AMATH 482 HW 3

Liyuan Tang

February 2019

Abstract

In this homework, we are given the data of movie files (Matlab files) which contains an oscillating can. The movie is created from the cameras at 3 different angles in 4 cases, including ideal case, noisy case, horizontal displacement, and horizontal displacement and rotation. We are going to illustrate different aspects of the principal component analysis and its behaviors in 4 different cases. Finally, I will plot the principal components with the energy percentage to determine the number of dominant component.

1 Introduction

We have the data created by 3 cameras at 3 different angles in 4 cases. The first case is the ideal case where a small displacement of can occurs in the z direction. The second case is a noisy case. It just repeats the ideal case but with shaking cameras. The third one is horizontal displacement. The can is released off-center so it produces the motion in all x, y, and z directions. Ideally, it is a pendulum motion with a simple harmonic oscillation. The last one is repeating the third case but with a rotation. In each case, I will reproduce the movie with those data files in order to find the displacement of the can under each camera and store them through x and y coordinates. Then, I will construct the matrix by those 6 vectors and apply the singular value decomposition to do the principal component analysis.

2 Theoretical Background

2.1 Singular Value Decomposition

The singular value decomposition is a factorization of matrix into a number of constitutive components all of which have a specific meaning in applications [1]. It is a transformation that stretches or compresses a given set of vector and then rotate it. The representation of full SVD is:

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

The size of A is m $\times$ n. And U is an m $\times$ m unitary matrix, with orthogonal columns. Σ is an m $\times$ n diagonal matrix. And V is an n $\times$ n unitary matrix. Besides, the diagonal elements of Σ are nonnegative and sorted from largest to smallest. Thus, the SVD of the matrix A shows that the matrix first uses a unitary transformation which is a rotation

through V^*. Then, it applies a stretching operation by Σ. Finally, it rotates again by the unitary transformation U.

Based on the theorem in the notes, every matrix A has a singular value decomposition. Thus, it enables every matrix to be diagonalized if we choose some proper basis for the range and domain, U and V.

2.2 Principal Component Analysis

Principal component analysis is one of the applications of the SVD. It usually used as a method to clean data which is unknown but might be low dimensional. Through the process of PCA, we want to remove the noisy and redundant data in order to extract the ideal or simplified behavior [1]. First, we need to collect the data and store them in a single matrix, X. In this homework, the size of X should be $6 \times$ n where 6 is the number of measurement types and n is the number of data points, which is the frame numbers in this homework. Then, we can compute the covariance matrix based on the definition:

$$C_X = \frac{1}{n-1} X X^T. \tag{2}$$

C_X is a square and symmetric matrix whose diagonal represents the variance of corresponding measurement. The off-diagonal elements are the covariance between two different measurement types. A large off diagonal term indicates the redundancy and a large diagonal term indicates a strong fluctuation in that variable. Thus, we want to remove the redundancy and identify the signals with large variance. As a result, we need to diagonalize the covariance matrix. However, this is what exactly SVD does. With the diagonalized matrix Σ, we can compute the energy for each mode which is the sigma of the mode divides the sum of the sigma. The mode with the largest energy is the most dominant one.

3 Algorithm Implementation and Development

The first case is the ideal case and the object should move up and down in only one direction. I load the data in cam$N$_1.mat where N=1,2,3. When I called "size(vidFrames1_1)", the answer is [480, 640, 3, 226]. The first two represent the size of x and y. The third one is three since it is an RGB image. The last one represent the number of frames in the data. I extracted the number of frames by using "numFrames1 = size(vidFrames1_1, 4)".

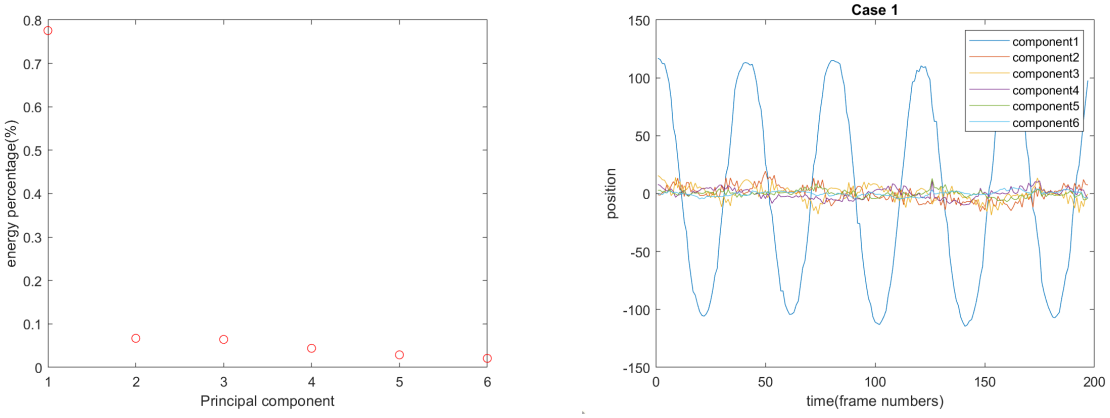
I used a for loop from 1 to numFrames1. For each iteration, I extracted the data at that frame, used "rgb2gray" to convert the image to grayscale so that I can reduce the size of the data. Then, I converted the data to double precision by the command "double()". At this point, the data is 480×640 double which the value at every point indicates the value under grayscale. I used "pcolor" to reproduce the movie and found a interval of the mass moving. I kept the value of data in that interval and make the value outside the interval to be 0, just like the Shannon filter in the last homework. Under gray scale, larger value indicates white color which means 0 indicates black. Since I converted the background outside that interval to black, I tried to find the max value in this data and its index by using "[V, I] = max(X(:))". Instead of converting the index to subscript values of x and y by using "ind2sub", I used "[aaa, bbb] = find(X1 $\geq$ V * 11 / 12)". It gives me two vectors which represent the x and y coordinates of the points in X1 that

is greater of equal than $\frac{11}{12}V$. This method will generate a more accurate location of the can. Then I took the mean of aaa and bbb and stored them in two separate vectors that allows me to see how the can moves in that direction as time goes.

I did the same thing for the other two cameras except changing the numFrames and using different filters. One thing to notice is that the third camera flips the x and y. So based on my implementation, when I plot the movement of the can I should plot x for the third camera while we plot y for the first two camera. At the end, I found those three cameras didn't start at the same time. So I tried to line up them at the point where they reach the first maximum. I used "[v, i] = max(y_1(1:41))" to find the index and used "x_1 = x_1(i:end); y_1 = y_1(i:end);" to cut the data. And I did the same thing for the rest two cameras.

After doing these things, I have three x, y pairs with different length. I want to put them in one matrix, then I need to make the length of these vectors the same. I found the shortest length, l, among those 3 pairs by using "l = min([length(y_1), length(y_2), length(x_3)])", cut longer vectors into the same length and put them in the matrix X row by row. I want to subtract mean of each row in order to make them all center at 0. I find the size of X by using "[m, n] = size(X)". And I find the mean of each row by using "mn = mean(X, 2)" which mn is a column vector and each element indicates the mean of corresponding row in X. Then, I used "X=X-repmat(mn,1,n)" to subtract the mean in each row. With this matrix X, I can do the singular value decomposition by using "[u, s, v] = svd(X, 'econ')". Here "econ" means a economic-size decomposition that the size of v is n $\times$ m. I plot the diagonal of s matrix divided by the sum of diagonal that represents the energy percentage of each mode. Finally, I used v * s which is the location of principle component in time multiplies by the corresponding energy. I plotted every column in this matrix which clearly shows the movement of each principle component in order to find the number of dominant components.

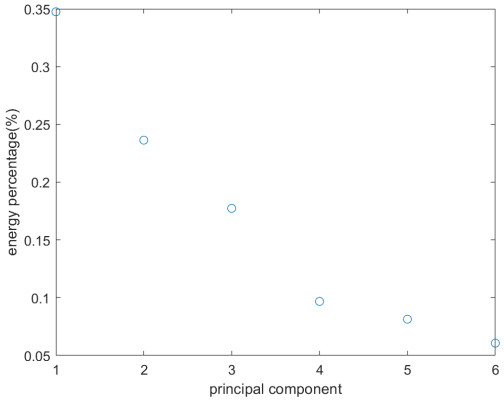
For the other three cases, I did the same thing to construct the matrix X and the singular value decomposition to find the dominant component.



(a) Energy percentage for the ideal case

(b) The principle component in the basis of u for the ideal case

Figure 1: Energy percentage and the principle components for the ideal case (case 1).



(a) Energy percentage for the noisy case

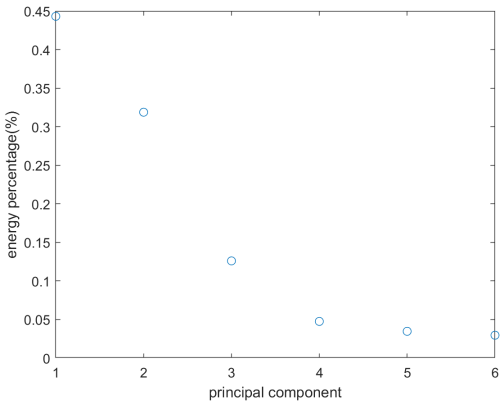
(b) The principle component in the basis of u for the noisy case

Figure 2: Energy percentage and the principle components for the noisy case (case 2).

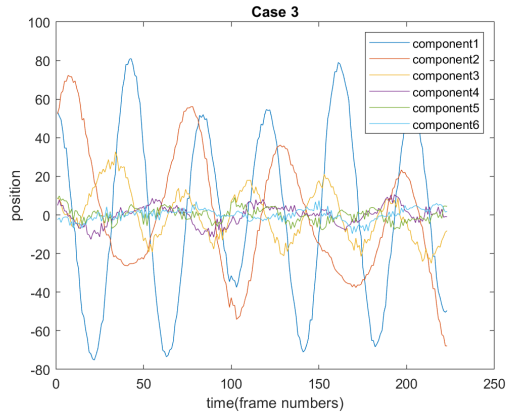
4 Computational Results

1. Case 1: Based on the sigma (Figure 1(a)), it is clear that there is only one dominant component in the ideal case. It is reasonable since the can is only moving up and down under a simple harmonic oscillation. From the Figure 1(b), when I used the sigma times v, it is clear the first component is the dominant component in the ideal case.

2. Case 2: This is the noisy case where the cameras are shaking. As a result, the data may have a lot of noise. From the sigma plot(Figure 2(a)), there is still one dominant component but the percentage is less than 40% while in the ideal case it is close to 80%. From Figure 2(b), we can still see the component 1 is the most dominant one but the component 2 is also significant in some way. This is because a wide moving range of the can under the shaking cameras makes it hard to extract clear data. Thus, the data may have a lot of displacement in the x direction due to the shaking camera which may result in the behavior of component 2 in the Figure 2(b).

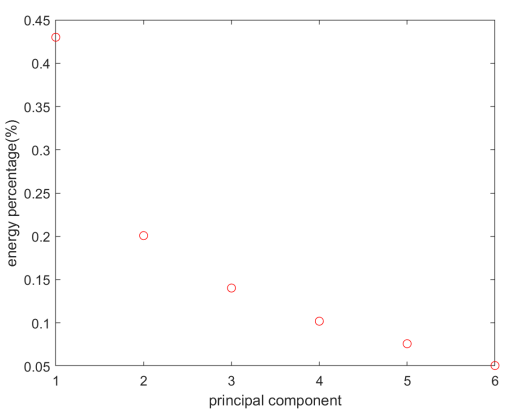


(a) Energy percentage for the horizontal displacement case

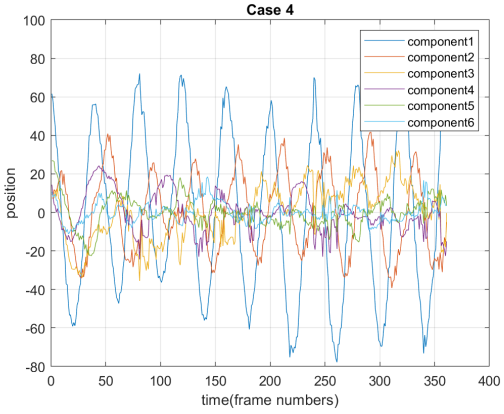


(b) The principle component in the basis of u for the horizontal displacement

Figure 3: Energy percentage and the principle components for the horizontal displacement case (case 3).



(a) Energy percentage for the horizontal displacement and rotation case



(b) The principle component in the basis of u for the horizontal displacement and rotation

Figure 4: Energy percentage and the principle components for the horizontal displacement and rotation (case 4).

3. Case 3: In this case, the can is released off-center so as to produce motion in the x-y plane as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillation which indicates there should be two dominant components. From the sigma plot, we can see that the energy percentage of the second component increases and there is a clear gap between the first two and the rest. From Figure 3(b), we can see the component 1 and component 2 are much larger than the rest components which is multiplied with the first two sigmas in the Figure 3(a). Thus, there are two dominant components in this case.

4. Case 4: In this case, the can is released off-center and rotates so as to produce motion in the x-y plane, rotation as well as the z direction. From the sigma plot, it seems that there is only one dominant direction in this case although the can should move in x-y plane and the z direction. But comparing with Figure 1(a), the energy percentage of the first mode decreases a lot and the second and the third increase. When looking at Figure 4(b), the plot of component 2 and 3 are also clearly greater than the rest components although it is a perfect graph. It indicates that there should be 3 dominant components although these two are less obvious than the first one.

5 Conclusion

Through the exploration of the PCA method on this problem, I found PCA could determine the dominant components and remove the redundant data. I constructed the matrix with six measurement types of three cameras for each case. I then used the SVD decomposition as the main tool to do the analysis. I plotted the energy percentage for each case to determine the number of dominant components. In ideal case, it is clear that there is only one dominant component. For the noisy case, it is still one dominant direction but the energy percentage decreases significantly due to the noisy data extracted from the shaking cameras. In the third case, the energy percentage of the second mode is really close to the first one because there is both a pendulum motion and a simple harmonic oscillation. In the last case, which has an additional rotation, there seems to be only one dominant direction. But when we look at the plot of principal components

(Figure 4(b)), the second and third components also have some clear patterns although they are not as obvious as the first one.

References

[1] J. Nathan Kutz. *AMATH 582 Computation Methods for Data Analysis*. 2010.

6 Appendix A

1. X1 = double(rgb2gray(vidFrames1_1(:,:,:,j))). "rgb2gray" can convert RGB image or colormap to grayscale. Here, "double" will change the data from uint8 to double-precision.

2. [V, I] = max(X1(:)). It will return the max value and index in the array. Here X1(:) will convert the matrix into a column vector.

3. [aaa, bbb] = find(X1 >= V * 11 / 12). It will return two vectors of row and column subscripts of each nonzero element in the matrix X1 which corresponding value is greater or equal than $\frac{11}{12}$ * V.

4. mn = mean(X, 2). It will return the mean along dimension "dim". Here X is a matrix and dim = 2, so it will return a column vector containing the mean of each row.

5. A = repmat(mn,1,n). This command will repeat copies of array mn. Since mn is 6×1 which is in 2 dimension, 1 and n indicates how mn will be arranged in each dimension. Thus, it will generate a matrix with size $6 \times$ n.

6. [u, s, v] = svd(X, 'econ'). This will produce an economy-size singular value decomposition of m $\times$ n matrix X. The u, s, and v represents the U, Σ and V. By Equation (1), we know X = U * Σ * V'. Since m ¡ n, then only the first m columns of v are computed, so the size of v is n $\times$ m. And the size of u and s are both m $\times$ m.

7 Appendix B

```
%% normal
load cam1_1;
load cam2_1;
load cam3_1;
load cam1_2;
load cam2_2;
load cam3_2;
load cam1_3;
load cam2_3;
load cam3_3;
load cam1_4;
load cam2_4;
load cam3_4;
%%
numFrames1 = size(vidFrames1_1, 4);
numFrames2 = size(vidFrames2_1, 4);
numFrames3 = size(vidFrames3_1, 4);
%%
y_1 = [];
x_1 = [];
for j=1:numFrames1
    X1 = double(rgb2gray(vidFrames1_1(:,:,:,j)));
    X1(:, 1:300) = 0; X1(:, 400:end) = 0; X1(1:200, :) = 0; ...
        X1(400:end, :)=0;
    [V, I] = max(X1(:));
    [aaa, bbb] = find(X1 ≥ V * 11 / 12);
    y_1(j) = mean(aaa);
    x_1(j) = mean(bbb);
end
[v, i] = max(y_1(1:50));
x_1 = x_1(30:end);
y_1 = y_1(30:end);
%%
y_2 = [];
x_2 = [];
for j=1:numFrames2
    X2 = double(rgb2gray(vidFrames2_1(:,:,:,j)));
    X2(:, 1:220) = 0; X2(:, 350:end) = 0; X2(1:100, :) = 0; ...
        X2(350:end, :)=0;
    [V, I] = max(X2(:));
    [aaa, bbb] = find(X2 ≥ V * 11 / 12);
    y_2(j) = mean(aaa);
    x_2(j) = mean(bbb);
end
[v, i] = max(y_2(1:41));
x_2 = x_2(i:end);
y_2 = y_2(i:end);
%%
x_3 = [];
y_3 = [];
for j=1:numFrames3
    X3 = double(rgb2gray(vidFrames3_1(:,:,:,j)));
    X3(1:230, :) = 0; X3(350:end, :) = 0; X3(:, 1:200) = 0; X3(:, ...
        480:end)=0;
```

```
    [V, I] = max(X3(:));
    [aaa, bbb] = find(X3 ≥ 11 / 12 * V);
    y_3(j) = mean(aaa);
    x_3(j) = mean(bbb);
end

[v, i] = max(y_3(1:41));
x_3 = x_3(i:end);
y_3 = y_3(i:end);
%%
l = min([length(y_1), length(y_2), length(x_3)]);
X = [x_1(1:l); y_1(1:l); x_2(1:l); y_2(1:l); x_3(1:l); y_3(1:l)];
[m, n] = size(X);
mn = mean(X, 2);
X=X-repmat(mn,1,n); % subtract mean --- put in the same scale
[u, s, v] = svd(X, 'econ');
figure(1)
plot(diag(s)./sum(diag(s)), 'ro')
xlabel('Principal component'); ylabel('energy percentage(%)');
v = v*s;
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(frame numbers)')
ylabel('position')
title('Case 1')
legend('component1', 'component2', 'component3', 'component4', ...
    'component5', 'component6')
%% noise
[m1, n1] = size(vidFrames1_2(:,:,1,1));
[m2, n2] = size(vidFrames2_2(:,:,1,1));
[m3, n3] = size(vidFrames3_2(:,:,1,1));
numFrames1 = size(vidFrames1_2, 4);
numFrames2 = size(vidFrames2_2, 4);
numFrames3 = size(vidFrames3_2, 4);
%%
y_21 = [];
x_21 = [];
for j=1:numFrames1
    X1 = double(rgb2gray(vidFrames1_2(:,:,:,j)));
    X1(:, 1:300) = 0; X1(:, 400:end) = 0; X1(1:200, :) = 0; ...
        X1(400:end, :)=0;
    [V, I] = max(X1(:));
    [aaa, bbb] = find(X1 ≥ V * 11 / 12);
    y_21(j) = mean(aaa);
    x_21(j) = mean(bbb);
end
[v,i] = max(y_21(1:30));
x_21 = x_21(i:end);
y_21 = y_21(i:end);
%%
y_22 = [];
x_22 = [];
for j=1:numFrames2
```

```
    X2 = double(rgb2gray(vidFrames2_2(:,:,:,j)));
    X2(:, 1:200) = 0; X2(:, 400:end) = 0; X2(1:50, :) = 0; ...
        X2(370:end, :)=0;
    [V, I] = max(X2(:));
    [aaa, bbb] = find(X2 ≥ V * 11/12);
    y_22(j) = mean(aaa);
    x_22(j) = mean(bbb);
end
[v,i] = max(y_22(1:30));
x_22 = x_22(i:end);
y_22 = y_22(i:end);
%%
x_23 = [];
y_23 = [];
for j=1:numFrames3
    X3 = double(rgb2gray(vidFrames3_2(:,:,:,j)));
    X3(1:180, :) = 0; X3(320:end, :) = 0; X3(:, 1:250) = 0; X3(:, ...
        480:end)=0;
    [V, I] = max(X3(:));
    [aaa, bbb] = find(X3 ≥ 5 / 6 * V);
    x_23(j) = mean(bbb);
    y_23(j) = mean(aaa);
end
[v,i] = max(x_23(1:40));
x_23 = x_23(i:end);
y_23 = y_23(i:end);
%%
l = min([length(y_21), length(y_22), length(x_23)]);
X = [x_21(1:l); y_21(1:l);x_22(1:l); y_22(1:l); x_23(1:l); y_23(1:l)];
[m, n] = size(X);
mn = mean(X, 2);
X=X-repmat(mn,1,n); % subtract mean
[u, s, v] = svd(X, 'econ');
figure(1)
plot(diag(s) ./ sum(diag(s)), 'o');
xlabel('principal component'); ylabel('energy percentage(%)');
v = v*s;
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(frame numbers)')
ylabel('position')
title('Case 2')
grid
legend('component1', 'component2', 'component3', 'component4', ...
    'component5', 'component6')
%% horizon
[m1, n1] = size(vidFrames1_3(:,:,1,1));
[m2, n2] = size(vidFrames2_3(:,:,1,1));
[m3, n3] = size(vidFrames3_3(:,:,1,1));
numFrames1 = size(vidFrames1_3, 4);
numFrames2 = size(vidFrames2_3, 4);
numFrames3 = size(vidFrames3_3, 4);
%%
```

```
y_31 = [];
x_31 = [];
for j=1:numFrames1
    X1 = double(rgb2gray(vidFrames1_3(:,:,:,j)));
    X1(:, 1:250) = 0; X1(:, 400:end) = 0; X1(1:200, :) = 0; ...
        X1(400:end, :)=0;
    [V, I] = max(X1(:));
    [aaa, bbb] = find(X1 ≥ V * 11 / 12);
    y_31(j) = mean(aaa);
    x_31(j) = mean(bbb);
end
[v,i] = max(y_31(1:30));
x_31 = x_31(i:end);
y_31 = y_31(i:end);
%%
y_32 = [];
x_32 = [];
for j=1:numFrames2
    X2 = double(rgb2gray(vidFrames2_3(:,:,:,j)));
    X2(:, 1:220) = 0; X2(:, 420:end) = 0; X2(1:150, :) = 0; ...
        X2(400:end, :)=0;
    [V, I] = max(X2(:));
    [aaa, bbb] = find(X2 ≥ V * 11 / 12);
    y_32(j) = mean(aaa);
    x_32(j) = mean(bbb);
end
[v,i] = max(y_32(1:50));
x_32 = x_32(i:end);
y_32 = y_32(i:end);
%%
x_33 = [];
y_33 = [];
for j=1:numFrames3
    X3 = double(rgb2gray(vidFrames3_3(:,:,:,j)));
    X3(1:180, :) = 0; X3(350:end, :) = 0; X3(:, 1:150) = 0; X3(:, ...
        480:end)=0;
    [V, I] = max(X3(:));
    [aaa, bbb] = find(X3 ≥ V * 11 / 12);
    x_33(j) = mean(bbb);
    y_33(j) = mean(aaa);
end
[v,i] = max(x_33(1:30));
x_33 = x_33(i:end);
y_33 = y_33(i:end);
%%
l = min([length(y_31), length(y_32), length(x_33)]);
X = [x_31(1:l); y_31(1:l);x_32(1:l); y_32(1:l); x_33(1:l); y_33(1:l)];
[m, n] = size(X);
mn = mean(X, 2);
X=X-repmat(mn,1,n); % subtract mean
[u, s, v] = svd(X, 'econ');
figure(1)
plot(diag(s) ./ sum(diag(s)), 'o');
xlabel('principal component'); ylabel('energy percentage(%)');
v = v*s;
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(frame numbers)')
ylabel('position')
title('Case 3')
legend('component1', 'component2', 'component3', 'component4', ...
    'component5', 'component6')
%% rotation
[m1, n1] = size(vidFrames1_4(:,:,1,1));
[m2, n2] = size(vidFrames2_4(:,:,1,1));
[m3, n3] = size(vidFrames3_4(:,:,1,1));
numFrames1 = size(vidFrames1_4, 4);
numFrames2 = size(vidFrames2_4, 4);
numFrames3 = size(vidFrames3_4, 4);
%%
y_41 = [];
x_41 = [];
for j=1:numFrames1
    X1 = double(rgb2gray(vidFrames1_4(:,:,:,j)));
    X1(:, 1:300) = 0; X1(:, 470:end) = 0; X1(1:200, :) = 0; ...
        X1(380:end, :)=0;
    [V, I] = max(X1(:));
    [aaa, bbb] = find(X1 ≥ V * 11 / 12);
    y_41(j) = mean(aaa);
    x_41(j) = mean(bbb);
end
[v,i] = max(y_41(1:41));
x_41 = x_41(i:end);
y_41 = y_41(i:end);
%%
y_42 = [];
x_42 = [];
for j=1:numFrames2
    X2 = double(rgb2gray(vidFrames2_4(:,:,:,j)));
    X2(:, 1:220) = 0; X2(:, 410:end) = 0; X2(1:50, :) = 0; ...
        X2(400:end, :)=0;
    [V, I] = max(X2(:));
    [aaa, bbb] = find(X2 ≥ 11 / 12 * V);
    y_42(j) = mean(aaa);
    x_42(j) = mean(bbb);
end
[v,i] = max(y_42(1:50));
x_42 = x_42(i:end);
y_42 = y_42(i:end);
%%
x_43 = [];
y_43 = [];
for j=1:numFrames3
    X3 = double(rgb2gray(vidFrames3_4(:,:,:,j)));
    X3(1:150, :) = 0; X3(290:end, :) = 0; X3(:, 1:300) = 0; X3(:, ...
        510:end)=0;
    [V, I] = max(X3(:));
    [aaa, bbb] = find(X3 ≥ 11 / 12 * V);
    x_43(j) = mean(bbb);
    y_43(j) = mean(aaa);
end
```

```
[v,i] = max(x_43(1:50));
x_43 = x_43(i:end);
y_43 = y_43(i:end);
%%
l = min([length(y_41), length(y_42), length(x_43)]);
X = [x_41(1:l); y_41(1:l);x_42(1:l); y_42(1:l); x_43(1:l); y_43(1:l)];
[m, n] = size(X);
mn = mean(X, 2);
X=X-repmat(mn,1,n); % subtract mean
[u, s, v] = svd(X, 'econ');
figure(1)
plot(diag(s)./sum(diag(s)), 'ro')
xlabel('principal component'); ylabel('energy percentage(%)');
v = v*s;
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(frame numbers)')
ylabel('position')
title('Case 4')
grid
legend('component1', 'component2', 'component3', 'component4', ...
    'component5', 'component6')
```