Principle Component Analysis on Video

By Dexian Chen

Abstract

The report intends to explore the application of Principle Component Analysis (PCA) on analyzing data. Generally, we extract data from video to gather information on a matrix and apply Singular Value Decomposition (SVD) on it to remove the redundant information and find out the dominant components.

Introduction and Overview

In 4 different scenes of spring mass systems, we have 3 cameras from different perspectives to record the movement of a can with a bright candle on the top. The first scene is an ideal case and its entire motion is in the z direction with a simple harmonic motion. The second scene is a noisy case with some shakings while recording. The third case is simple harmonic oscillations in the z direction and a pendulum motion in the x-y plane. The fourth case is the same as the third case, but it adds some extra rotations in the z direction motion. Given the video information, we can build up a data matrix by tracking the candle in each frame for later PCA. Once we perform SVD on the data matrix, we can figure the most significant principle component and the energy captured. Therefore, we can reduce the system's dimension and remove redundancy.

Theoretical Background

Singular Value Decomposition (Equation 1) is a factorization as the following:

$$A = U\sum V^* \qquad (1)$$

For any matrix A with a size of m × n, U is a m × m unitary matrix, $\sum$ is an m × n matrix whose diagonals are listing from largest to smallest and are non-negative real numbers called singular values denoted as σ, and V is a n × n unitary matrix. To apply PCA, we will perform SVD. The PCA allows us to reduce our data to a lower dimension. To understand how PCA can remove redundancy, we need to look at the covariance of our target matrix. After we build up the matrix, say matrix X, we can get its covariance by performing the calculation:

$$C_X = XX^T/(n-1) \qquad (2)$$

By definition of the covariance, we know that C_X is a square symmetric matrix whose diagonal is variance of X. In Machine Learning and PCA, we'll need the diagonal to be large, which indicates strong variations. However, the large off-diagonal terms indicate redundancies. Therefore, we need to diagonalize it which is what the SVD does.

Algorithm Implementation and Development

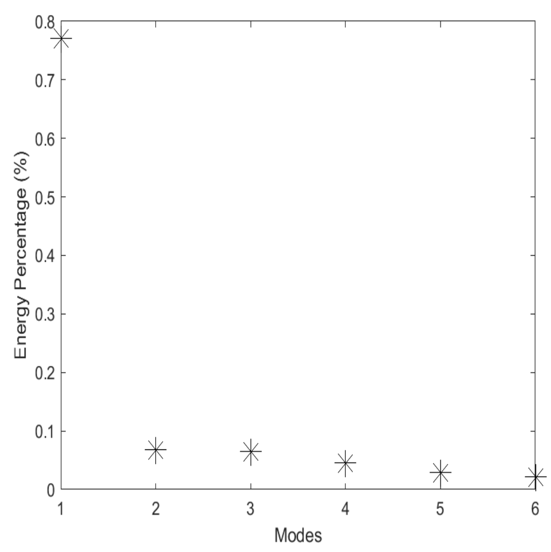
Firstly, I load the data cam1_1, cam2_1, cam3_1, etc. For each case, I do a **for loop** iteration to extract the data for preparation to build up my matrix for SVD. In each iteration, we use **rgb2gray** and **double** to convert the data for analyzation. Then I use **imshow** to determine the movement range of the can so that I can set apply the filter around that certain area. After several tries to find the range of can movement in the coordinates, I set everything outside of that area to be 0, which is applying a Shannon step function filter. Because it's in gray scale, 0 shows black and 1 shows white colors. I find the maximum with **max** command, and use **find** to locate the points of each frame which is greater or equal to a threshold a little bit smaller than the maximum. Once I store the average of all those data, I'm almost finished preparing for the data. One difficulty here is that each video has different time length so I need to enforce them to be the same length when I analyze. I tackle this by cutting all of the frames before the first maximum value. **[Max, Index] = max(input)** will allow me to find the index of maximum in the first 30, 40, or 50 frames. I then enforce the three set of data from three camera to be the same length as the shortest video, with the command **min**. Finally, I build up my matrix for analysis by subtracting the mean of each row. The perfect command **repmat** allows me to do that, and we have the data matrix X for PCA! Just simply use the **svd** command in an **econ** size, and I can get my principle component and energy captured, and then plot them with the **plot** command.

Computational Results

Case one (Ideal): From figure 1, we can tell that the first component captures most of the energy and is the dominant component. Hence, I draw it out in figure 2 to show its fluctuations. Reducing it to one dimension makes sense. Comparing to the first scene which shows mainly can motion in z direction correspond to one dimension.

Case two (Noisy): From figure 3, we can observe that there are maybe 2 or 3 significant components that captures a lot of energy, especially for the first one. This is very similar to case one. Comparing our result to the video, this makes sense because it's now a noisy case, which explains why we have a seemingly significant component generated by the noise. When we look at figure 5, it's obvious that component 1 fluctuates way stronger than component 2.

Case three (Horizontal Displacement): From figure 5, we can observe that there seems to be 4 dominant components because there are 3 or 4 singular value showing a relatively large values in the figure. After I draw out figure 6 and compare it with what I see in the video. It also makes sense for the video, which shows pendulum motion combined with a simple harmonic motion.



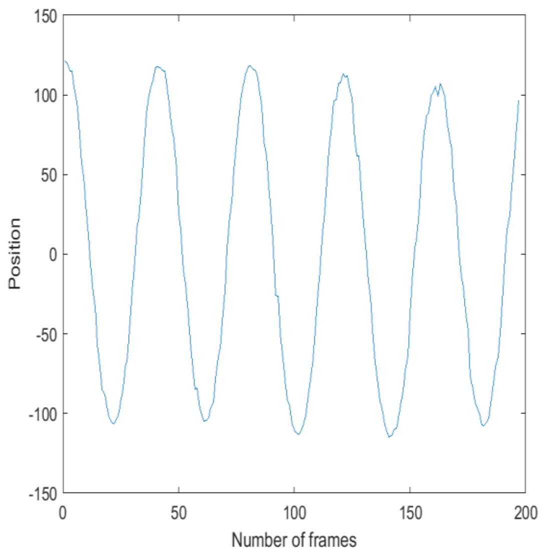
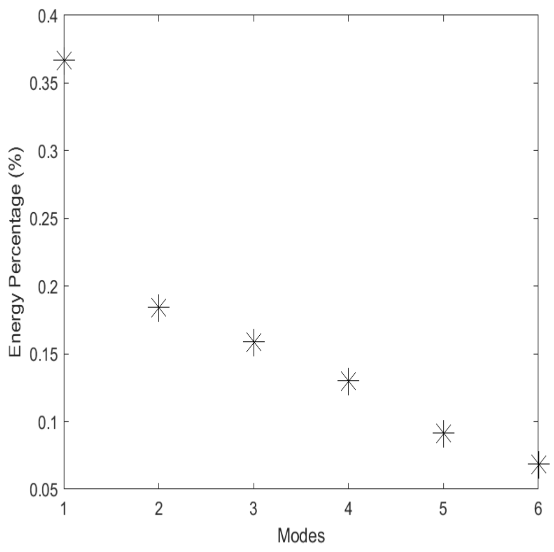


Figure 1 (left): Energy Captured in case one

Figure 2 (right): Dominant Principle Component in case one



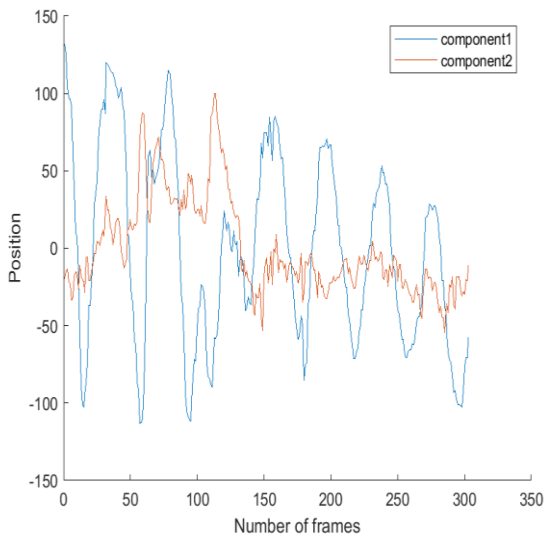
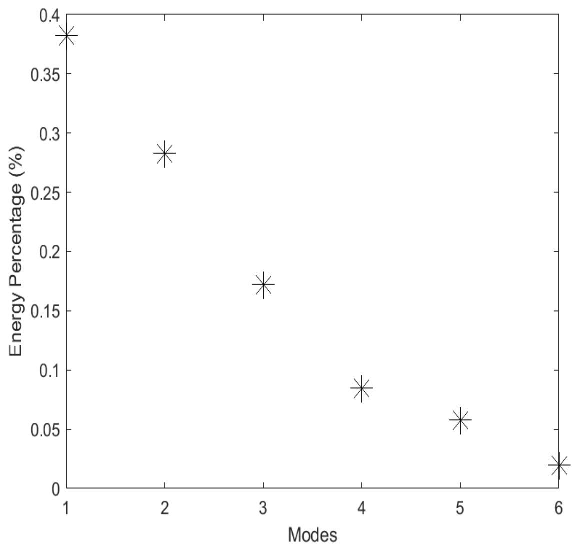


Figure 3 (left): Energy Captured in case two

Figure 4 (right): Dominant Principle Component in case two



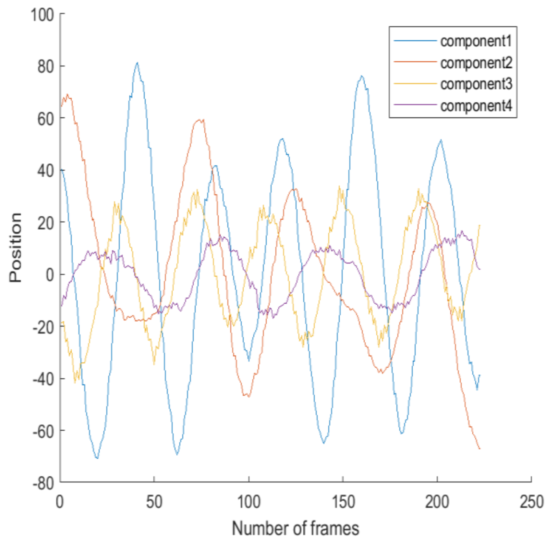
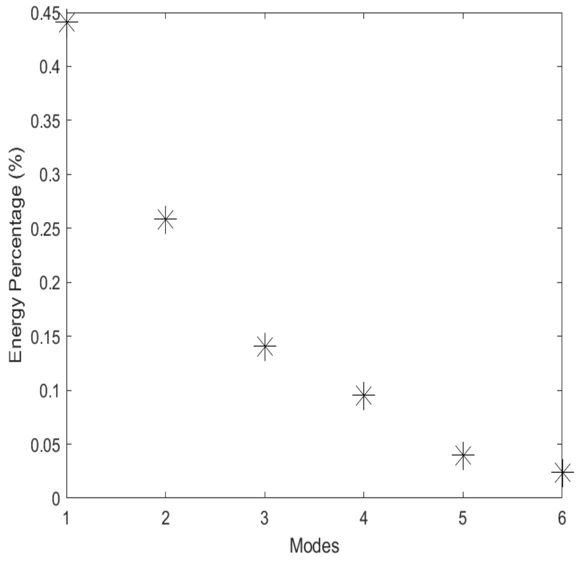


Figure 5 (left): Energy Captured in case three

Figure 6 (right): Dominant Principle Component in case three



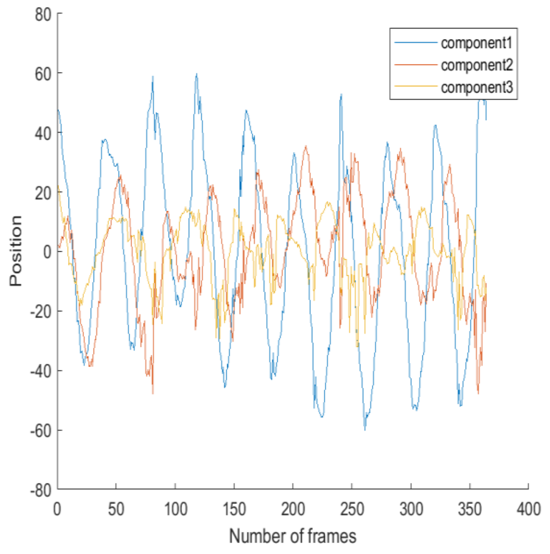


Figure 7 (left): Energy Captured in case 4

Figure 8 (right): Dominant Principle Component in case 4

Case four (Horizontal Displacement and Rotation): From figure 7, I see three relatively large energy capture, so I draw out the 3 significant principle components. This correspond to the video showing oscillatory motion and rotation.

Summary and Conclusions

In conclusion, by performing PCA, we can keep track of how many energies captured in different scenes and figure out the significant orthogonal principle components with strong fluctuations. This will allow us to identify the redundancy and important component in analysis. Furthermore, PCA is very useful in multidimension analysis, even though determining the significance of a component can be vague by just look at its energy quantity. One of SVD's application is PCA, which means not much restriction on PCA for any matrix can perform SVD.

Appendix A

mean(input,2): return the each row's mean in the input matrix

rgb2gray(input): convert a 3 D matrix into a 2 D matrix with 0 corresponding to black and 1 corresponding to white.

double(X): convert the input unit into double precision

[M I] = max(input): find the maximum value M of the input matrix and the maximum index I

find(X >= 5): return all value in matrix X that is greater or equal to 5

[u,s,v] = svd(input): perform svd decomposition on the input matrix.

Appendix B

```
%% Homework 3
close all; clear all; clc;

%%
% Test 1
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
% Play Video of test 1
% implay(vidFrames1_1)
% implay(vidFrames2_1)
% implay(vidFrames3_1)
 Frame_num_11 = size(vidFrames1_1,4);
 Frame_num_21 = size(vidFrames2_1,4);
 Frame_num_31 = size(vidFrames3_1,4);

 % create a matrix to store each frame's can coordinate in (x,y)
x11 = zeros(1,Frame_num_11);
y11 = zeros(1,Frame_num_11);
for j=1:Frame_num_11
```

```
    frame = double(rgb2gray(vidFrames1_1(:,:,:,j)));
    frame(:, [1:300 400:end]) = 0;
    frame([1:200 400:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y11(j) = mean(x_index);
    x11(j) = mean(y_index);
end
[M, I] = max(y11(1:50));
x11 = x11(30:end);
y11 = y11(30:end);


x21 = zeros(1,Frame_num_21);
y21 = zeros(1,Frame_num_21);
for j=1:Frame_num_21
    frame = double(rgb2gray(vidFrames2_1(:,:,:,j)));
    frame(:, [1:210 350:end]) = 0;
    frame([1:100 360:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y21(j) = mean(x_index);
    x21(j) = mean(y_index);
end
[M, I] = max(y21(1:41));
x21 = x21(I:end);
y21 = y21(I:end);


x31 = zeros(1,Frame_num_31);
y31 = zeros(1,Frame_num_31);
for j=1:Frame_num_31
    frame = double(rgb2gray(vidFrames3_1(:,:,:,j)));
    frame(:, [1:200 490:end]) = 0;
    frame([1:220 350:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y31(j) = mean(x_index);
    x31(j) = mean(y_index);
end
[M, I] = max(y31(1:41));
x31 = x31(I:end);
y31 = y31(I:end);

time = min([length(y11), length(y21), length(y31)]);
X = [x11(1:time); y11(1:time); x21(1:time); y21(1:time); x31(1:time); y31(1:time)];
```

```
mean1 = mean(X,2);
[m, n] = size(X);
x_mean = repmat(mean1,1,n);
X = X - x_mean;
[u,s,v] = svd(X,'econ');
figure(1)
plot(diag(s)./sum(diag(s)), 'k*', 'MarkerSize', 14);
xlabel('Modes');
ylabel('Energy Percentage (%)')
print(gcf,'-dpng','figure 1.png');
figure(2)
v = v*s;
plot(v(:,1));
xlabel('Number of frames');
ylabel('Position');
print(gcf,'-dpng','figure 2.png');

%%
% Test 2
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
% Play Video of test 2
% implay(vidFrames1_2)
% implay(vidFrames2_2)
% implay(vidFrames3_2)
 Frame_num_12 = size(vidFrames1_2,4);
 Frame_num_22 = size(vidFrames2_2,4);
 Frame_num_32 = size(vidFrames3_2,4);
x12 = zeros(1,Frame_num_12);
y12 = zeros(1,Frame_num_12);
for j=1:Frame_num_12
    frame = double(rgb2gray(vidFrames1_2(:,:,:,j)));
    frame(:, [1:300 400:end]) = 0;
    frame([1:200 400:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y12(j) = mean(x_index);
    x12(j) = mean(y_index);
end
[M, I] = max(y12(1:30));
x12 = x12(I:end);
y12 = y12(I:end);

x22 = zeros(1,Frame_num_22);
y22 = zeros(1,Frame_num_22);
```

```
for j=1:Frame_num_22
    frame = double(rgb2gray(vidFrames2_2(:,:,:,j)));
    frame(:, [1:200 400:end]) = 0;
    frame([1:50 380:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y22(j) = mean(x_index);
    x22(j) = mean(y_index);
end
[M, I] = max(y22(1:30));
x22 = x22(I:end);
y22 = y22(I:end);


x32 = zeros(1,Frame_num_32);
y32 = zeros(1,Frame_num_32);
for j=1:Frame_num_32
    frame = double(rgb2gray(vidFrames3_2(:,:,:,j)));
    frame(:, [1:250 490:end]) = 0;
    frame([1:170 330:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y32(j) = mean(x_index);
    x32(j) = mean(y_index);
end
[M, I] = max(y32(1:40));
x32 = x32(I:end);
y32 = y32(I:end);

time = min([length(y12), length(y22), length(y32)]);
X = [x12(1:time); y12(1:time); x22(1:time); y22(1:time); x32(1:time); y32(1:time)];
mean2 = mean(X,2);
[m, n] = size(X);
x_mean = repmat(mean2,1,n);
X = X - x_mean;
[u,s,v] = svd(X,'econ');
figure(3)
plot(diag(s)./sum(diag(s)), 'k*', 'MarkerSize', 14);
xlabel('Modes');
ylabel('Energy Percentage (%)')
print(gcf, '-dpng', 'figure 3.png');
figure(4)
v = v*s;
hold on;
for i = 1:2
    plot(v(:,i));
```

```
end
legend('component1', 'component2', 'location', 'best');
xlabel('Number of frames');
ylabel('Position');
print(gcf, '-dpng', 'figure 4.png');
%%
% Test 3
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
% Play Video of test 3
% implay(vidFrames1_3)
% implay(vidFrames2_3)
 Frame_num_13 = size(vidFrames1_3,4);
 Frame_num_23 = size(vidFrames2_3,4);
 Frame_num_33 = size(vidFrames3_3,4);
x13 = zeros(1,Frame_num_13);
y13 = zeros(1,Frame_num_13);
for j=1:Frame_num_13
    frame = double(rgb2gray(vidFrames1_3(:,:,:,j)));
    frame(:, [1:240 400:end]) = 0;
    frame([1:200 400:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y13(j) = mean(x_index);
    x13(j) = mean(y_index);
end
[M, I] = max(y13(1:30));
x13 = x13(I:end);
y13 = y13(I:end);

x23 = zeros(1,Frame_num_23);
y23 = zeros(1,Frame_num_23);
for j=1:Frame_num_23
    frame = double(rgb2gray(vidFrames2_3(:,:,:,j)));
    frame(:, [1:210 430:end]) = 0;
    frame([1:150 400:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y23(j) = mean(x_index);
    x23(j) = mean(y_index);
end
[M, I] = max(y23(1:50));
x23 = x23(I:end);
y23 = y23(I:end);
```

```
x33 = zeros(1,Frame_num_33);
y33 = zeros(1,Frame_num_33);
for j=1:Frame_num_33
    frame = double(rgb2gray(vidFrames3_3(:,:,:,j)));
    frame(:, [1:150 480:end]) = 0;
    frame([1:170 350:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y33(j) = mean(x_index);
    x33(j) = mean(y_index);
end
[M, I] = max(y33(1:30));
x33 = x33(I:end);
y33 = y33(I:end);

time = min([length(y13), length(y23), length(y33)]);
X = [x13(1:time); y13(1:time); x23(1:time); y23(1:time); x33(1:time); y33(1:time)];
mean3 = mean(X,2);
[m, n] = size(X);
x_mean = repmat(mean3,1,n);
X = X - x_mean;
[u,s,v] = svd(X,'econ');
figure(5)
plot(diag(s)./sum(diag(s)), 'k*', 'MarkerSize', 14);
xlabel('Modes');
ylabel('Energy Percentage (%)')
print(gcf, '-dpng', 'figure 5.png');
figure(6)
v = v*s;
hold on;
for i = 1:4
    plot(v(:,i));
end
legend('component1', 'component2', 'component3', 'component4', 'location', 'best');
xlabel('Number of frames');
ylabel('Position');
print(gcf, '-dpng', 'figure 6.png');
%%
% test 4
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
% Play Video of test 4
% implay(vidFrames1_4)
% implay(vidFrames2_4)
% implay(vidFrames3_4)
 Frame_num_14 = size(vidFrames1_4,4);
 Frame_num_24 = size(vidFrames2_4,4);
 Frame_num_34 = size(vidFrames3_4,4);
x14 = zeros(1,Frame_num_14);
y14 = zeros(1,Frame_num_14);
for j=1:Frame_num_14
    frame = double(rgb2gray(vidFrames1_4(:,:,:,j)));
    frame(:, [1:300 480:end]) = 0;
    frame([1:200 380:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y14(j) = mean(x_index);
    x14(j) = mean(y_index);
end
[M, I] = max(y14(1:41));
x14 = x14(I:end);
y14 = y14(I:end);

x24 = zeros(1,Frame_num_24);
y24 = zeros(1,Frame_num_24);
for j=1:Frame_num_24
    frame = double(rgb2gray(vidFrames2_4(:,:,:,j)));
    frame(:, [1:210 410:end]) = 0;
    frame([1:50 400:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y24(j) = mean(x_index);
    x24(j) = mean(y_index);
end
[M, I] = max(y24(1:50));
x24 = x24(I:end);
y24 = y24(I:end);


x34 = zeros(1,Frame_num_34);
y34 = zeros(1,Frame_num_34);
for j=1:Frame_num_34
    frame = double(rgb2gray(vidFrames3_4(:,:,:,j)));
    frame(:, [1:150 300:end]) = 0;
    frame([1:300 520:end], :) = 0;
    [Max, I] = max(frame(:));
    [x_index, y_index] = find(frame >= Max * 11 / 12);
    y34(j) = mean(x_index);
    x34(j) = mean(y_index);
end
```

```
[M, I] = max(y34(1:50));
x34 = x34(I:end);
y34 = y34(I:end);

time = min([length(y14), length(y24), length(y34)]);
X = [x14(1:time); y14(1:time); x24(1:time); y24(1:time); x34(1:time); y34(1:time)];
mean4 = mean(X,2);
[m, n] = size(X);
x_mean = repmat(mean4,1,n);
X = X - x_mean;
[u,s,v] = svd(X,'econ');
figure(7)
plot(diag(s)./sum(diag(s)), 'k*', 'MarkerSize', 14);
xlabel('Modes');
ylabel('Energy Percentage (%)')
print(gcf, '-dpng', 'figure 7.png');
figure(8)
v = v*s;
hold on;
for i = 1:3
    plot(v(:,i));
end
legend('component1', 'component2', 'component3');
xlabel('Number of frames');
ylabel('Position');
print(gcf, '-dpng', 'figure 8.png');
```