AMATH 482 Winter 2020
Homework 3: Principal Component Analysis

Ozan Erdal

February 7, 2020

1 Introduction and Overview

Many of the applications of math occur in the field of physical sciences. In this case, the dynamics associated with the harmonic oscillation of a hanging mass system can be measured and analyzed using concepts from linear algebra and statistics. The main tool for this experiment was Singular Value Decomposition (SVD). In particular, an extension of it called Principal Component Analysis (PCA) was used.
The data, which was collected from several videos by tracking the motion of a paint can. 4 different experiments were conducted to highlight the strengths and shortcomings of PCA. The 4 experiments were:

1. Ideal case: Well recorded videos of a correctly conducted experiment.
2. Noisy case: Shakily recorded videos of a correctly conducted experiment.
3. Horizontal displacement: The mass is released with motion in the x-y direction and z plane.
4. Horizontal displacement rotation: Similar to case 3, but there is also rotation added to the can.

The collection of the data itself was a tedious task in and of itself, so the process for doing so will be discussed at length. The difficulty associated with data collection from the videos was more realistic of a real experiment and would negatively impact the accuracy of the subsequent PCA when done poorly. For each case, 3 cameras were used, each of which was from a different angle. Additionally, every recording was of a different length, necessitating further adjustments before conducting PCA. When data collection and manipulation was completed well, the *principal components* obtained from PCA provided significant useful information on the motion of the object. These principal components are important as they are statistically uncorrelated, allowing for redundancies which arise from 3-dimensional data being collected using 3 separate 2-dimensional measurements.

2 Theoretical Background

Data Collection

The eventual data of interest for the PCA was a matrix of the paint can's positions in each dimension for each frame of the video. The x and y locations of the can for each camera angle resulting in 6 measurement types of data. For each case, a different number of frames was used since the recordings were all of different lengths. As a result, the matrix used was $\mathbf{X} \in \mathbb{R}^{m \times n}$ with $m = 6$ and n being the number of frames in the adjusted videos.
Tracking the object was most effectively done by utilizing the reflective properties of a metal paint can. Light from the room reflected of the rim of the can, providing a small area of very bright white in the recording. In terms of pixel values under the RGB color model, pure white is represented by a maximum pixel value for each color. Following this small collection of very high pixel values very accurately provided location information for the first 3 cases and still provided relatively insightful information even in the case when the object was rotating.
This process required further manipulation of the video. To get meaningful motion data, unrelated portions of the video were "masked". Converting the video to a series of camera frames allowed for direct access to the pixel information at each position and time. By setting each of the RGB values for unrelated pixels to 0, they would be essentially removed from the tracked area.
After the vectors of these positions were constructed, they were trimmed and shifted to all be the same length, and such that the footage aligned. In other words, the can is at the same part of the trajectory in each camera angle. From a physics standpoint, this is referred to the oscillations being "in phase". Although some data was lost in this process, harmonic motion is periodic, so important information was not compromised.

PCA

Once collected, singular value decomposition can be performed on the *snapshot matrix* $\mathbf{X}$ using the MATLAB function *svd()*. From this, information on the singular values and the related eigenvalues can be determined from the results of the decomposition. From this, the *energy* associated with each singular value can be determined. For the purposes of PCA, this determines how much information is stored in each of the principal components. For the cases where the motion is purely harmonic (case 1 and 2), almost all of the information should be in one principal component. Noise will complicate this conclusion however. For the cases where the motion is harmonic and displacement is present in the *x*-*y* plane, the information should be mostly in the first three principal components.
This particular distribution of energy among the singular values shows the redundancy of data and is indicative of the lack of need for all 6 measurements.

Algorithm Implementation and Development

Data Collection

The size and shape of the masks was gradually adjusted until the can was barely in frame at all points. Although using a stationary mask did not seem like it would be the best idea in terms of accuracy, when experimenting with a moving mask, very little difference in tracking ability resulted. The third camera angle for each case was peculiar in that it was rotated significantly off the axis of motion compared to the other two recordings. Since PCA is concerned with variance and dimension reduction rather than a relation to an axis or a line of best fit like other statistical techniques, this angle did not matter in the PCA.

PCA

For each case PCA was conducted. After *svd()* was used, the singular values were obtained from the diagonal of the Σ matrix using MATLAB's *diag()* command. The energy for each singular value was obtained using the following formula: $energy_i = \frac{\sigma_i^2}{\sum sigma^2}$.
Furthermore, the principal component projection, which was the goal to obtain, was found from $\mathbf{Y} = \mathbf{U}^T\mathbf{X}$.

Computational Results

Data Collection

For each case, the trajectory determined from the first camera is shown below in **Figure 1**. The noisy case, although conducted as a good example had significant, although consistent, motion in the x and y directions. Regardless, in all 4 cases, clear vertical motion of the object was visible.

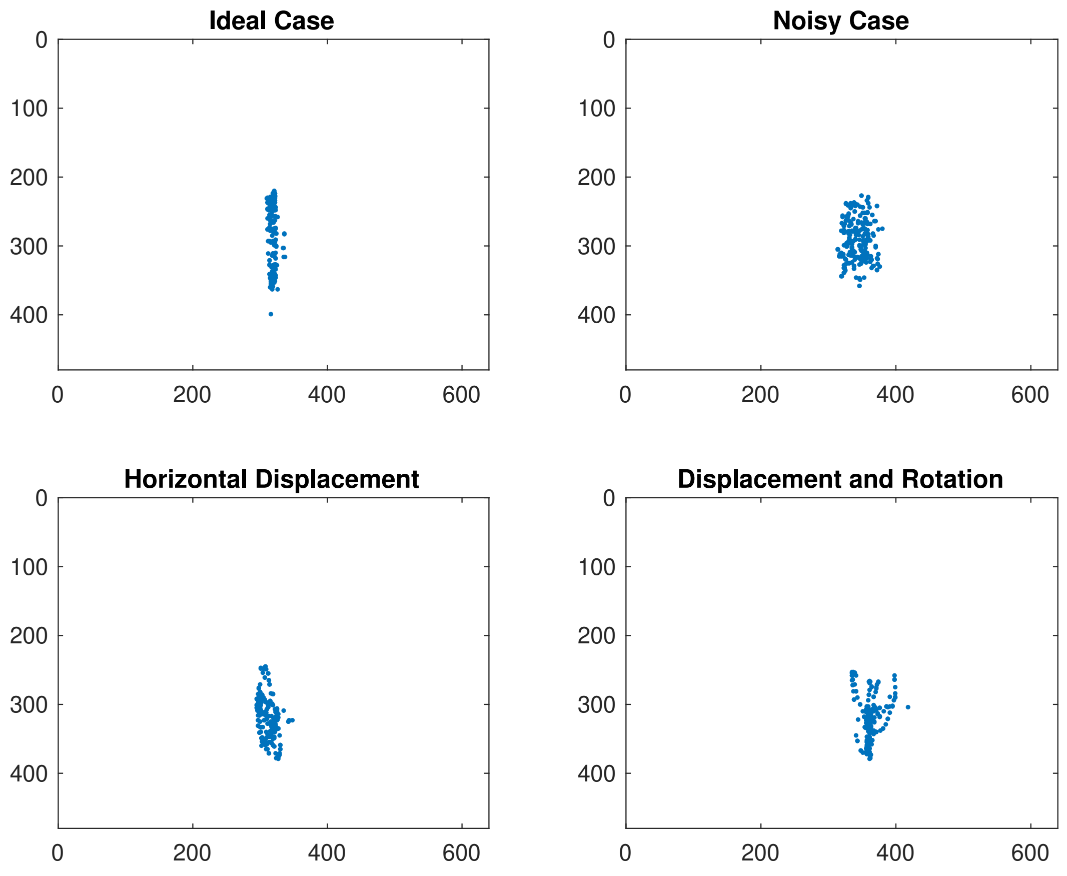

Figure 1: Trajectories from camera 1 for each case.

Below in **Figure 2** are stills of each camera angle with the respective masks applied. Since these are stills, the ideal case is shown since the frames are the least blurry.

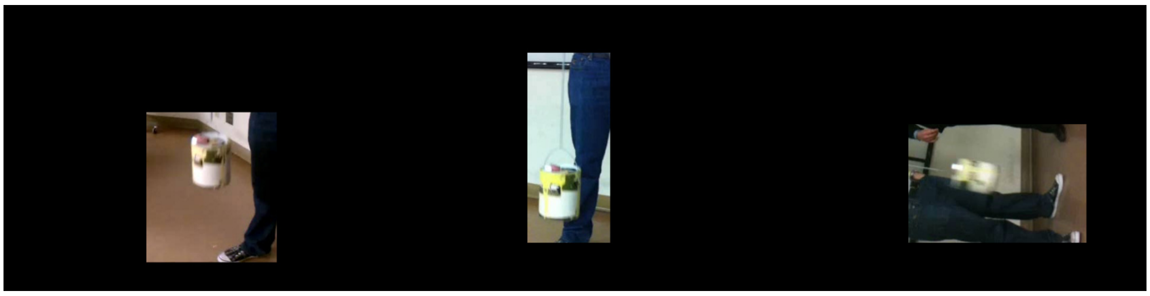

Figure 2: Masks for case 1 for each camera.

PCA

For each case, these **Y** matrices can be plotted using the MATLAB *waterfall()* command. Below are these plots of the snapshot matrix side-by-side with the principal component projections to highlight the difference between the two.

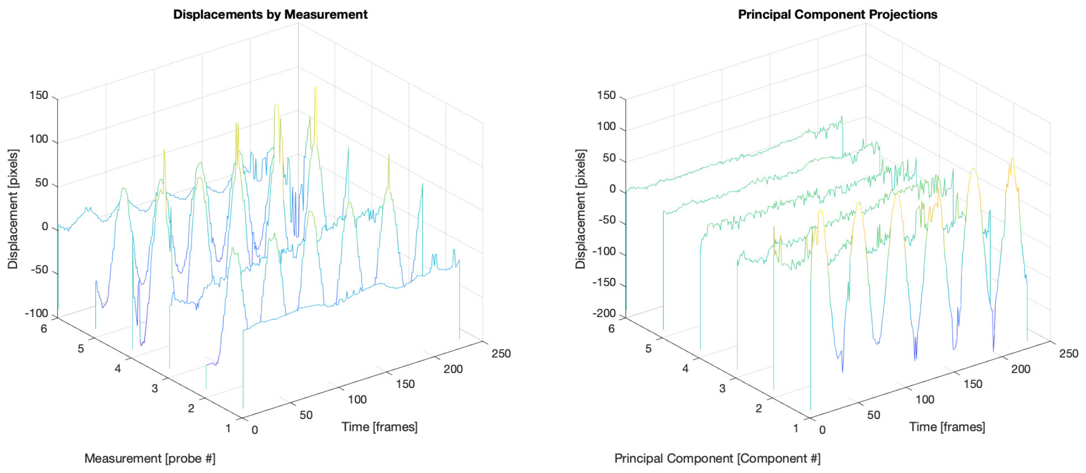

Figure 3: Waterfall plot for ideal case.

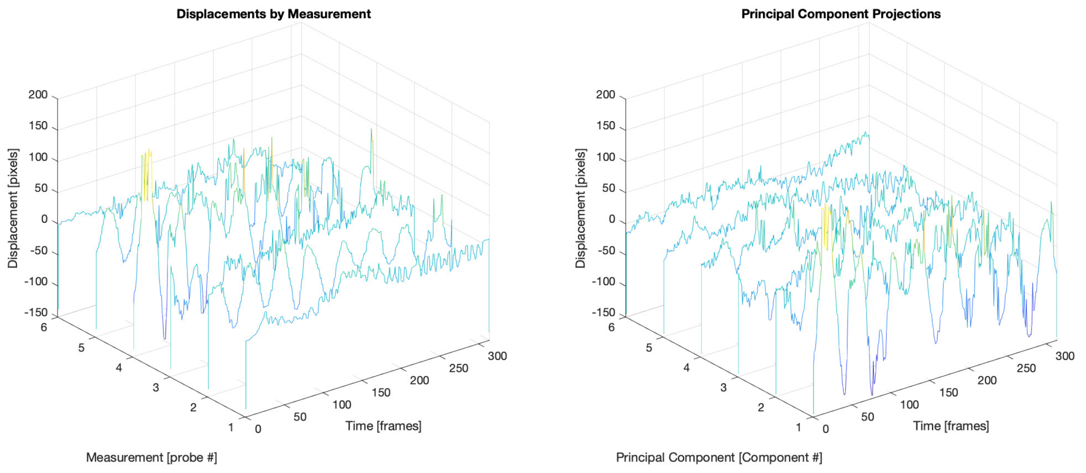

Figure 4: Waterfall plot for noisy case.

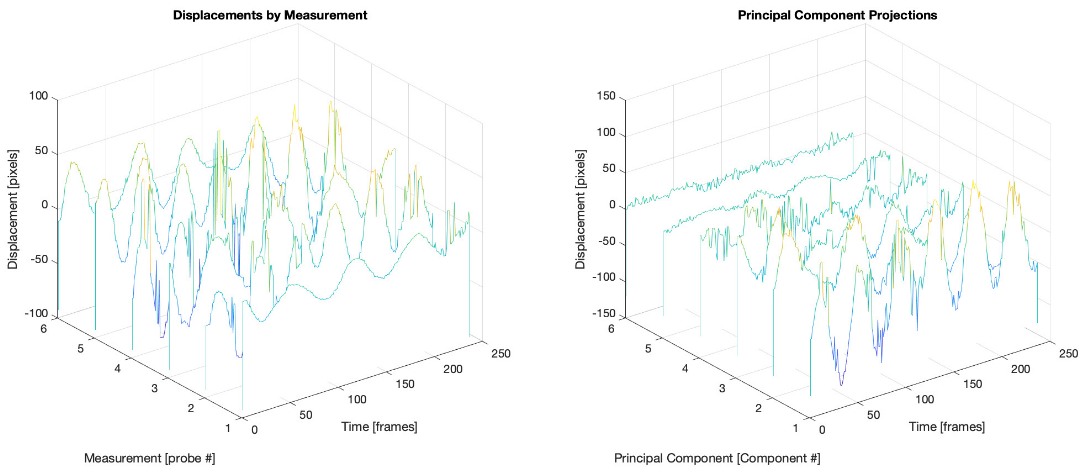

Figure 5: Waterfall plot for horizontal displacement case.

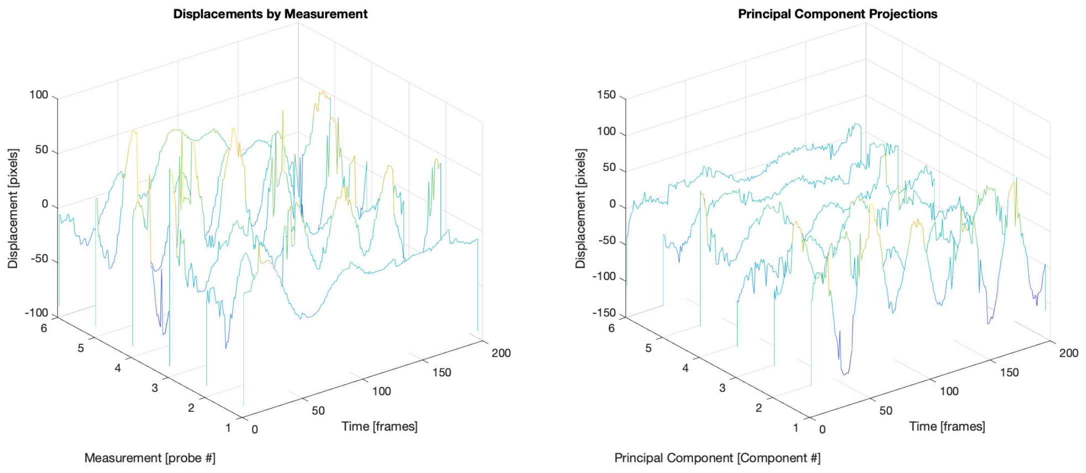

Figure 6: Waterfall plot for rotation and displacement case.

Summary and Conclusions

Although pretty much no knowledge of physics was used, information on the motion of the hanging mass system was determined. PCA was powerful enough even in the cases where the data was not accurately representing the physical situation, and although this experiment was fairly simple, the applications of PCA for modeling physical situations are plentiful.

Appendix A

[U,S,V]=svd(X)

Computes the singular value decomposition of the **X** matrix and breaks it into 3 matrices **U**, **S**, and **V**. For the purposes of this project, only the **U** and **S** matrices were used.

waterfall(X)

Multidimensional plotting tool. Used for plotting a matrix **X** against time in this experiment.

diag($\mathbf{X}$)

Returns a vector of the diagonal entries of $\mathbf{X}$.

Appendix B

Ideal Case

```
clear; close all; clc;

%% Load Video
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')

%% Play Video
implay(vidFrames1_1)

%% 1_1
figure(1)
vid = vidFrames1_1;
numFrames = size(vid,4);

x1_1 = zeros(1,numFrames);
y1_1 = zeros(1,numFrames);

for j = 1:1
    V = vid(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,x1_1(j)] = max(mean(max(V,[],1),3));
    [~,y1_1(j)] = max(mean(max(V,[],2),3));
    c1 = imshow(V);
end

% plot(x1_1,y1_1,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% 2_1
figure(2)
vid = vidFrames2_1;
numFrames = size(vid,4);

x2_1 = zeros(1,numFrames);
y2_1 = zeros(1,numFrames);

for j = 1:1
    V = vid(:,:,:,j);
    V(1:80,:,:) = 0;
    V(400:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,380:640,:) = 0;

    [~,x2_1(j)] = max(sum(max(V,[],1),3));
    [~,y2_1(j)] = max(sum(max(V,[],2),3));
    c2 = imshow(V);
end

% plot(x2_1,y2_1,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% 3_1
figure(3)
```

```
vid = vidFrames3_1;
numFrames = size(vid,4);

x3_1 = zeros(1,numFrames);
y3_1 = zeros(1,numFrames);

for j = 1:1
    V = vid(:,:,:,j);
    V(1:200,:,:) = 0;
    V(400:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,540:640,:) = 0;

    [~,x3_1(j)] = max(sum(max(V,[],1),3));
    [~,y3_1(j)] = max(sum(max(V,[],2),3));
    c3 = imshow(V);
end

% plot(x3_1,y3_1,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% Images
c1 = get(c1,'CData');
c2 = get(c2,'CData');
c3 = get(c3,'CData');
%%
montage(cat(4,c1,c2,c3),'size',[1 NaN]);

%% Adjust video lengths
len = length(x1_1); % shortest vid

x2_1 = x2_1(10:len+10-1);
y2_1 = y2_1(10:len+10-1);
x3_1 = x3_1(1:len);
y3_1 = y3_1(1:len);

%%
close;
subplot(3,1,1)
plot(1:len,y1_1)
xlim([0 len])
subplot(3,1,2)
plot(1:len,y2_1)
xlim([0 len])
subplot(3,1,3)
plot(1:len,y3_1)
xlim([0 len])

%% PCA
X = [x1_1;y1_1;x2_1;y2_1;x3_1;y3_1];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X,'econ');
sig = diag(S);
lambda = diag(S).^2;
energy = sig.^2/sum(sig.^2); % 0.9970
rank_1 = U(:,1)*S(1,1)*V(:,1)';

Y = U'*X;
```

```

close
subplot(1,2,1)
waterfall(X)
title('Displacements by Measurement')
xlabel('Time [frames]'), ylabel('Measurement [probe #]'), zlabel('Displacement [
    pixels]')
subplot(1,2,2)
waterfall(Y)
title('Principal Component Projections')
xlabel('Time [frames]'), ylabel('Principal Component [Component #]'), zlabel('
    Displacement [pixels]')
```

Noisy Case

```
clear; close all; clc;

%% Load Video
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')

%% Play Video
implay(vidFrames3_2)

%% 1_1
figure(1)
vid = vidFrames1_2;
numFrames = size(vid,4);

x1_2 = zeros(1,numFrames);
y1_2 = zeros(1,numFrames);
for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,x1_2(j)] = max(mean(max(V,[],1),3));
    [~,y1_2(j)] = max(mean(max(V,[],2),3));
%     imshow(V);
end

plot(x1_2,y1_2,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% 2_1
figure(2)
vid = vidFrames2_2;
numFrames = size(vid,4);

x2_2 = zeros(1,numFrames);
y2_2 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:80,:,:) = 0;
    V(400:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,380:640,:) = 0;

    [~,x2_2(j)] = max(sum(max(V,[],1),3));
    [~,y2_2(j)] = max(sum(max(V,[],2),3));
%     imshow(V);
end

plot(x2_2,y2_2,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% 3_1
figure(3)
vid = vidFrames3_2;
numFrames = size(vid,4);

x3_2 = zeros(1,numFrames);
y3_2 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:180,:,:) = 0;
    V(380:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,540:640,:) = 0;

    [~,x3_2(j)] = max(sum(max(V,[],1),3));
    [~,y3_2(j)] = max(sum(max(V,[],2),3));
%     imshow(V);
end

plot(x3_2,y3_2,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% Adjust video lengths
len = length(x1_2); % shortest vid

x2_2 = x2_2(20:len+20-1);
y2_2 = y2_2(20:len+20-1);
x3_2 = x3_2(1:len);
y3_2 = y3_2(1:len);

%%
close;
subplot(3,1,1)
plot(1:len,y1_2)
xlim([0 len])
subplot(3,1,2)
plot(1:len,y2_2)
xlim([0 len])
subplot(3,1,3)
plot(1:len,y3_2)
xlim([0 len])

%% PCA
X = [x1_2;y1_2;x2_2;y2_2;x3_2;y3_2];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X,'econ');
sig = diag(S);
lambda = diag(S).^2;
energy = sig.^2/sum(sig.^2);
rank_1 = U(:,1)*S(1,1)*V(:,1)';
```

```
rank_3 = U(:,1:3)*S(1:3,1:3)*V(:,1:3)';

% 90.74 of energy in first 3 modes

Y = U'*X;

close
subplot(1,2,1)
waterfall(X)
title('Displacements by Measurement')
xlabel('Time [frames]'), ylabel('Measurement [probe #]'), zlabel('Displacement [
    pixels]')
subplot(1,2,2)
waterfall(Y)
title('Principal Component Projections')
xlabel('Time [frames]'), ylabel('Principal Component [Component #]'), zlabel('
    Displacement [pixels]')
```

Horizontal Displacement Case

```
clear; close all; clc;

%% Load Video
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')

%% Play Video
implay(vidFrames1_3)

%% 1_1
figure(1)
vid = vidFrames1_3;
numFrames = size(vid,4);

x1_3 = zeros(1,numFrames);
y1_3 = zeros(1,numFrames);
for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,460:640,:) = 0;

    [~,x1_3(j)] = max(mean(max(V,[],1),3));
    [~,y1_3(j)] = max(mean(max(V,[],2),3));
%     imshow(V);
end

plot(x1_3,y1_3,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% 2_1
figure(2)
vid = vidFrames2_3;
numFrames = size(vid,4);

x2_3 = zeros(1,numFrames);
y2_3 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:160,:,:) = 0;
    V(400:480,:,:) = 0;
    V(:,1:220,:) = 0;
    V(:,360:640,:) = 0;

    [~,x2_3(j)] = max(sum(max(V,[],1),3));
    [~,y2_3(j)] = max(sum(max(V,[],2),3));
%      imshow(V);
end

plot(x2_3,y2_3,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% 3_1
figure(3)
vid = vidFrames3_3;
numFrames = size(vid,4);

x3_3 = zeros(1,numFrames);
y3_3 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:180,:,:) = 0;
    V(340:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,540:640,:) = 0;

    [~,x3_3(j)] = max(sum(max(V,[],1),3));
    [~,y3_3(j)] = max(sum(max(V,[],2),3));
%      imshow(V);
end

plot(x3_3,y3_3,'.'); axis([0,480,0,640]); set(gca, 'YDir','reverse')

%% Adjust video lengths
min_len = min([length(x1_3) length(x2_3) length(x3_3)]);
len = length(x1_3); % first vid

%%
x1_3 = x1_3(len-min_len+1:len);
y1_3 = y1_3(len-min_len+1:len);
x2_3 = x2_3(35:min_len+35-1);
y2_3 = y2_3(35:min_len+35-1);
x3_3 = x3_3(1:min_len);
y3_3 = y3_3(1:min_len);

%%
close;
len = min_len;
subplot(3,1,1)
plot(1:len,y1_3)
xlim([0 len])
subplot(3,1,2)
plot(1:len,y2_3)
xlim([0 len])
subplot(3,1,3)
plot(1:len,y3_3)
xlim([0 len])

%% PCA
```

```
X = [x1_3;y1_3;x2_3;y2_3;x3_3;y3_3];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X,'econ');
sig = diag(S);
lambda = diag(S).^2;
energy = sig.^2/sum(sig.^2);
rank_1 = U(:,1)*S(1,1)*V(:,1)';

% 93.42 of energy in first 3 modes

Y = U'*X;

close
subplot(1,2,1)
waterfall(X)
title('Displacements by Measurement')
xlabel('Time [frames]'), ylabel('Measurement [probe #]'), zlabel('Displacement [
    pixels]')
subplot(1,2,2)
waterfall(Y)
title('Principal Component Projections')
xlabel('Time [frames]'), ylabel('Principal Component [Component #]'), zlabel('
    Displacement [pixels]')
```

Rotation and Displacement Case

```
clear; close all; clc;

%% Load Video
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')

%% Play Video
implay(vidFrames3_4)

%% 1_1
vid = vidFrames1_4;
numFrames = size(vid,4)/2;

x1_4 = zeros(1,numFrames);
y1_4 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:180,:,:) = 0;
    V(:,1:280,:) = 0;
    V(:,480:640,:) = 0;
%     V(1:y_mask_start(j),:,:) = 0;
%     V(y_mask_end(j):480,:,:) = 0;
%     V(:,1:x_mask_start(j),:) = 0;
%     V(:,x_mask_end(j):640,:) = 0;

    [~,x1_4(j)] = max(mean(max(V,[],1),3));
    [~,y1_4(j)] = max(mean(max(V,[],2),3));
%     imshow(V);
end
```

```

%%

x_mask_start = x1_4 - 50;
x_mask_end = x1_4 + 50;
y_mask_start = y1_4 - 50;
y_mask_end = y1_4 + 50;

plot(x1_4,y1_4,'.'); axis([0,640,0,480]); set(gca, 'YDir','reverse')

%% 2_1
figure(2)
vid = vidFrames2_4;
numFrames = size(vid,4);

x2_4 = zeros(1,numFrames);
y2_4 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:100,:,:) = 0;
    V(350:480,:,:) = 0;
    V(:,1:200,:) = 0;
    V(:,400:640,:) = 0;

    [~,x2_4(j)] = max(sum(max(V,[],1),3));
    [~,y2_4(j)] = max(sum(max(V,[],2),3));
%     imshow(V);
end

plot(x2_4,y2_4,'.'); axis([0,640,0,480]); set(gca, 'YDir','reverse')

%% 3_1
figure(3)
vid = vidFrames3_4;
numFrames = size(vid,4);

x3_4 = zeros(1,numFrames);
y3_4 = zeros(1,numFrames);

for j = 1:numFrames
    V = vid(:,:,:,j);
    V(1:150,:,:) = 0;
    V(340:480,:,:) = 0;
    V(:,1:240,:) = 0;
    V(:,540:640,:) = 0;

    [~,x3_4(j)] = max(sum(max(V,[],1),3));
    [~,y3_4(j)] = max(sum(max(V,[],2),3));
%     imshow(V);
end

plot(x3_4,y3_4,'.'); axis([0,640,0,480]); set(gca, 'YDir','reverse')

%% Adjust video lengths
len = length(x1_4); % first vid

%%
x1_4 = x1_4(1:len);
y1_4 = y1_4(1:len);
```

```
x2_4 = x2_4(35:len+35-1);
y2_4 = y2_4(35:len+35-1);
x3_4 = x3_4(1:len);
y3_4 = y3_4(1:len);

%%
close;
subplot(3,1,1)
plot(1:len,y1_4)
xlim([0 len])
subplot(3,1,2)
plot(1:len,y2_4)
xlim([0 len])
subplot(3,1,3)
plot(1:len,y3_4)
xlim([0 len])

%% PCA
X = [x1_4;y1_4;x2_4;y2_4;x3_4;y3_4];
[m,n] = size(X);
mn = mean(X,2);
X = X - repmat(mn,1,n);

[U,S,V] = svd(X,'econ');
sig = diag(S);
lambda = diag(S).^2;
energy = sig.^2/sum(sig.^2);
rank_1 = U(:,1)*S(1,1)*V(:,1)';

Y = U'*X;

close
subplot(1,2,1)
waterfall(X)
title('Displacements by Measurement')
xlabel('Time [frames]'), ylabel('Measurement [probe #]'), zlabel('Displacement [
    pixels]')
subplot(1,2,2)
waterfall(Y)
title('Principal Component Projections')
xlabel('Time [frames]'), ylabel('Principal Component [Component #]'), zlabel('
    Displacement [pixels]')
```