Image Processing with Principal Component Analysis

Casper Hsiao

2020-02-20

Abstract: The objective of this project is to implement principal component analysis on a high-dimensional data of the simple mass-spring system. The mass-spring system underwent four different experiments, and the experimental data were captured by the three different cameras. The principal component analysis was successfully implemented on the experimental data to quantify the low-dimensional dynamics of the mass-spring system.

I. Introduction and Overview

The principal component analysis (PCA) uses the singular-value decomposition (SVD) to decompose a high-dimensional data of a system into a low-dimensional data that consists the descriptive dynamics of the system. To demonstrate the PCA, a mass-spring system is experimented with four different tests, and the data were captured by three different cameras at different perspectives. All four experiments were to observe the oscillation of the mass-spring system. The first experiment focuses on the ideal z-axis oscillation, and the second experiment introduced noises to the cameras while observing the oscillation. The third experiment focuses on the off-axis displacement of the system during oscillation, and the last experiment observes the additional rotational motion with off-axis displacement during the oscillation. The four different experiment would demonstrate the strengths and limitations of the PCA.

II. Theoretical Background

The PCA is based on the theories of linear algebra and matrix decomposition, therefore, the SVD is introduced complementary.

II.A. Singular Value Decomposition (SVD)

The singular value decomposition is a factorization of a matrix into a number of constitutive components. Furthermore, it is a transformation that scales and rotates a given set of vectors. The full SVD is shown in Eq. 1 where $U \in \mathbb{C}^{m\times m}$ is unitary, $V \in \mathbb{C}^{n\times n}$ is unitary

and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal. A unitary matrix means that its conjugate transpose is equal to its inverse. A diagonal matrix means that its off-diagonal entries are zeros.

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

The SVD of matrix A is performed through applying V^* for unitary transformation to preserve the unit sphere, and a scaling transformation to create an ellipse with principal semiaxes determined by Σ. Lastly, a unitary rotational transformation is applied to hyperellipse by U. Furthermore, the theorem states that every matrix $A \in \mathbb{C}^{m\times n}$ has a SVD.

II.B. principal Component Analysis (PCA)

The PCA reduces a complex high-dimensional data to a low-dimensional data that consists the dynamics and behaviors of the system. Therefore, the PCA is utilized to analyze unknown and potentially low-dimensional data.

The covariance matrix determines the similarity between two variables. The covariance matrix is calculated as shown in Eq.2. $X \in \mathbb{R}^{m\times n}$ is the vector storing the sample points of the variables where m represents the number of measurement types and n is the number of sample points.

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

The covariance matrix C_X is a square, symmetric $m \times m$ matrix where the diagonal entries represents the variances of associated measurement type, and the off-diagonal entries are the covariances between different measurement types. A large off-diagonal entry represents redundancy in the measurement types, and small off-diagonal entry represents Independence between the measurement types. Furthermore, large diagonal entry represents strong fluctuations in that variable, which would be the dynamics of interest. As a result, SVD is utilized to diagonalize the covariance matrix to obtain the variances of the variables.

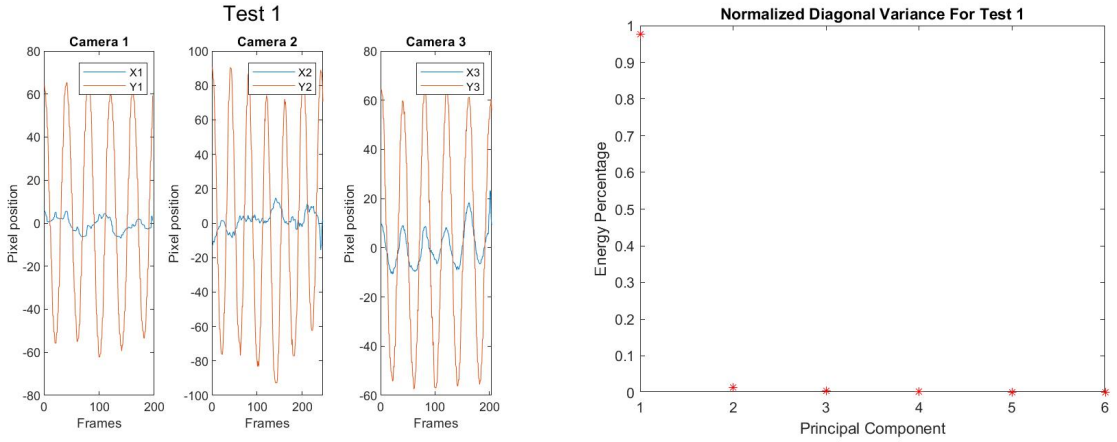
III. Algorithm Implementation and Development

Load all the twelve movie data from the four different test into the MATLAB workspace. Each test is recorded by 3 cameras, therefore, three data for each test. Convert the data files from RGB format to gray scale format. At each frame of the data, apply a filter the around the mass in the spatial domain. Find the maximum value in each frame to locate the position of the mass. Initialize all the data around the frame with the first local maximum to synchronize the data in time. Normalize the data by subtracting the mean value. Reshape

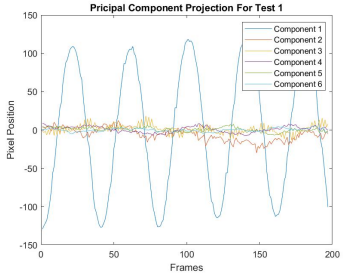
the data to the same size for each test, and concatenate into one single matrix. Lastly, perform SVD to calculate the variance and the projection for each principal component.

IV. Computational Results

The PCA was performed on the movie data where the governing equation is totally unknown. The data was reduced to a lower dimension that described the dynamics of the spring-mass system. The results of the four different tests, ideal, noisy, horizontal displacement and rotation, are shown respectively in Figure 1, 2, 3 and 4. As shown in Figure 1, it is apparent in test one that there is only one dominant principal component. Furthermore, Figure 1a showed the motion location of the mass with high resolution as the data were ideal in the test. Figure 1c also showed the dominance of the dominant component with the projected position.



(a) The x and y direction position of the mass. **(b)** The variances of the principal components.

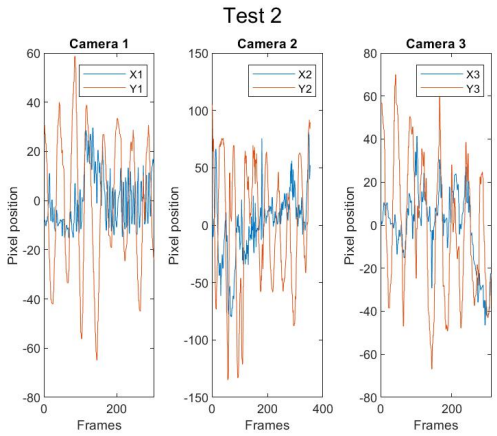


(c) Projection of the principal components.

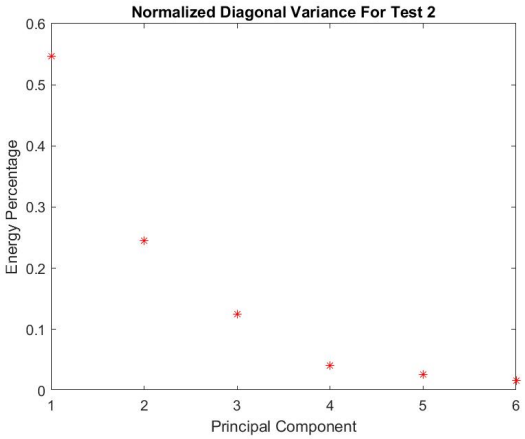
Fig. 1: The principal component analysis of test 1.

Figure 2a showed the unstable location of the mass due to the noisy data from test 2. The variance and projection of the principal components were shown in Figure 2b and 2c. Besides

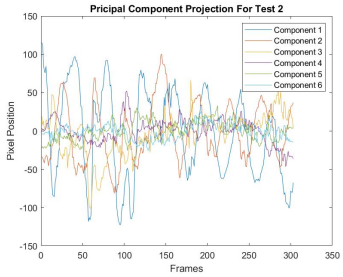
the most dominant component, the second most dominant component is around 25% of the energy. Furthermore, the projection of the components were more distributed as compared to the results from test 1, which represents the vibration from the cameras.



(a) The x and y direction position of the mass.



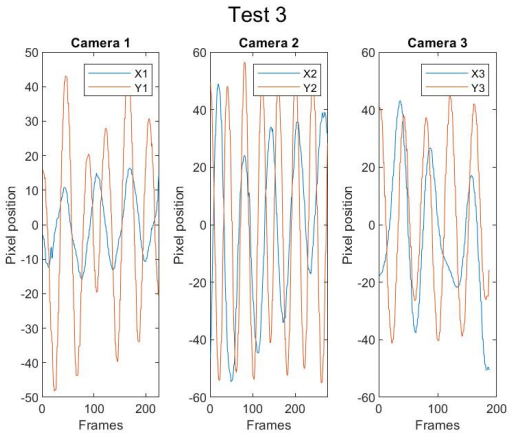
(b) The variances of the principal components.

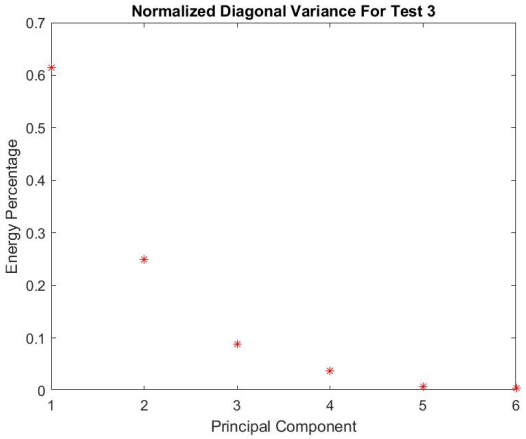


(c) Projection of the principal components.

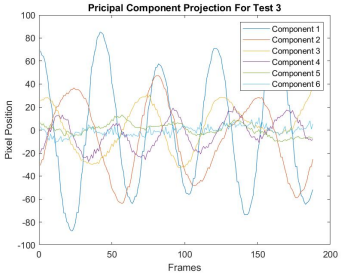
Fig. 2: The principal component analysis of test 2.

Figure 3b showed that the variance of the principal components were relatively distributed as the horizontal displacement was introduced in addition to the vertical displacement. The location of the mass position were significantly smoother than test 2.





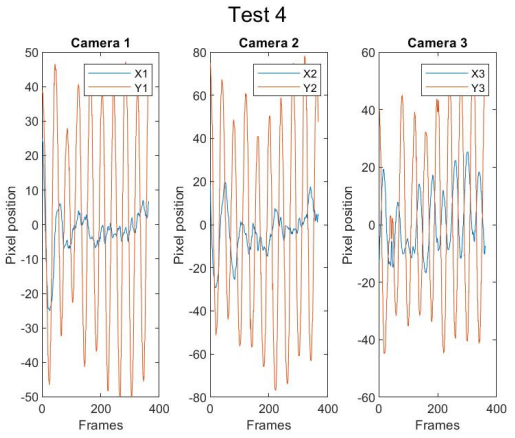
(a) The x and y direction position of the mass. **(b)** The variances of the principal components.

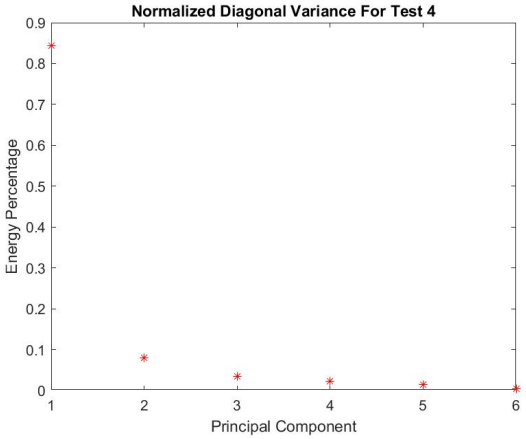


(c) Projection of the principal components.

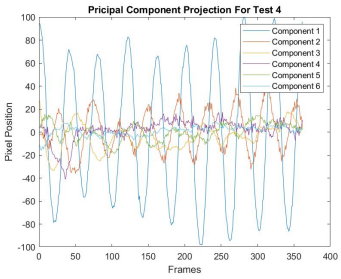
Fig. 3: The principal component analysis of test 3.

Figure 4b showed that the first component dominates the variance energy of the system. Furthermore, this was verified by the component projection as shown in Figure 4c.





(a) The x and y direction position of the mass. **(b)** The variances of the principal components.



(c) Projection of the principal components.

Fig. 4: The principal component analysis of test 4.

V. Summary and Conclusions

The principal component analysis is a efficient and robust method for analyzing unknown system with the assumption of its low dimension property. The results of the tests showed the ability of the PCA in reducing redundancy information from the high dimensions. The singular value decomposition identified the covariance of the system. The weakness of PCA is the assumption of its linearity system. Furthermore, mean and variance values are required for the analysis.

Appendix A

$mean(x)$: Computes the mean value of x.
$svd(x, `econ')$: Performs SVD on x and returns the reduced U and Σ matrix. $zeros(n, n)$: Creates an n-by-n matrix with entries of zero.
$diag(x)$: Extract the diagonal entries of a square matrix x.
$rgb2gray$: Converts the data from RGB format to gray scale format.
$max(x)$: Finds the maximum value in x and the index of the value.
$abs(x)$: Computes the complex magnitude of each element in x.
$sum(x)$: Computes the sum of the values in matrix x.

Appendix B

Contents


```
clc; clear all; close all
```

Load test 1 data

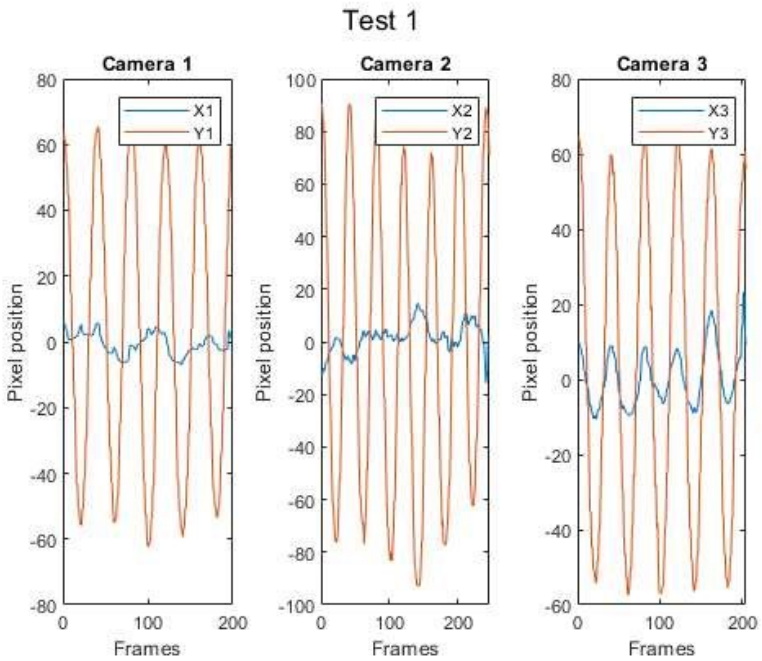
```
load('cam1_1.mat')
% implay(vidFrames1_1)
load('cam2_1.mat')
% implay(vidFrames2_1)
load('cam3_1.mat')
% implay(vidFrames3_1)
```

Obtain the x and y variable data points

```
numFrames11 = size(vidFrames1_1, 4);
numFrames21 = size(vidFrames2_1, 4);
numFrames31 = size(vidFrames3_1, 4);

x11 = zeros(numFrames11, 1);
y11 = x11;
bottom = 430;
top = 190;
left = 300;
right = 390;
for i = 1 : numFrames11
    X11 = double(rgb2gray(vidFrames1_1(:, :, :, i)));
    X11(:, 1:left) = 0;
    X11(:, right:end) = 0;
    X11(1:top, :) = 0;
    X11(bottom:end, :) = 0;
    [M, I] = max(max(X11));
    [row, col] = find(X11 >= 0.9*M);
    x11(i) = mean(col);
    y11(i) = mean(row);
end
x11 = x11 - mean(x11);
y11 = y11 - mean(y11);
[M, I] = max(y11(1:50));
x11 = x11(I:end);
y11 = y11(I:end);


figure(1)
set(gca, 'FontSize', 15)
sgtitle('Test 1');
subplot(1, 3, 1)
```

```
plot(x11);
hold on;
plot(y11);
legend('X1', 'Y1');
title('Camera 1');
xlabel('Frames');
ylabel('Pixel position');

x21 = zeros(numFrames21, 1);
y21 = x21;
bottom = 370;
top = 100;
left = 250;
right = 340;
for i = 1 : numFrames21
    X21 = double(rgb2gray(vidFrames2_1(:, :, :, i)));
    X21(:, 1:left) = 0;
    X21(:, right:end) = 0;
    X21(1:top, :) = 0;
    X21(bottom:end, :) = 0;
    [M, I] = max(max(X21));
    [row, col] = find(X21 >= 0.95*M);
    x21(i) = mean(col);
    y21(i) = mean(row);
end
x21 = x21 - mean(x21);
y21 = y21 - mean(y21);
[M, I] = max(y21(1:50));
x21 = x21(I:end);
y21 = y21(I:end);

subplot(1, 3, 2)
plot(x21);
hold on;
plot(y21);
legend('X2', 'Y2');
title('Camera 2');
xlabel('Frames');
ylabel('Pixel position');

x31 = zeros(numFrames31, 1);
y31 = x31;
bottom = 460;
top = 240;
left = 280;
right = 490;
for i = 1 : numFrames31
    X31 = double(rgb2gray(vidFrames3_1(:, :, :, i)));
    X31(:, 1:left) = 0;
    X31(:, right:end) = 0;
    X31(1:top, :) = 0;
    X31(bottom:end, :) = 0;
    [M, I] = max(max(X31));
    [row, col] = find(X31 >= 0.9*M);
    y31(i) = mean(col);
    x31(i) = mean(row);
end
```

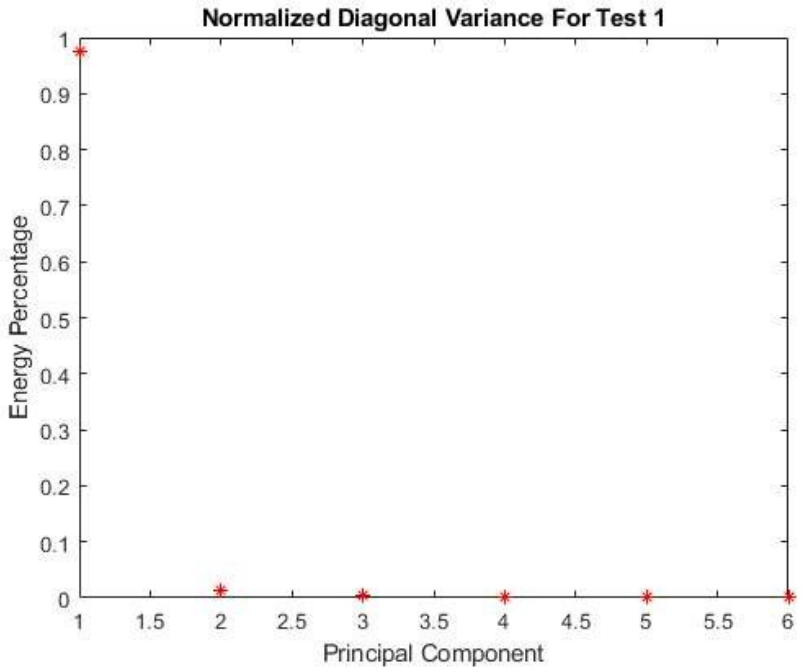
```
x31 = x31 - mean(x31);
y31 = y31 - mean(y31);
[M, I] = max(y31(1:50));
x31 = x31(I:end);
y31 = y31(I:end);

subplot(1, 3, 3)
plot(x31);
hold on;
plot(y31);
legend('X3', 'Y3');
title('Camera 3');
xlabel('Frames');
ylabel('Pixel position');
saveas(gcf, 'Position_Test1.jpg')
```

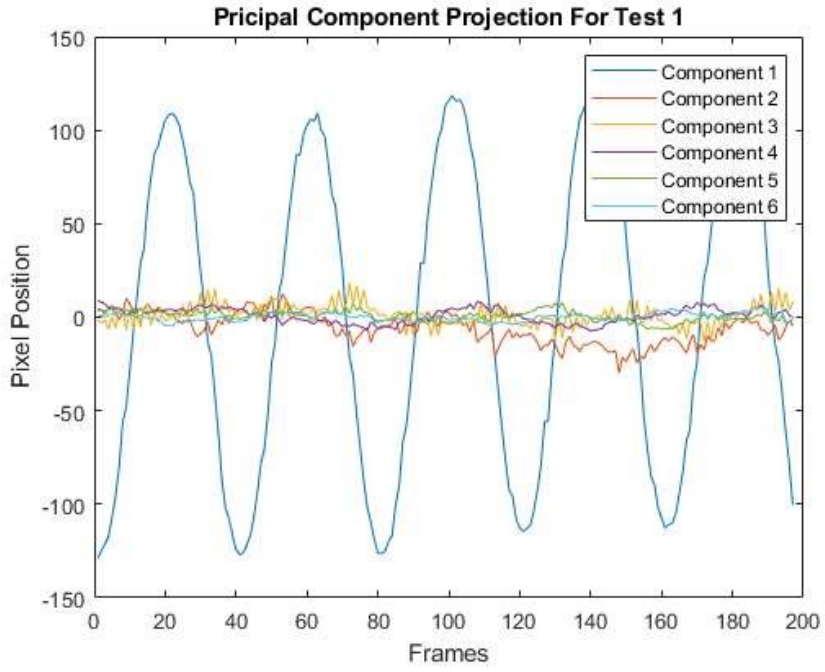


Reshape data

```
n = min([length(y11), length(y21), length(y31)]);
X = [x11(1:n)'; y11(1:n)'; x21(1:n)'; y21(1:n)'; x31(1:n)'; y31(1:n)'];
[U, S, V] = svd(X/sqrt(n - 1), 'econ');
lambda = diag(S).^2;
Y = U'*X;

figure(2)
set(gca, 'FontSize', 10)
lambdaSum = sum(lambda);
plot(lambda./lambdaSum, 'r*');
```

```
title('Normalized Diagonal Variance For Test 1');
xlabel('Principal Component');
ylabel('Energy Percentage');
saveas(gcf, 'Variance_Test1.jpg');

figure(3)
set(gca, 'FontSize', 10)
plot(Y(1, :));
hold on;
plot(Y(2, :));
plot(Y(3, :));
plot(Y(4, :));
plot(Y(5, :));
plot(Y(6, :));
legend('Component 1', 'Component 2', 'Component 3', 'Component 4', 'Component 5', 'Component
6');
title('Pricipal Component Projection For Test 1');
xlabel('Frames');
ylabel('Pixel Position');
saveas(gcf, 'Projection_Test1.jpg');
```





Published with MATLAB® R2019b

Contents


```
clc; clear all; close all
```

Load test 2 data

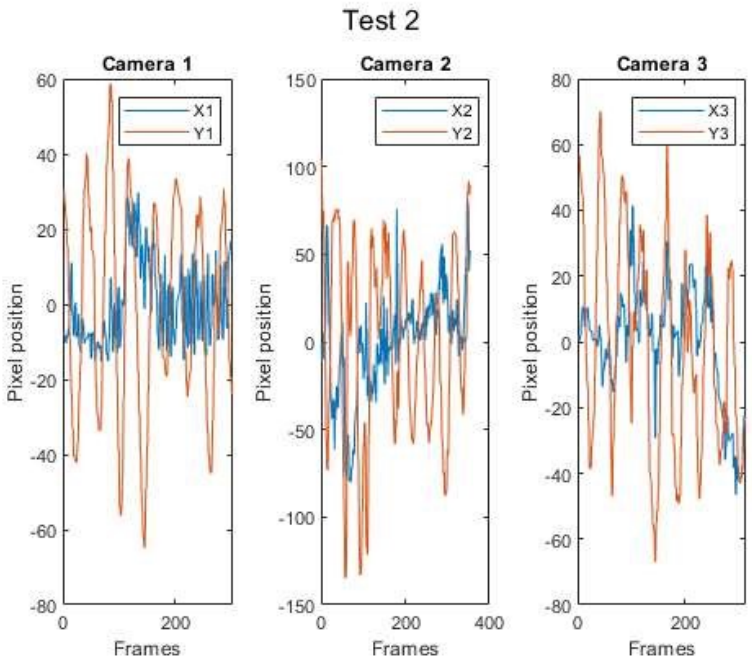
```
load('cam1_2.mat')
% implay(vidFrames1_2)
load('cam2_2.mat')
% implay(vidFrames2_2)
load('cam3_2.mat')
% implay(vidFrames3_2)
```

Obtain the x and y variable data points

```
numFrames12 = size(vidFrames1_2, 4);
numFrames22 = size(vidFrames2_2, 4);
numFrames32 = size(vidFrames3_2, 4);

x12 = zeros(numFrames12, 1);
y12 = x12;
bottom = 410;
top = 200;
left = 330;
right = 420;
for i = 1 : numFrames12
    X12 = double(rgb2gray(vidFrames1_2(:, :, :, i)));
    X12(:, 1:left) = 0;
    X12(:, right:end) = 0;
    X12(1:top, :) = 0;
    X12(bottom:end, :) = 0;
    [M, I] = max(max(X12));
    [row, col] = find(X12 >= 0.9*M);
    x12(i) = mean(col);
    y12(i) = mean(row);
end
x12 = x12 - mean(x12);
y12 = y12 - mean(y12);
[M, I] = max(y12(1:50));
x12 = x12(I:end);
y12 = y12(I:end);


figure(1)
set(gca, 'FontSize', 15)
sgtitle('Test 2');
subplot(1, 3, 1)
```

```
plot(x12);
hold on;
plot(y12);
legend('X1', 'Y1');
title('Camera 1');
xlabel('Frames');
ylabel('Pixel position');

x22 = zeros(numFrames22, 1);
y22 = x22;
bottom = 370;
top = 80;
left = 210;
right = 390;
for i = 1 : numFrames22
    X22 = double(rgb2gray(vidFrames2_2(:, :, :, i)));
    X22(:, 1:left) = 0;
    X22(:, right:end) = 0;
    X22(1:top, :) = 0;
    X22(bottom:end, :) = 0;
    [M, I] = max(max(X22));
    [row, col] = find(X22 >= 0.95*M);
    x22(i) = mean(col);
    y22(i) = mean(row);
end
x22 = x22 - mean(x22);
y22 = y22 - mean(y22);
[M, I] = max(y22(1:50));
x22 = x22(I:end);
y22 = y22(I:end);

subplot(1, 3, 2)
plot(x22);
hold on;
plot(y22);
legend('X2', 'Y2');
title('Camera 2');
xlabel('Frames');
ylabel('Pixel position');

x32 = zeros(numFrames32, 1);
y32 = x32;
bottom = 320;
top = 200;
left = 250;
right = 500;
for i = 1 : numFrames32
    X32 = double(rgb2gray(vidFrames3_2(:, :, :, i)));
    X32(:, 1:left) = 0;
    X32(:, right:end) = 0;
    X32(1:top, :) = 0;
    X32(bottom:end, :) = 0;
    [M, I] = max(max(X32));
    [row, col] = find(X32 >= 0.9*M);
    y32(i) = mean(col);
    x32(i) = mean(row);
end
```

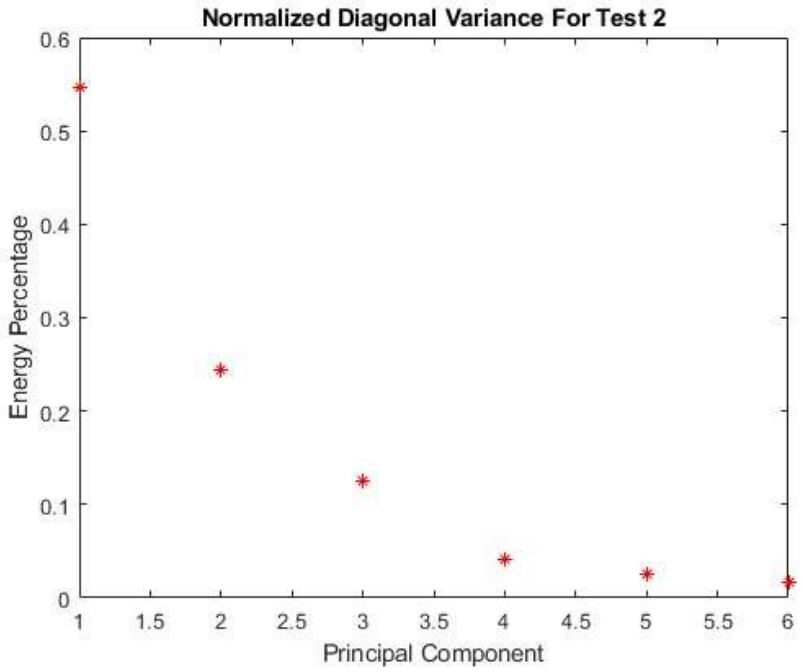
```
x32 = x32 - mean(x32);
y32 = y32 - mean(y32);
[M, I] = max(y32(1:50));
x32 = x32(I:end);
y32 = y32(I:end);

subplot(1, 3, 3)
plot(x32);
hold on;
plot(y32);
legend('X3', 'Y3');
title('Camera 3');
xlabel('Frames');
ylabel('Pixel position');
saveas(gcf, 'Position_Test2.jpg');
```

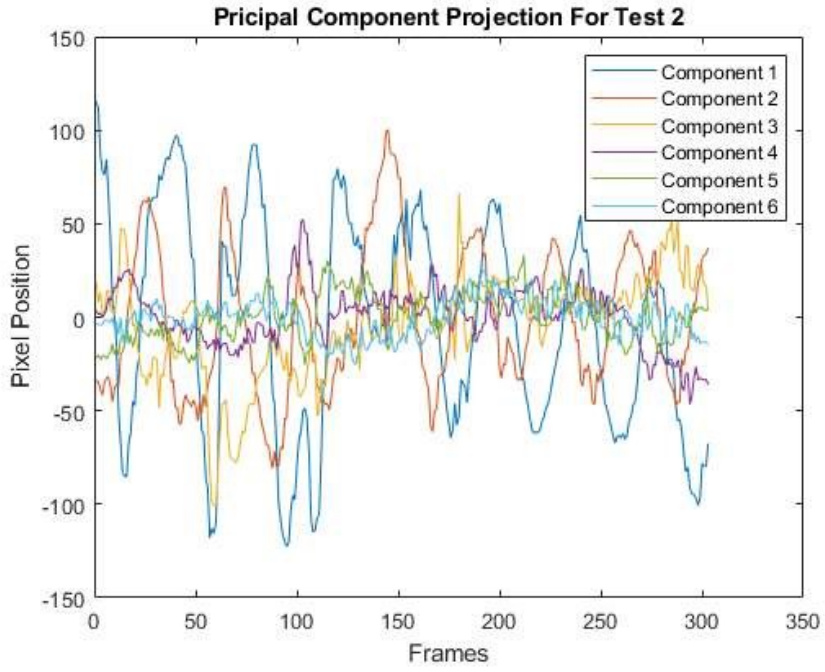


Reshape data

```
n = min([length(y12), length(y22), length(y32)]);
X = [x12(1:n)'; y12(1:n)'; x22(1:n)'; y22(1:n)'; x32(1:n)'; y32(1:n)'];
[U, S, V] = svd(X/sqrt(n - 1), 'econ');
lambda = diag(S).^2;
Y = U'*X;

figure(2)
set(gca, 'FontSize', 10)
lambdaSum = sum(lambda);
plot(lambda./lambdaSum, 'r*');
```

```
title('Normalized Diagonal Variance For Test 2');
xlabel('Principal Component');
ylabel('Energy Percentage');
saveas(gcf, 'Variance_Test2.jpg');

figure(3)
set(gca, 'FontSize', 10)
plot(Y(1, :));
hold on;
plot(Y(2, :));
plot(Y(3, :));
plot(Y(4, :));
plot(Y(5, :));
plot(Y(6, :));
legend('Component 1', 'Component 2', 'Component 3', 'Component 4', 'Component 5', 'Component
6');
title('Pricipal Component Projection For Test 2');
xlabel('Frames');
ylabel('Pixel Position');
saveas(gcf, 'Projection_Test2.jpg');
```





Published with MATLAB® R2019b

Contents


```
clc; clear all; close all
```

Load test 2 data

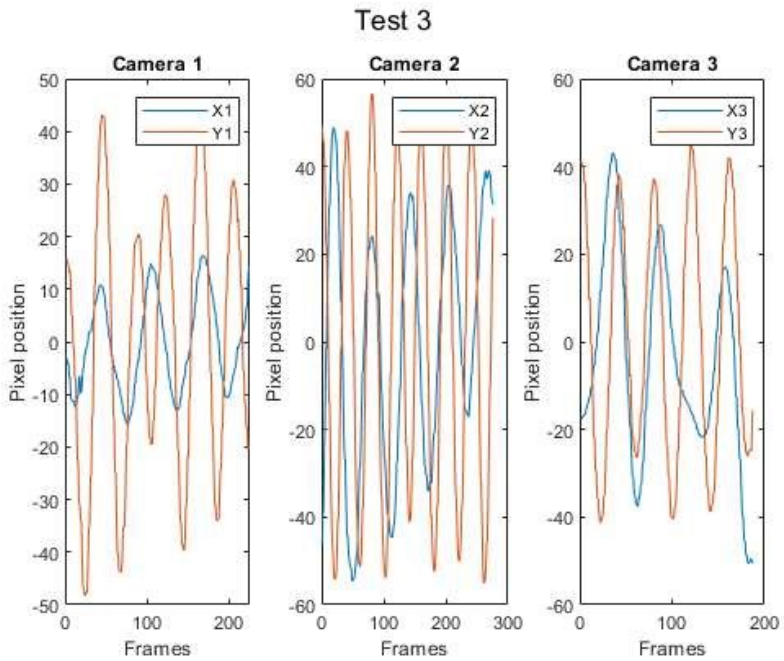
```
load('cam1_3.mat')
% implay(vidFrames1_3)
load('cam2_3.mat')
% implay(vidFrames2_3)
load('cam3_3.mat')
% implay(vidFrames3_3)
```

Obtain the x and y variable data points

```
numFrames13 = size(vidFrames1_3, 4);
numFrames23 = size(vidFrames2_3, 4);
numFrames33 = size(vidFrames3_3, 4);

x13 = zeros(numFrames13, 1);
y13 = x13;
bottom = 410;
top = 190;
left = 240;
right = 420;
for i = 1 : numFrames13
    X13 = double(rgb2gray(vidFrames1_3(:, :, :, i)));
    X13(:, 1:left) = 0;
    X13(:, right:end) = 0;
    X13(1:top, :) = 0;
    X13(bottom:end, :) = 0;
    [M, I] = max(max(X13));
    [row, col] = find(X13 >= 0.9*M);
    x13(i) = mean(col);
    y13(i) = mean(row);
end
x13 = x13 - mean(x13);
y13 = y13 - mean(y13);
[M, I] = max(y13(1:50));
x13 = x13(I:end);
y13 = y13(I:end);


figure(1)
set(gca, 'FontSize', 15)
sgtitle('Test 3');
subplot(1, 3, 1)
```

```
plot(x13);
hold on;
plot(y13);
legend('X1', 'Y1');
title('Camera 1');
xlabel('Frames');
ylabel('Pixel position');

x23 = zeros(numFrames23, 1);
y23 = x23;
bottom = 390;
top = 150;
left = 250;
right = 410;
for i = 1 : numFrames23
    X23 = double(rgb2gray(vidFrames2_3(:, :, :, i)));
    X23(:, 1:left) = 0;
    X23(:, right:end) = 0;
    X23(1:top, :) = 0;
    X23(bottom:end, :) = 0;
    [M, I] = max(max(X23));
    [row, col] = find(X23 >= 0.95*M);
    x23(i) = mean(col);
    y23(i) = mean(row);
end
x23 = x23 - mean(x23);
y23 = y23 - mean(y23);
[M, I] = max(y23(1:50));
x23 = x23(I:end);
y23 = y23(I:end);

subplot(1, 3, 2)
plot(x23);
hold on;
plot(y23);
legend('X2', 'Y2');
title('Camera 2');
xlabel('Frames');
ylabel('Pixel position');

x33 = zeros(numFrames33, 1);
y33 = x33;
bottom = 320;
top = 180;
left = 250;
right = 480;
for i = 1 : numFrames33
    X33 = double(rgb2gray(vidFrames3_3(:, :, :, i)));
    X33(:, 1:left) = 0;
    X33(:, right:end) = 0;
    X33(1:top, :) = 0;
    X33(bottom:end, :) = 0;
    [M, I] = max(max(X33));
    [row, col] = find(X33 >= 0.9*M);
    y33(i) = mean(col);
    x33(i) = mean(row);
end
```

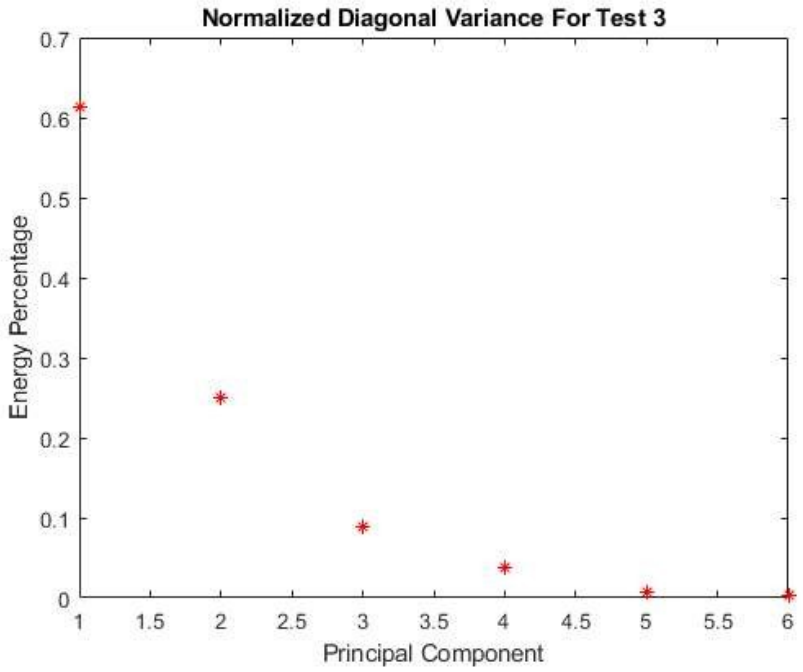
```
x33 = x33 - mean(x33);
y33 = y33 - mean(y33);
[M, I] = max(y33(1:50));
x33 = x33(I:end);
y33 = y33(I:end);

subplot(1, 3, 3)
plot(x33);
hold on;
plot(y33);
legend('X3', 'Y3');
title('Camera 3');
xlabel('Frames');
ylabel('Pixel position');
saveas(gcf, 'Position_Test3.jpg')
```

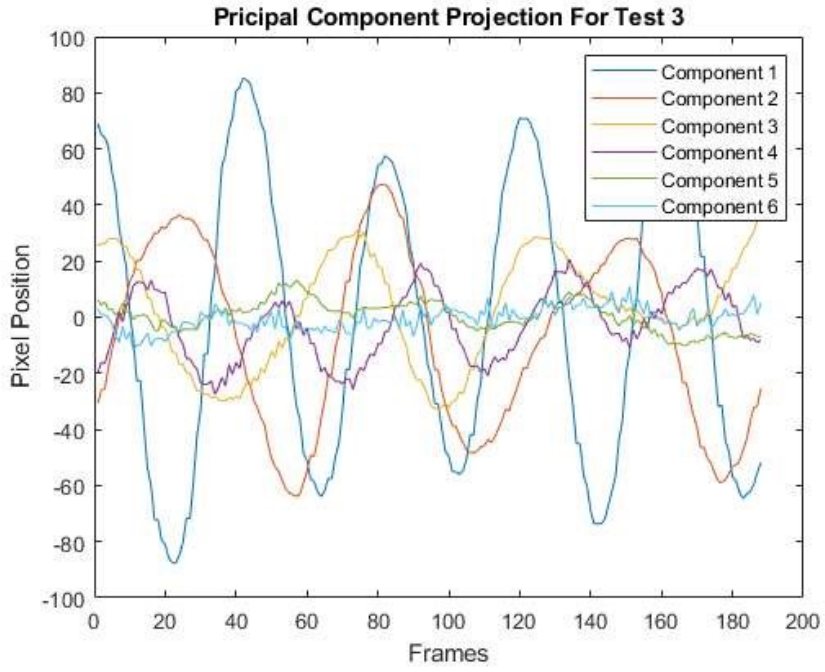


Reshape data

```
n = min([length(y13), length(y23), length(y33)]);
X = [x13(1:n)'; y13(1:n)'; x23(1:n)'; y23(1:n)'; x33(1:n)'; y33(1:n)'];
[U, S, V] = svd(X, 'econ');
lambda = diag(S).^2;
Y = U'*X;

figure(2)
set(gca, 'FontSize', 10)
lambdaSum = sum(lambda);
plot(lambda./lambdaSum, 'r*');
```

```
title('Normalized Diagonal Variance For Test 3');
xlabel('Principal Component');
ylabel('Energy Percentage');
saveas(gcf, 'Variance_Test3.jpg')

figure(3)
set(gca, 'FontSize', 10)
plot(Y(1, :));
hold on;
plot(Y(2, :));
plot(Y(3, :));
plot(Y(4, :));
plot(Y(5, :));
plot(Y(6, :));
legend('Component 1', 'Component 2', 'Component 3', 'Component 4', 'Component 5', 'Component
6');
title('Pricipal Component Projection For Test 3');
xlabel('Frames');
ylabel('Pixel Position');
saveas(gcf, 'Projection_Test3.jpg')
```





Published with MATLAB® R2019b

Contents


```
clc; clear all; close all
```

Load test 2 data

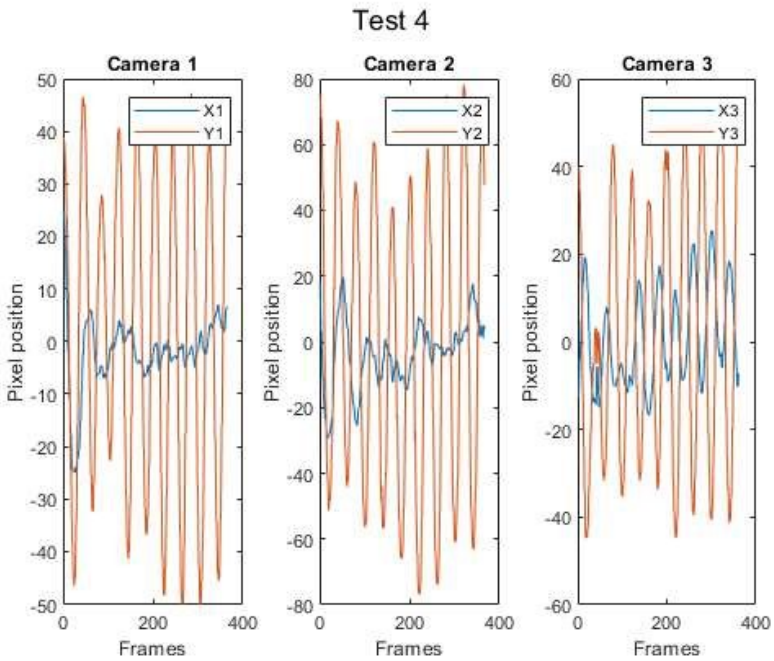
```
load('cam1_4.mat')
% implay(vidFrames1_4)
load('cam2_4.mat')
% implay(vidFrames2_4)
load('cam3_4.mat')
% implay(vidFrames3_4)
```

Obtain the x and y variable data points

```
numFrames14 = size(vidFrames1_4, 4);
numFrames24 = size(vidFrames2_4, 4);
numFrames34 = size(vidFrames3_4, 4);

x14 = zeros(numFrames14, 1);
y14 = x14;
bottom = 410;
top = 190;
left = 240;
right = 420;
for i = 1 : numFrames14
    X14 = double(rgb2gray(vidFrames1_4(:, :, :, i)));
    X14(:, 1:left) = 0;
    X14(:, right:end) = 0;
    X14(1:top, :) = 0;
    X14(bottom:end, :) = 0;
    [M, I] = max(max(X14));
    [row, col] = find(X14 >= 0.9*M);
    x14(i) = mean(col);
    y14(i) = mean(row);
end
x14 = x14 - mean(x14);
y14 = y14 - mean(y14);
[M, I] = max(y14(1:50));
x14 = x14(I:end);
y14 = y14(I:end);


figure(1)
set(gca, 'FontSize', 15)
sgtitle('Test 4');
subplot(1, 3, 1)
```

```
plot(x14);
hold on;
plot(y14);
legend('X1', 'Y1');
title('Camera 1');
xlabel('Frames');
ylabel('Pixel position');

x24 = zeros(numFrames24, 1);
y24 = x24;
bottom = 390;
top = 150;
left = 250;
right = 410;
for i = 1 : numFrames24
    X24 = double(rgb2gray(vidFrames2_4(:, :, :, i)));
    X24(:, 1:left) = 0;
    X24(:, right:end) = 0;
    X24(1:top, :) = 0;
    X24(bottom:end, :) = 0;
    [M, I] = max(max(X24));
    [row, col] = find(X24 >= 0.95*M);
    x24(i) = mean(col);
    y24(i) = mean(row);
end
x24 = x24 - mean(x24);
y24 = y24 - mean(y24);
[M, I] = max(y24(1:50));
x24 = x24(I:end);
y24 = y24(I:end);

subplot(1, 3, 2)
plot(x24);
hold on;
plot(y24);
legend('X2', 'Y2');
title('Camera 2');
xlabel('Frames');
ylabel('Pixel position');

x34 = zeros(numFrames34, 1);
y34 = x34;
bottom = 320;
top = 180;
left = 250;
right = 480;
for i = 1 : numFrames34
    X34 = double(rgb2gray(vidFrames3_4(:, :, :, i)));
    X34(:, 1:left) = 0;
    X34(:, right:end) = 0;
    X34(1:top, :) = 0;
    X34(bottom:end, :) = 0;
    [M, I] = max(max(X34));
    [row, col] = find(X34 >= 0.9*M);
    y34(i) = mean(col);
    x34(i) = mean(row);
end
```

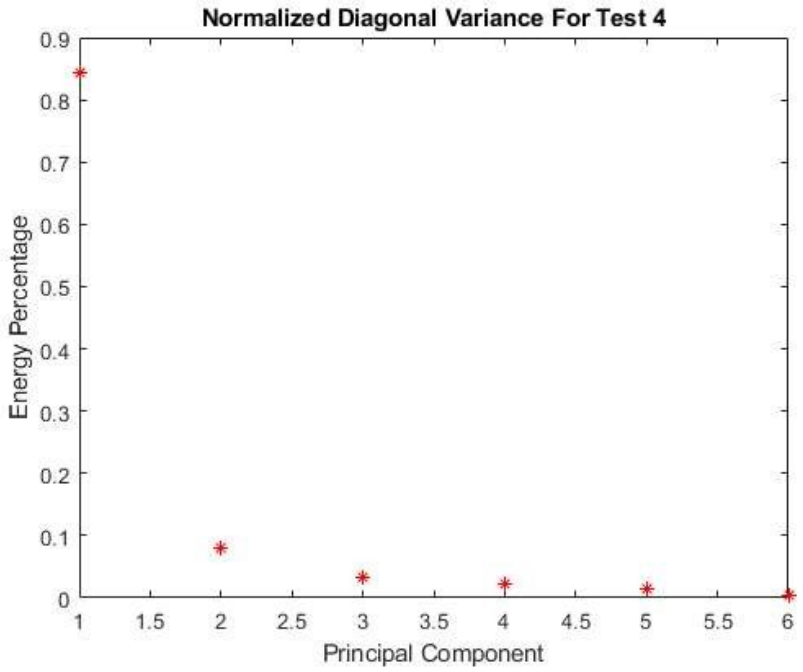
```
x34 = x34 - mean(x34);
y34 = y34 - mean(y34);
[M, I] = max(y34(1:50));
x34 = x34(I:end);
y34 = y34(I:end);

subplot(1, 3, 3)
plot(x34);
hold on;
plot(y34);
legend('X3', 'Y3');
title('Camera 3');
xlabel('Frames');
ylabel('Pixel position');
saveas(gcf, 'Position_Test4.jpg')
```

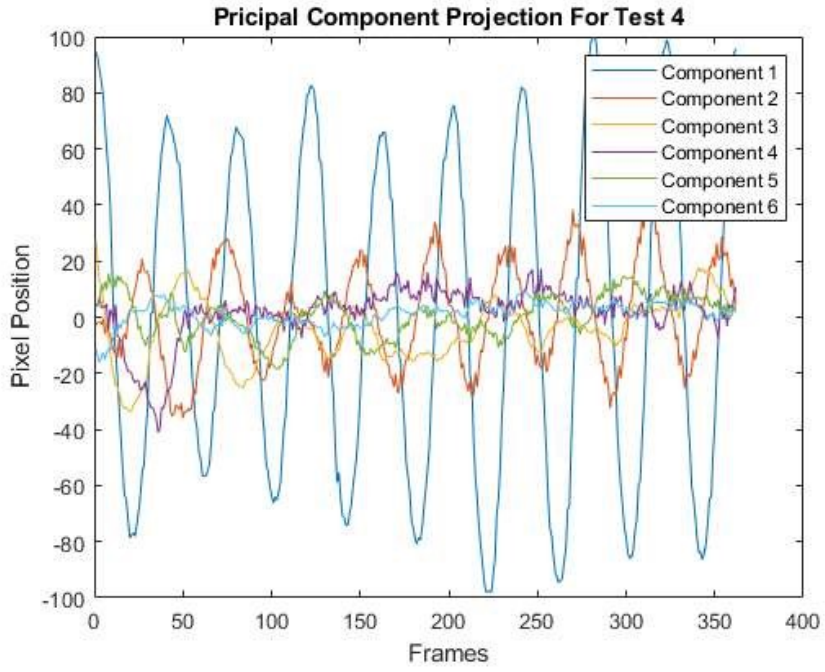


Reshape data

```
n = min([length(y14), length(y24), length(y34)]);
X = [x14(1:n)'; y14(1:n)'; x24(1:n)'; y24(1:n)'; x34(1:n)'; y34(1:n)'];
[U, S, V] = svd(X, 'econ');
lambda = diag(S).^2;
Y = U'*X;

figure(2)
set(gca, 'FontSize', 10)
lambdaSum = sum(lambda);
plot(lambda./lambdaSum, 'r*');
```

```
title('Normalized Diagonal Variance For Test 4');
xlabel('Principal Component');
ylabel('Energy Percentage');
saveas(gcf, 'Variance_Test4.jpg')

figure(3)
set(gca, 'FontSize', 10)
plot(Y(1, :));
hold on;
plot(Y(2, :));
plot(Y(3, :));
plot(Y(4, :));
plot(Y(5, :));
plot(Y(6, :));
legend('Component 1', 'Component 2', 'Component 3', 'Component 4', 'Component 5', 'Component
6');
title('Pricipal Component Projection For Test 4');
xlabel('Frames');
ylabel('Pixel Position');
saveas(gcf, 'Projection_Test4.jpg')
```





Published with MATLAB® R2019b