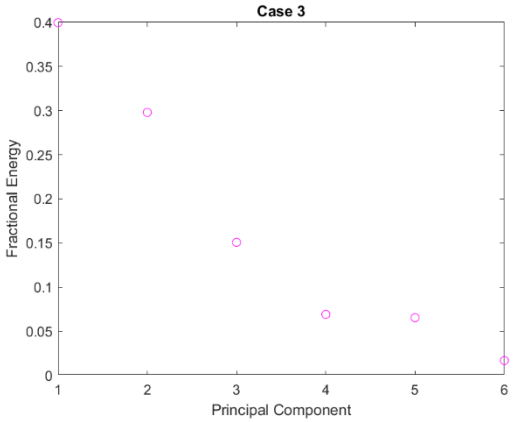

Figure 6. Energy contribution of each principle component for Case 3

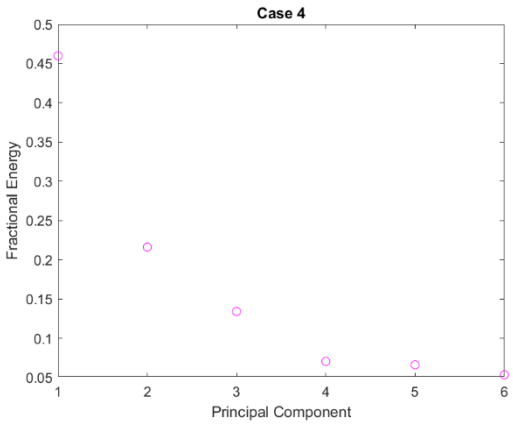

Figure 8. Energy contribution of each principle component for Case 4

V. Summary and Conclusions

In this assignment, we were provided with four sets of three videos in a .mat format. Each set of videos corresponded to a particular case, Case 1 was the ideal case in which a can underwent simple harmonic motion in the z direction, Case 2 was the noisy case in which behavior similar to Case 1 was observed through shaky cameras. Case 3 was the same as Case 1 with the addition of a pendulum-like motion of the can. Case 4 was the same as Case 3 with the addition of the can rotating. Principle component analysis was applied to the three videos in each case and the results of this analysis were interpreted through graphs depicting the energy and displacement contributions of each principle component. Based on the principle component analysis performed in this assignment, I would have to say that it is an effective means of extracting information necessary to model simple systems. However, in Case 4, the method appeared to be less effective due to the complexity of the movement of the can being analyzed. When there are too many principle components contributing to the systems behavior, it becomes more difficult to distinguish these components from noise. The difficulty of interpreting principle component analysis in the presence of noise was demonstrated in Case 2 and this was an illustration of one of the downfalls of PCA.

Appendix A

svd- matlab function to perform the singular value decomposition

Appendix B

```
%% AMATH 482: Assignment #3
close all, clear, clc

load cam1_1;
numFrames1_1 = size(vidFrames1_1, 4);
load cam2_1;
numFrames2_1 = size(vidFrames2_1, 4);
load cam3_1;
numFrames3_1 = size(vidFrames3_1, 4);
load cam1_2;
numFrames1_2 = size(vidFrames1_2, 4);
load cam2_2;
numFrames2_2 = size(vidFrames2_2, 4);
load cam3_2;
numFrames3_2 = size(vidFrames3_2, 4);
load cam1_3;
numFrames1_3 = size(vidFrames1_3, 4);
load cam2_3;
numFrames2_3 = size(vidFrames2_3, 4);
load cam3_3;
numFrames3_3 = size(vidFrames3_3, 4);
load cam1_4;
numFrames1_4 = size(vidFrames1_4, 4);
load cam2_4;
numFrames2_4 = size(vidFrames2_4, 4);
```

```
load cam3_4;
numFrames3_4 = size(vidFrames3_4, 4);

%% Test 1: Ideal Case
% Camera 1
x1_1 = [];
y1_1 = [];
for j=1:numFrames1_1
    A = double(rgb2gray(vidFrames1_1(:,:,:,j)));
    A(:, [1:300 400:end]) = 0;
    A([1:200 400:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x1_1(j) = mean(N);
    y1_1(j) = mean(M);
end

[Y, I] = max(y1_1(1:50));
x1_1 = x1_1(30:end);
y1_1 = y1_1(30:end);

% Camera 2
x2_1 = [];
y2_1 = [];
for j=1:numFrames2_1
    A = double(rgb2gray(vidFrames2_1(:,:,:,j)));
    A(:, [1:220 350:end]) = 0;
    A([1:100 350:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x2_1(j) = mean(N);
    y2_1(j) = mean(M);
end

[Y, I] = max(y2_1(1:41));
x2_1 = x2_1(I:end);
y2_1 = y2_1(I:end);

% Camera 3
x3_1 = [];
y3_1 = [];
for j=1:numFrames3_1
    A = double(rgb2gray(vidFrames3_1(:,:,:,j)));
    A(:, [1:200 480:end]) = 0;
    A([1:230 350:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
```

```
    x3_1(j) = mean(N);
    y3_1(j) = mean(M);
end

[Y, I] = max(y3_1(1:41));
x3_1 = x3_1(I:end);
y3_1 = y3_1(I:end);

%% PCA
L = min([length(y1_1), length(y2_1), length(x3_1)]);
X = [x1_1(1:L); y1_1(1:L); x2_1(1:L); y2_1(1:L); x3_1(1:L);
y3_1(1:L)];
[m, n] = size(X);
mn = mean(X, 2);
X = X-repmat(mn,1,n);
[U, S, V] = svd(X, 'econ');
V = V*S;

figure(1)
plot(diag(S)./sum(diag(S)), 'mo')
xlabel('Principal Component')
ylabel('Fractional Energy')
title('Case 1')

figure(2)
plot(V(:,1)), hold on
plot(V(:,2))
plot(V(:,3))
plot(V(:,4))
plot(V(:,5))
plot(V(:,6)), hold off
xlabel('Time (frames)')
ylabel('Displacement (pixels)')
title('Case 1')
lgd = legend('PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6');
title(lgd, 'Principal Components')

print('-f1','Energy_1','-dpng')
print('-f2','Case_1','-dpng')

%% Test 2: Noisy Case
% Camera 1
x1_2 = [];
y1_2 = [];
for j=1:numFrames1_2
    A = double(rgb2gray(vidFrames1_2(:,:,:,j)));
    A(:, [1:300 400:end]) = 0;
```

```
    A([1:200 400:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x1_2(j) = mean(N);
    y1_2(j) = mean(M);
end

[Y, I] = max(y1_2(1:30));
x1_2 = x1_2(I:end);
y1_2 = y1_2(I:end);

% Camera 2
x2_2 = [];
y2_2 = [];
for j=1:numFrames2_2
    A = double(rgb2gray(vidFrames2_2(:,:,:,j)));
    A(:, [1:200 400:end]) = 0;
    A([1:50 370:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x2_2(j) = mean(N);
    y2_2(j) = mean(M);
end

[Y, I] = max(y2_2(1:30));
x2_2 = x2_2(I:end);
y2_2 = y2_2(I:end);

% Camera 3
x3_2 = [];
y3_2 = [];
for j=1:numFrames3_2
    A = double(rgb2gray(vidFrames3_2(:,:,:,j)));
    A(:, [1:250 480:end]) = 0;
    A([1:180 320:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x3_2(j) = mean(N);
    y3_2(j) = mean(M);
end

[Y, I] = max(y3_2(1:40));
x3_2 = x3_2(I:end);
y3_2 = y3_2(I:end);

%% PCA
L = min([length(y1_2), length(y2_2), length(x3_2)]);
```

```
X = [x1_2(1:L); y1_2(1:L); x2_2(1:L); y2_2(1:L); x3_2(1:L);
y3_2(1:L)];
[m, n] = size(X);
mn = mean(X, 2);
X = X-repmat(mn,1,n);
[U, S, V] = svd(X, 'econ');
V = V*S;

figure(3)
plot(diag(S)./sum(diag(S)), 'mo')
xlabel('Principal Component')
ylabel('Fractional Energy')
title('Case 2')

figure(4)
plot(V(:,1)), hold on
plot(V(:,2))
plot(V(:,3))
plot(V(:,4))
plot(V(:,5))
plot(V(:,6)), hold off
xlabel('Time (frames)')
ylabel('Displacement (pixels)')
title('Case 2')
lgd = legend('PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6');
title(lgd, 'Principal Components')

print('-f3','Energy_2','-dpng')
print('-f4','Case_2','-dpng')

%% Test 3: Horizontal Displacement
% Camera 1
x1_3 = [];
y1_3 = [];
for j=1:numFrames1_3
    A = double(rgb2gray(vidFrames1_3(:,:,:,j)));
    A(:, [1:250 400:end]) = 0;
    A([1:200 400:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x1_3(j) = mean(N);
    y1_3(j) = mean(M);
end

[Y, I] = max(y1_3(1:30));
x1_3 = x1_3(I:end);
y1_3 = y1_3(I:end);
```

```
% Camera 2
x2_3 = [];
y2_3 = [];
for j=1:numFrames2_3
    A = double(rgb2gray(vidFrames2_3(:,:,:,j)));
    A(:, [1:220 420:end]) = 0;
    A([1:150 400:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x2_3(j) = mean(N);
    y2_3(j) = mean(M);
end

[Y, I] = max(y2_3(1:50));
x2_3 = x2_3(I:end);
y2_3 = y2_3(I:end);

% Camera 3
x3_3 = [];
y3_3 = [];
for j=1:numFrames3_3
    A = double(rgb2gray(vidFrames3_3(:,:,:,j)));
    A(:, [1:150 480:end]) = 0;
    A([1:180 350:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x3_3(j) = mean(N);
    y3_3(j) = mean(M);
end

[Y, I] = max(y3_3(1:30));
x3_3 = x3_3(I:end);
y3_3 = y3_3(I:end);

%% PCA
L = min([length(y1_3), length(y2_3), length(x3_3)]);
X = [x1_3(1:L); y1_3(1:L); x2_3(1:L); y2_3(1:L); x3_3(1:L);
y3_3(1:L)];
[m, n] = size(X);
mn = mean(X, 2);
X = X-repmat(mn,1,n);
[U, S, V] = svd(X, 'econ');
V = V*S;

figure(5)
plot(diag(S)./sum(diag(S)), 'mo')
```

```
xlabel('Principal Component')
ylabel('Fractional Energy')
title('Case 3')

figure(6)
plot(V(:,1)), hold on
plot(V(:,2))
plot(V(:,3))
plot(V(:,4))
plot(V(:,5))
plot(V(:,6)), hold off
xlabel('Time (frames)')
ylabel('Displacement (pixels)')
title('Case 3')
lgd = legend('PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6');
title(lgd, 'Principal Components')

print('-f5','Energy_3','-dpng')
print('-f6','Case_3','-dpng')

%% Test 4: Horizontal Displacement and Rotation
% Camera 1
x1_4 = [];
y1_4 = [];
for j=1:numFrames1_4
    A = double(rgb2gray(vidFrames1_4(:,:,:,j)));
    A(:, [1:300 470:end]) = 0;
    A([1:200 380:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x1_4(j) = mean(N);
    y1_4(j) = mean(M);
end

[Y, I] = max(y1_4(1:41));
x1_4 = x1_4(I:end);
y1_4 = y1_4(I:end);

% Camera 2
x2_4 = [];
y2_4 = [];
for j=1:numFrames2_4
    A = double(rgb2gray(vidFrames2_4(:,:,:,j)));
    A(:, [1:220 410:end]) = 0;
    A([1:50 400:end], :) = 0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
```

```
    x2_4(j) = mean(N);
    y2_4(j) = mean(M);
end

[Y, I] = max(y2_4(1:50));
x2_4 = x2_4(I:end);
y2_4 = y2_4(I:end);

% Camera 3
x3_4 = [];
y3_4 = [];
for j=1:numFrames3_4
    A = double(rgb2gray(vidFrames3_4(:,:,:,j)));
    A(:, [1:300 510:end]) = 0;
    A([1:150 290:end], :) = 0;
    A(1:150, :) = 0; A(290:end, :) = 0; A(:, 1:300) = 0; A(:,
510:end)=0;
    [Y, I] = max(A(:));
    [M, N] = find(A >= 11/12 * Y);
    x3_4(j) = mean(N);
    y3_4(j) = mean(M);
end

[Y, I] = max(y3_4(1:50));
x3_4 = x3_4(I:end);
y3_4 = y3_4(I:end);

%% PCA
L = min([length(y1_4), length(y2_4), length(x3_4)]);
X = [x1_4(1:L); y1_4(1:L); x2_4(1:L); y2_4(1:L); x3_4(1:L);
y3_4(1:L)];
[m, n] = size(X);
mn = mean(X, 2);
X = X-repmat(mn,1,n);
[U, S, V] = svd(X, 'econ');
V = V*S;

figure(7)
plot(diag(S)./sum(diag(S)), 'mo')
xlabel('Principal Component')
ylabel('Fractional Energy')
title('Case 4')

figure(8)
plot(V(:,1)), hold on
plot(V(:,2))
plot(V(:,3))
```

```
plot(V(:,4))
plot(V(:,5))
plot(V(:,6)), hold off
xlabel('Time (frames)')
ylabel('Displacement (pixels)')
title('Case 4')
lgd = legend('PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6');
title(lgd, 'Principal Components')

print('-f7','Energy_4','-dpng')
print('-f8','Case_4','-dpng')
```