Principal Component Analysis

AMATH 482 Homework 3

Robin Yan

Feb. 17, 2020

Abstract

Principal component analysis (PCA) is a process to reduce the redundancy in measured data and extract essential information that represent the original data. This homework demonstrates the capability of PCA in data analysis through a series of motion analyses on twelve videos of an oscillating can. The videos fall under four scenarios, each with different levels of motions. The script uses singular value decomposition (SVD) to perform PCA and successfully computes the degree of freedom in each scenario: trial 1 has one degree of freedom, trial 2 has three degree of freedom, trial 3 has two and trial 4 has three degree of freedom.

Section I – Introduction and Overview

The purpose of this homework is to explore the implementation of PCA in data analysis through motion analyses on an oscillating can under four different scenarios. PCA can be implemented with either eigenvalue decomposition for certain types of data or SVD which applies to all data types. This project uses SVD due to the matrix size requirement of eigenvalue decomposition.

The videos being analyzed are 640 by 480 resolution recordings of a paint can attached to a spring and are taken by three phone cameras in four trials: trial one is an oscillating can with only vertical motion, trial two is the same motion with noise introduced by camera shaking, trial three is the can with both vertical and horizontal motion introduced by releasing off center and trial four is the can with vertical, horizontal and rotational motion. The can has a flashlight mounted on the top for easier tracking.

Upon importing, the script converts the videos into grey scale and analyzes them frame by frame through highlight tracking. The tracked pixels are averaged to calculate the relative X and Y location of the can at each frame. These location data are combined to summarize the motion of the can from the perspective of each camera. The motion data from all three cameras are then aligned and stacked for SVD and PCA. The covariance matrix from PCA provides insight into the degree of freedom in each system.

Section II – Theoretical Background

Singular Value Decomposition

Standard singular value decomposition of an m by n matrix X takes the form of:

$$X = U * \Sigma * V^T \quad (E.1)$$

In which U and V are both unitary square matrices with dimensions m by m and n by n, and Σ is an m by n matrix consisting of a diagonal part of m by m filled with silent rows/columns to reach m by n dimension. By convention, m is the number of variables and n is the number of observations. This process is applicable to any matrix X regardless of dimensions.

Covariance Matrix

The covariance matrix of a series of observations X is calculate as:

$$C_X = \frac{1}{n-1} XX^T \quad (E.2)$$

In which n is the number of observations, X is the combined matrix of observations with dimensions m by n and m is the number of variables, X_T is the transpose of matrix X. The diagonal of matrix C_X are the variance of all variables, and off-diagonal terms are the covariance between variables. Higher covariance implies higher

dependency between variables. To reduce the redundancy in data, it is necessary to diagonalize the covariance matrix.

Principal Component Analysis

Based on results from SVD, the basis of original matrix X can be modified by:

$$Y = U^* * X \quad (E.3)$$

In which U^* is the transpose of matrix U from SVD, for real numbers. Y is the transformed data with new basis. Then the covariance matrix of Y is:

$$C_Y = \frac{1}{n-1} YY^T = \frac{1}{n-1}(U^*X)(X^TU) = \frac{1}{n-1} U^TU\Sigma V^T V\Sigma U^T U = \frac{1}{n-1}\Sigma^2 \quad (E.4)$$

The new covariance matrix is diagonal; hence the new matrix has reduced redundancy between variables. The magnitude of the variance of on the diagonal predicts the possible degree of freedom among the variables.

Modal Power

The power of the first k modes in PCA describes how well the first k modes represent the original data, and is quantified as:

$$P_k = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{m} \sigma_i^2} \quad (E.5)$$

Section III – Algorithm Implementation and Development

Motion Tracking

The color videos are converted into grey scale frames in MATLAB. Because of the high brightness of the can and flashlight, this script tracks the location of the can by tracking the pixels with intensity above a certain threshold. The exact values of the threshold are set individually after iterations on each video and listed below in Table 1.

Table 1 Brightness threshold for each video (max brightness is 255)

	Camera 1	Camera 2	Camera 3
Trial 1	250	250	245
Trial 2	245	245	240
Trial 3	250	250	240
Trial 4	245	250	235

Since there are other area in the videos with high brightness, the script avoids certain "stay out zone" based on manual identification. The list of stay out zones in X and Y coordinates are summarized below in Table 2.

Table 2 Stay out zone for each video in X and Y coordinates (pixels)

		Y (horizontal up to 640)	X (vertical up to 480)
Trial 1	Camera 1	1-300, 400-640	1-200
	Camera 2	0	0
	Camera 3	1-200	0

Trial 2	Camera 1	1-300, 400-640	1-200
	Camera 2	0	0
	Camera 3	1-200	1-200
Trial 3	Camera 1	1-300, 400-640	1-200
	Camera 2	0	0
	Camera 3	1-200	0
Trial 4	Camera 1	1-300, 400-640	1-200
	Camera 2	0	1-150
	Camera 3	1-200	0

Because the last trial requires rotation tracking, which is highly reliant on accurately tracking the flashlight, an additional filter is implemented on camera 1 and 2 to exclude all tracked points five pixels below the top point.

Alignment and Truncation

The data are synchronized by aligning the first point with max vertical position, corresponding to Y coordinate in camera 1 and 2 and X coordinate in camera 3. The tails of the data are truncated per camera with lowest observations. The aligned and truncated data are then stacked to form a 6 by n matrix, in which n is the number of observations during the trial.

SVD and PCA

The data matrix is normalized by subtracting the average of each row. SVD is carried out with MATLAB's built-in function svd, which outputs three matrices U, Σ and V. PCA is implemented by first converting the basis of matrix X by E.3. The new covariance matrix is calculated using MATLAB's built-in function cov, which outputs a diagonal matrix corresponding to the variance of the new matrix with reduced redundancy.

Section IV – Computational Results and Discussions

Tracked Location of Paint Can

Based on the brightness threshold and stay out zones mentioned above, the tracked pixels on the paint can from one frame is illustrated below in Fig. 1.

Fig. 1 Grey scale image of frame 1, camera 1 during Trial 1 with tracked pixels in black

The tracked area on the can is painted black. Throughout all videos, the thresholds are set to ensure at least one area, either the sidewall of the can or the flashlight, is tracked. The script calculates the centroid of the tracked area to determine the motion of the can. The motion data from all three cameras during the four trials are summarized below in Fig. 2-5.

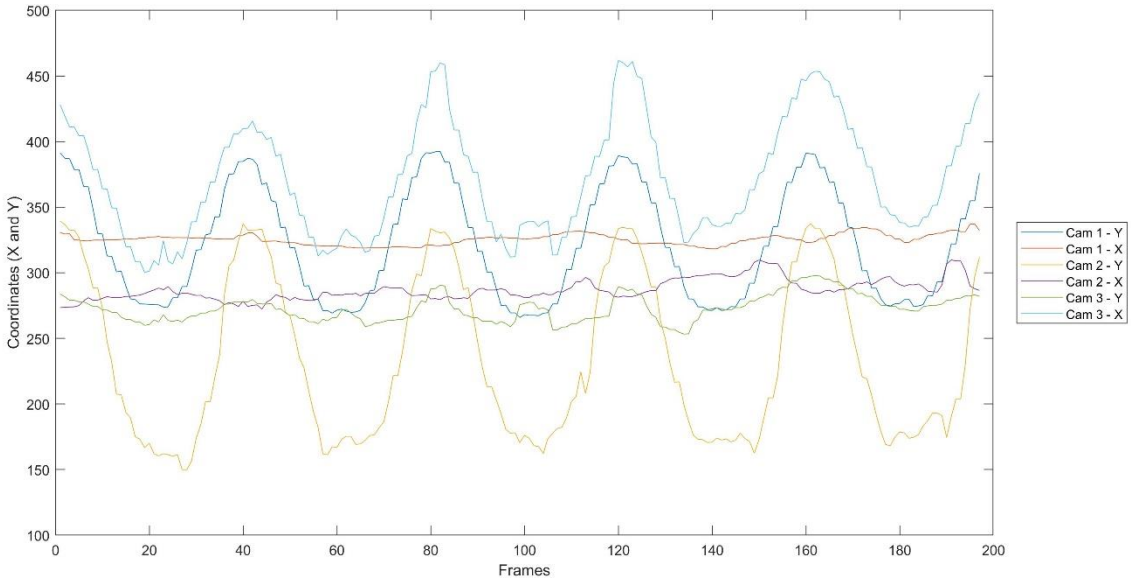

Fig. 2 Motion of paint can during trial 1: vertical oscillation

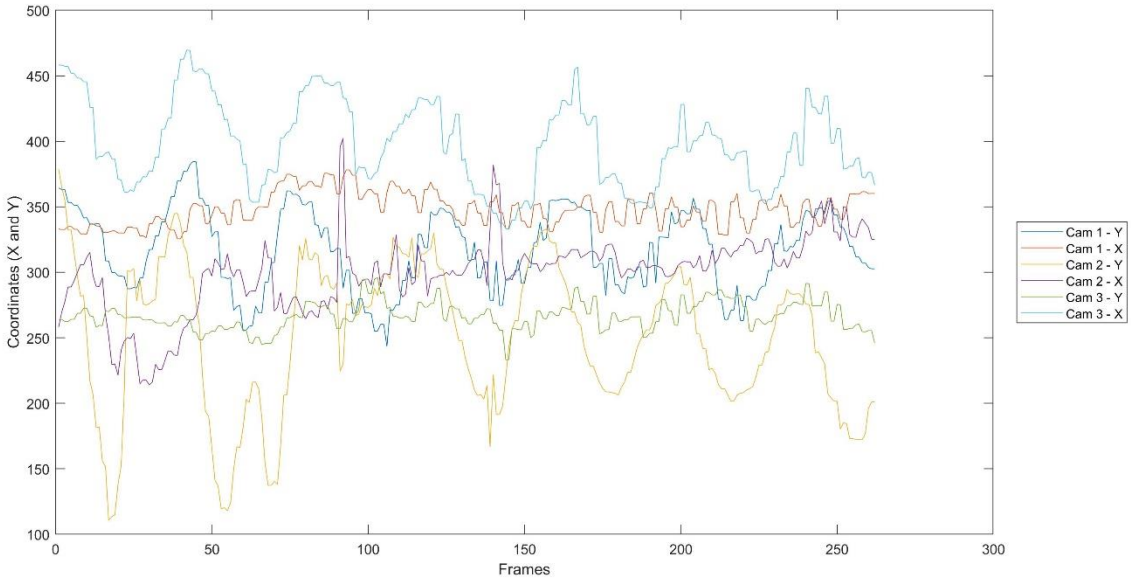

Fig. 3 Motion of paint can during trial 2: vertical oscillation with noise

Fig. 2 and Fig. 3 both describe the same vertical oscillation motion, but trial 2 has noise introduced by camera shaking. Data from trial 1 shows discernible cyclic motion from 3 variables (1-Y, 2-Y and 3-X) but data from trial 2 has high levels of noise without obvious pattern.

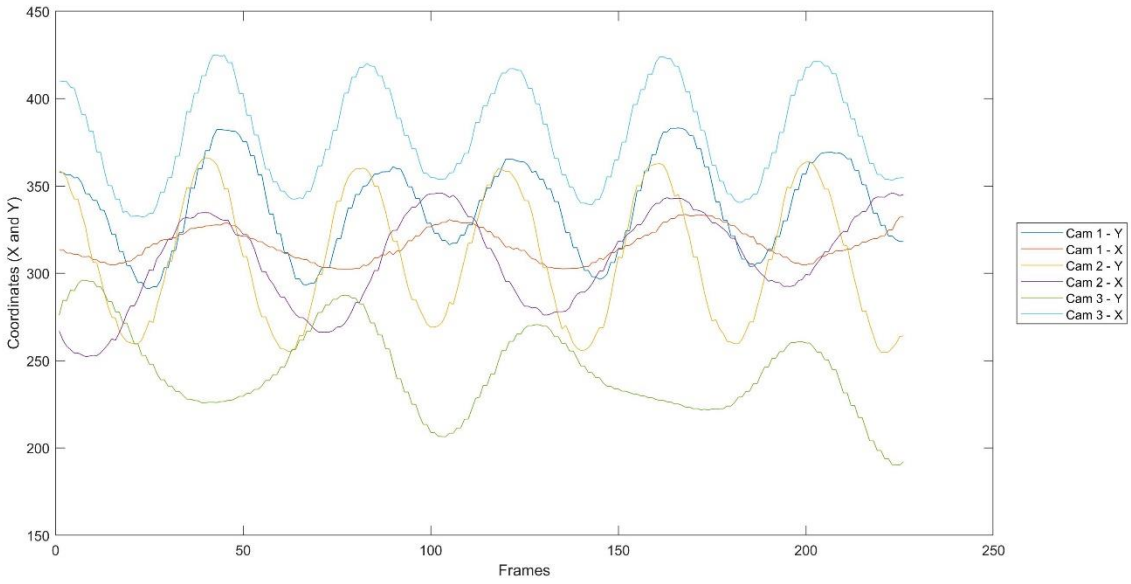

Fig. 4 Motion of paint can during trial 3: vertical oscillation with horizontal swing

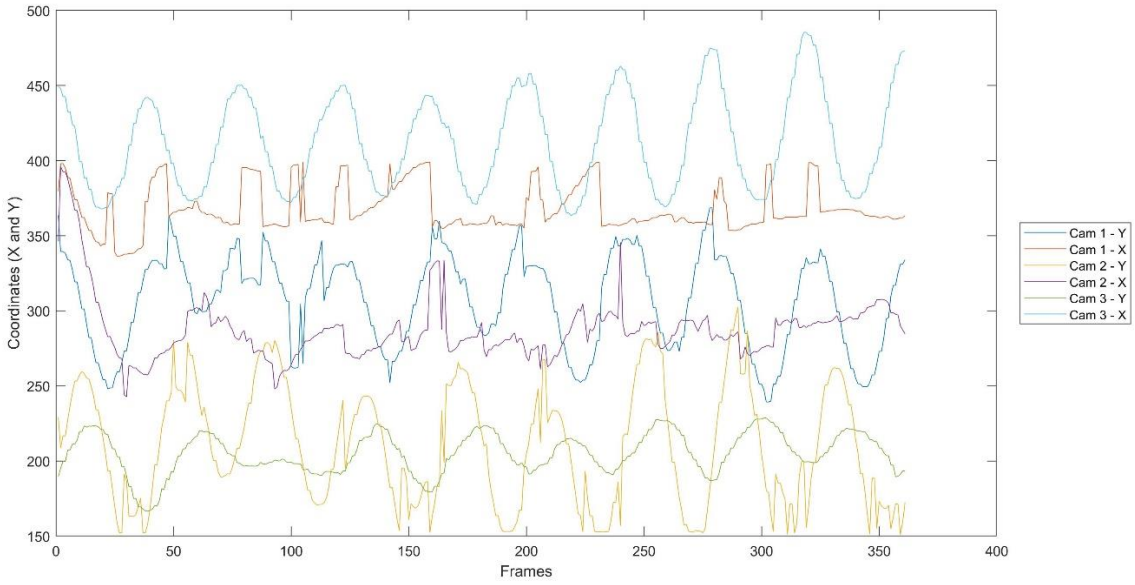

Fig. 5 Motion of paint can during trial 4: vertical oscillation with horizontal swing and rotation

Fig. 4 and Fig. 5 refer to trial 3 and trial 4, both of which include additional motion in the horizontal plane. Fig. 4 shows two distinct frequencies based on the patterns, but Fig. 5 shows a more erratic pattern, potentially due to failed tracking of the rotating flashlight.

Variance of Data with New Basis

Upon converting to a new basis, the covariance matrices of the four trials are calculated, and the diagonal entries, corresponding to variance of the new basis, are summarized below in Table 3.

Table 3 Variance of converted data during four trials (significant variance bolded)

Trial 1	Trial 2	Trial 3	Trial 4
7230.038	**3727.912**	**2541.137**	**1945.931**
328.7272	**1255.265**	**1413.462**	**1565.021**
115.2299	**808.4681**	251.0226	**467.8848**
30.85505	385.9819	32.03723	253.6688
16.92006	146.7553	18.91144	143.3845
12.09542	89.46789	11.80586	54.5641

The threshold for significant variance is set to 400, based on trial 1, assuming noise behaviors are similar. The noise is possibly coming from minor camera shaking and imprecise tracking during motion capture.

Degree of Freedom of Systems

Based on Table 3, the degree of freedom for each system is determined by the number of variances above the significance threshold. The results and their corresponding power are summarized below in Table 4. For trial 1, 3 and 4, the DOFs are within expectation considering the different levels of motions. However, DOF fails to predict the motion in trial 2, which consists of only 1-dimensional oscillation. This could potentially be attributed to the noise generated by camera shaking. As the shaking motion is highly random, the captured data includes 1-dimensional oscillation and 3-dimensional shaking, resulting in a 3-dimensional motion of the can from the cameras' perspectives.

Table 4 Degree of freedom of each trial and their power

	Trial 1	Trial 2	Trial 3	Trial 4
Degree of Freedom	1	3	2	3
Power	0.934854	0.90299	0.926488	0.898065

All DOFs identified produce power at approximately 90%, indicating a good representation of original data.

Section V – Summary and Conclusions

Based on the results, PCA based on SVD successfully computed the correct degree of freedom in most scenarios. But high level of noise from camera shaking can still contaminate the data and affect accuracy. Recommended next steps would be to filter the motion data in trial 2 using frequency analysis, e.g., Fourier transform, for better modal extraction. Additionally, future project can use pattern recognition based on spatial principal components between frames, similar to facial recognition, instead of intensity threshold for more accurate motion tracking. Nevertheless, PCA is a very powerful tool in dimension reduction and data analysis in general.

Appendix A – MATLAB functions used

- $rgb2gray$: converts color image into gray scale image.
- $find$: locate the nonzero entries in an array, returns logical.
- svd: performs standard singular value decomposition with silent rows/columns.
- cov: compute the covariance matrix of given data.

Appendix B – MATLAB codes


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

Loading data

```
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');
load('cam1_2.mat');
load('cam2_2.mat');
load('cam3_2.mat');
load('cam1_3.mat');
load('cam2_3.mat');
load('cam3_3.mat');
load('cam1_4.mat');
load('cam2_4.mat');
load('cam3_4.mat');
```

Test 1

```
% Camera 1
data = vidFrames1_1;
centroid_1 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 250);
    % filter stay out zone
    ind(:,[1:300 400:end]) = 0;
    ind(1:200,:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_1(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')
```

```
    % plot trajectory of centroid
%     scatter(centroid_1(j,2),480-centroid_1(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 2
data = vidFrames2_1;
centroid_2 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 250);
    % filter stay out zone
    ind(:,[]) = 0;
    ind([],:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_2(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_2(j,2),480-centroid_2(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 3
data = vidFrames3_1;
centroid_3 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 245);
    % filter stay out zone
    ind(:,1:200) = 0;
    ind([],:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right
```

```
    centroid_3(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_3(j,2),480-centroid_3(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Alignment
TF = islocalmax(centroid_1(:,1),'MinProminence',20);
Ind = find(TF);
centroid_1_t = centroid_1(Ind(1):end,:);
TF = islocalmax(centroid_2(:,1),'MinProminence',20);
Ind = find(TF);
centroid_2_t = centroid_2(Ind(1):end,:);
TF = islocalmax(centroid_3(:,2),'MinProminence',20); % D1 D2 swapped due to tilt of camera 3
Ind = find(TF);
centroid_3_t = centroid_3(Ind(1):end,:);

% Truncation
p_end = min([size(centroid_1_t,1) size(centroid_2_t,1) size(centroid_3_t,1)]);
centroid_1_t = centroid_1_t(1:p_end,:);
centroid_2_t = centroid_2_t(1:p_end,:);
centroid_3_t = centroid_3_t(1:p_end,:);
% % plot the centroid from 3 cameraa
% plot(1:p_end,centroid_1_t,1:p_end,centroid_2_t,1:p_end,centroid_3_t);
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')

% PCA
combined = [centroid_1_t centroid_2_t centroid_3_t]';
n = size(combined,2);
mn = mean(combined,2);
deviation = combined - repmat(mn,1,n); % calculate variance from mean
[U,~,~] = svd(deviation);
new = U'*deviation; % reconstruct matrix w/ new basis
% Covariance matrix
C_new = cov(new');
PCA_1 = diag(C_new);
DOF_1 = sum(PCA_1 > 400);

% % Reconstruct
% subplot((DOF_1+1),1,1)
% plot(1:n,deviation)
% title('Original Data')
```

```
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')
% for j = 1:DOF_1
%     recon = U(:,j)*S(j,j)*V(:,j)';
%     subplot((DOF_1+1),1,j+1)
%     plot(1:n,recon)
%     title(strcat('Mode',32,num2str(j)))
%     xlabel('Frames')
%     ylabel('Coordinates')
% end
```

Test 2

```
% Camera 1
data = vidFrames1_2;
centroid_1 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 245);
    % filter stay out zone
    ind(:,[1:300 400:end]) = 0;
    ind(1:200,:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_1(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_1(j,2),480-centroid_1(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 2
data = vidFrames2_2;
centroid_2 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);
```

```
    % locate beacon
    ind = (temp_gray > 245);
    % filter stay out zone
    ind(:,[]) = 0;
    ind([],:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_2(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_2(j,2),480-centroid_2(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 3
data = vidFrames3_2;
centroid_3 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 240);
    % filter stay out zone
    ind(:,1:200) = 0;
    ind(1:200,:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_3(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_3(j,2),480-centroid_3(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Alignment
TF = islocalmax(centroid_1(:,1),'MinProminence',20);
```

```
Ind = find(TF);
centroid_1_t = centroid_1(Ind(1):end,:);
TF = islocalmax(centroid_2(:,1),'MinProminence',20);
Ind = find(TF);
centroid_2_t = centroid_2(Ind(1):end,:);
TF = islocalmax(centroid_3(:,2),'MinProminence',20);
Ind = find(TF);
centroid_3_t = centroid_3(Ind(1):end,:);

% Truncation
p_end = min([size(centroid_1_t,1) size(centroid_2_t,1) size(centroid_3_t,1)]);
centroid_1_t = centroid_1_t(1:p_end,:);
centroid_2_t = centroid_2_t(1:p_end,:);
centroid_3_t = centroid_3_t(1:p_end,:);
% % plot the centroid from 3 cameraa
% plot(1:p_end,centroid_1_t,1:p_end,centroid_2_t,1:p_end,centroid_3_t);
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')

% PCA
combined = [centroid_1_t centroid_2_t centroid_3_t]';
n = size(combined,2);
mn = mean(combined,2);
deviation = combined - repmat(mn,1,n); % calculate variance from mean
[U,~,~] = svd(deviation);
new = U'*deviation; % reconstruct matrix w/ new basis
% Covariance matrix
C_new = cov(new');
PCA_2 = diag(C_new);
DOF_2 = sum(PCA_2 > 400);
P_2 = sum(PCA_2(1:DOF_2))/sum(PCA_2);

% %% filter test
%
% test = centroid_1-mean(centroid_1,1);
% n = size(test,1);
% t = 1:n;
% test_t = abs(fft(test));
% f = [0:(n)/2-1 -(n)/2:-1];
% test_ts = fftshift(test_t);
% fs = fftshift(f);
%
% a = .005;
% g = exp(-a*(f).^2)';
% % plot(fftshift(g))
% % plot(fs,test_ts);
%
% test_tf(:,1) = test_t(:,1).*g;
% test_tf(:,2) = test_t(:,2).*g;
% test_f = ifft(test_tf);
%
% subplot(2,1,1)
```

```
% plot(t,test);
% subplot(2,1,2)
% plot(t,test_f);

% % Reconstruct
% subplot((DOF_2+1),1,1)
% plot(1:n,deviation)
% title('Original Data')
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')
% for j = 1:DOF_2
%     recon = U(:,j)*S(j,j)*V(:,j)';
%     subplot((DOF_2+1),1,j+1)
%     plot(1:n,recon)
%     title(strcat('Mode',32,num2str(j)))
%     xlabel('Frames')
%     ylabel('Coordinates')
% end
```

Test 3

```
% Camera 1
data = vidFrames1_3;
centroid_1 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 250);
    % filter stay out zone
    ind(:,[1:300 400:end]) = 0;
    ind(1:200,:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_1(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_1(j,2),480-centroid_1(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end
```

```
% Camera 2
data = vidFrames2_3;
centroid_2 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 250);
    % filter stay out zone
    ind(:,[]) = 0;
    ind([],:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_2(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_2(j,2),480-centroid_2(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 3
data = vidFrames3_3;
centroid_3 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 240);
    % filter stay out zone
    ind(:,1:200) = 0;
    ind([],:) = 0;
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_3(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_3(j,2),480-centroid_3(j,1))
```

```
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Alignment
TF = islocalmax(centroid_1(:,1),'MinProminence',20);
Ind = find(TF);
centroid_1_t = centroid_1(Ind(1):end,:);
TF = islocalmax(centroid_2(:,1),'MinProminence',20);
Ind = find(TF);
centroid_2_t = centroid_2(Ind(1):end,:);
TF = islocalmax(centroid_3(:,2),'MinProminence',20); % swap b/t D1 and D2
Ind = find(TF);
centroid_3_t = centroid_3(Ind(1):end,:);

% Truncation
p_end = min([size(centroid_1_t,1) size(centroid_2_t,1) size(centroid_3_t,1)]);
centroid_1_t = centroid_1_t(1:p_end,:);
centroid_2_t = centroid_2_t(1:p_end,:);
centroid_3_t = centroid_3_t(1:p_end,:);
% % plot the centroid from 3 cameraa
% plot(1:p_end,centroid_1_t,1:p_end,centroid_2_t,1:p_end,centroid_3_t);
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')

% PCA
combined = [centroid_1_t centroid_2_t centroid_3_t]';
n = size(combined,2); % number of observations
mn = mean(combined,2);
deviation = combined - repmat(mn,1,n); % calculate variance from mean
[U,~,~] = svd(deviation);
new = U'*deviation; % reconstruct matrix w/ new basis
% Covariance matrix
C_new = cov(new');
PCA_3 = diag(C_new);
DOF_3 = sum(PCA_3 > 400);
P_3 = sum(PCA_3(1:DOF_3))/sum(PCA_3);

% % Reconstruct
% subplot((DOF_3+1),1,1)
% plot(1:n,deviation)
% title('Original Data')
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')
% for j = 1:DOF_3
%     recon = U(:,j)*S(j,j)*V(:,j)';
%     subplot((DOF_3+1),1,j+1)
%     plot(1:n,recon)
```

```
%     title(strcat('Mode',32,num2str(j)))
%     xlabel('Frames')
%     ylabel('Coordinates')
% end
```

Test 4

```
% Camera 1
data = vidFrames1_4;
centroid_1 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 245);
    % filter stay out zone
    ind(:,[1:300 400:end]) = 0;
    ind(1:200,:) = 0;
    % reduce the area tracked on can sidewall
    [Row,Col] = find(ind);
    if isempty(Row)
    else
        ind((min(Row)+5):end,:) = 0;
    end
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_1(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_1(j,2),480-centroid_1(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 2
data = vidFrames2_4;
centroid_2 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 250);
```

```
    % filter stay out zone
    ind(:,[]) = 0;
    ind(1:150,:) = 0;
    % reduce the area tracked on can sidewall
    [Row,Col] = find(ind);
    if isempty(Row)
    else
        ind((min(Row)+5):end,:) = 0;
    end
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_2(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')

    % plot trajectory of centroid
%     scatter(centroid_2(j,2),480-centroid_2(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Camera 3
data = vidFrames3_4;
centroid_3 = zeros(size(data,4),2);
for j = 1:size(data,4)
    % initialization
    temp = data(:,:,:,j);
    temp_gray = rgb2gray(temp);

    % locate beacon
    ind = (temp_gray > 235);
    % filter stay out zone
    ind(:,1:200) = 0;
    ind([],:) = 0;
%     % reduce the area tracked on can sidewall
%     [Row,Col] = find(ind);
%     if isempty(Row)
%     else
%         ind(:,(min(Col)+5):end) = 0;
%     end
    [D1,D2] = find(ind); % D1: top to bottom; D2: left to right

    centroid_3(j,:) = mean([D1 D2],1);

    % plot tracked pixels
%     temp_gray(ind) = 0;
%     imshow(temp_gray)
%     title('Tracked Pixels Labeled Black')
```

```
    % plot trajectory of centroid
%     scatter(centroid_3(j,2),480-centroid_3(j,1))
%     set(gca,'Ylim',[0 480],'Xlim',[0 640])
%     xlabel('Horizontal Location')
%     ylabel('Vertical Location')
%     drawnow
end

% Alignment
TF = islocalmax(centroid_1(:,1),'MinProminence',20);
Ind = find(TF);
centroid_1_t = centroid_1(Ind(1):end,:);
TF = islocalmax(centroid_2(:,1),'MinProminence',20);
Ind = find(TF);
centroid_2_t = centroid_2(Ind(1):end,:);
TF = islocalmax(centroid_3(:,2),'MinProminence',20); % swap b/t D1 and D2
Ind = find(TF);
centroid_3_t = centroid_3(Ind(1):end,:);

% Truncation
p_end = min([size(centroid_1_t,1) size(centroid_2_t,1) size(centroid_3_t,1)]);
centroid_1_t = centroid_1_t(1:p_end,:);
centroid_2_t = centroid_2_t(1:p_end,:);
centroid_3_t = centroid_3_t(1:p_end,:);
% % plot the centroid from 3 cameraa
% plot(1:p_end,centroid_1_t,1:p_end,centroid_2_t,1:p_end,centroid_3_t);
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')

% PCA
combined = [centroid_1_t centroid_2_t centroid_3_t]';
n = size(combined,2); % number of observations
mn = mean(combined,2);
deviation = combined - repmat(mn,1,n); % calculate variance from mean
[U,~,~] = svd(deviation);
new = U'*deviation; % reconstruct matrix w/ new basis
% Covariance matrix
C_new = cov(new');
PCA_4 = diag(C_new);
DOF_4 = sum(PCA_4 > 400);
P_4 = sum(PCA_4(1:DOF_4))/sum(PCA_4);

% % Reconstruct
% subplot((DOF_4+1),1,1)
% plot(1:n,deviation)
% title('Original Data')
% xlabel('Frames')
% ylabel('Coordinates (X and Y)')
% legend('Cam 1 - Y','Cam 1 - X','Cam 2 - Y','Cam 2 - X','Cam 3 - Y','Cam 3 -
X','location','eastoutside')
% for j = 1:DOF_4
%     recon = U(:,j)*S(j,j)*V(:,j)';
```

```
%     subplot((DOF_4+1),1,j+1)
%     plot(1:n,recon)
%     title(strcat('Mode',32,num2str(j)))
%     xlabel('Frames')
%     ylabel('Coordinates')
% end
```

Summary

```
PCA = [PCA_1 PCA_2 PCA_3 PCA_4];
DOF = [DOF_1 DOF_2 DOF_3 DOF_4];
% Power = [P_1 P_2 P_3 P_4];
```

Published with MATLAB® R2019b