Principal Component Analysis Using SVD to visualize Fundamental Motion of Physical Systems

Connor Schleicher

February 21, 2020

Abstract

The singular value decomposition (SVD) is a tool to separate any matrix into three separate matrices which can provide insight into specific properties of the original data matrix. This insight can be applied across many different mediums. This analysis explores the ability of the SVD to decompose movement from a video and determine the motions observed. The input data is taken from three separate videos taken from different positions for four separate experiments. The motion becomes increasingly more complex with each experiment and the ability of the SVD to deconstruct a system with multiple degrees of freedom is analyzed.

1 Introduction and Overview

Without having in depth knowledge of an observed system, separating out the principal components of a system from the noise can be challenging. In this analysis the SVD is used to help separate the principal components and help identify and remove the redundancy in the data measurements. Once we know the principal components, we can study and analyze these to generate an understanding of the physics behind the system.

This analysis was broken into four distinct parts, each studying a similar but slightly different system. Each part is comprised of three separate camera recordings, where each camera is recording the same thing but at different angels and positions, and third camera is rotated by ninety degrees. The first and simplest case is a simple oscillatory system of a paint can attached to a spring. The second part adds noise by shaking the cameras, the third adds a horizontal displacement and the fourth and final case adds horizontal displacement with rotation.

Each of these methods is studied in a similar manner. The first step is to record the x and y position vectors of the paint can for each camera. The paint can's position was tracked by trying to track the light that was atop of the can. The light was found by recording the brightest pixel in a narrowed window around the can. Then the videos are synchronized and the position vectors are truncated to all be the same length and combined to create the full data matrix. Once the data is gathered then the analysis can be performed. This is done by computing the covariance matrix, SVD, and principal components. These items are graphed and analyzed to understand the system for each scenario.

2 Theoretical Background

2.1 Singular Value Composition (SVD)

The singular value decomposition (SVD) is a way to represent any matrix as the product of three different matrix transformations. Any matrix performs a stretch or a rotation to the value it is applied to, as defined in Equation 1 [1]. In this definition, U and V are both unitary matrices, and Σ is a diagonal matrix. Sigma is also, equal to the square root of the eigenvalues of matrix `A`.

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

Due to the construction of the SVD the sigma matrix is made in a way so that each value is in decreasing order along the diagonal, therefore the largest and most significant sigma value is located in position one. Along with the sigma matrix, the U and V are constructed to obtain certain properties. U is constructed as an orthonormal set of basis vectors. All these properties come together to accurately describe the main components of a matrix without constructing the entire matrix, just the first couple of primary basis vectors will define the major components of the system.

2.2 Covariance Matrix

The covariance matrix, is a matrix of size `m x m` where the diagonal elements represent the covariances of the different measurements. The covariance matrix is constructed by having the columns be equal to the data point in time, while the rows are set as the different measurements. For example, the first row could be the x position of an object, while the second row would be the y position of the same object. This could repeat over several different measurement points to construct the X matrix. The covariance matrix is defined by Equation 2, where X is the matrix of measurement systems over time.

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

3 Algorithm Implementation and Development

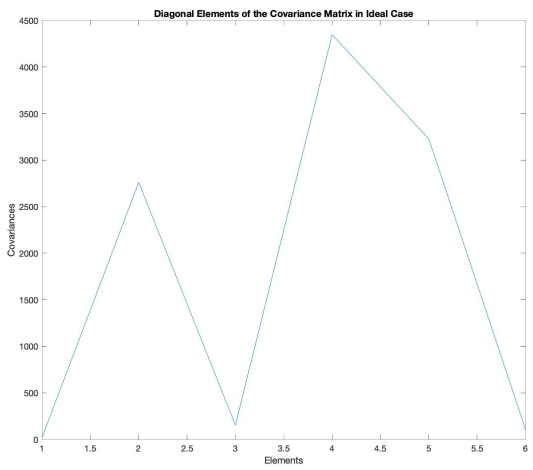
All four parts of this analysis followed the same algorithm structure. This structure is adaptable to each scenario, with just some manual maintenance to adjust the videos to start within the same time, and some minor tweaking of the tracking of the paint can. Algorithm 1 goes into some of the details that went into tracking the paint can and combining the videos. Once the data was gathered, then the analysis is performed by studying the SVD, the covariance matrix, and the principal components of the data to gather some insights into what was recorded. These next steps are outlined in Algorithm 2

Algorithm 1: Object Tracking

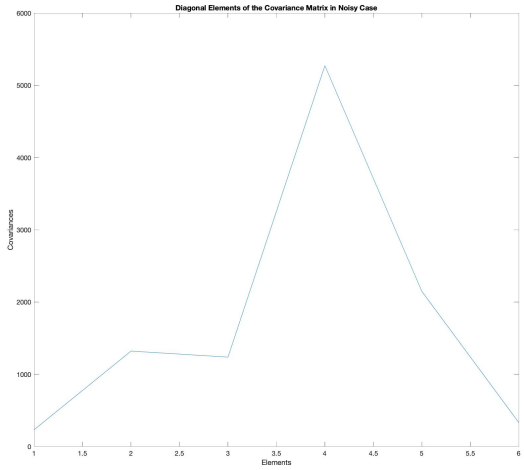
Import movie files
Convert each movie to grey scale for easier computations
for $i = 1 : 3$ **do**
  Will need to loop through each video and through each frame in each video
  for $j = 1 : numberframes$ **do**
    Create narrow window around the paint can's movement
    Remove all pixels but the brightest one
    Store the x and y positions in new vectors x_i and y_i
  end for
end for
Align the start times to match the paint can's position
Trim the longer vectors so each are the same length
Combine each vector to one matrix X such that $X = [x_1; y_1; ...]$

Algorithm 2: Video Analysis

Perform SVD on the data
Create the covariance matrix and principal components
Plot the singular values of the SVD
Plot the measurement and time dependent principal components



(a) Ideal Case



(b) Noisy Case

Figure 1: Graph of the Diagonal Covariances for the Ideal and Noisy Cases

4 Computational Results

4.1 Part I Ideal Case

The first part of this analysis was an ideal case of a mass oscillating on a spring. The diagonal elements of the covariance matrix is plotted in Figure 1a. Here we can see that the greatest variance is at measurements 2,4, and 5. Logically this makes sense since these measurements are the y position vectors for the first two cameras and the x position of the rotated third camera. Since these positions are changing then the covariances would be higher. In Figure 2 we can see the singular values and principal modes. From the first two panels the singular values drop off in intensity after the first three, with the first mode being the most significant. Then when the modes are graphed across the measurements and time we can see how they affect the overall data. Mode 1 is flat across both dimensions and would correspond to the stable x-axis, where little to no movement occurred. The other two modes show the oscillation that was observed in the y-axis, and is visible in each measurement to varying degrees. The third mode in this case seems like a redundant component and the motion would be mostly described by the first two. This is backed up by the covariance showing similar values for cameras 1 and 2.

4.2 Part II Ideal Case With Noise

The noisy case is predominantly the same as the ideal case. The covariance elements, seen in Figure 1b differed mostly at measurements 3 and 5. This would lead to belief that the noise greatly affected the measurement in the x-axis on camera 2 and the measurement in the y-axis for camera 3 respectively. These assumptions are validated by watching the videos of the oscillations. Figure 3 shows very similar singular values and principal modes as in the ideal case. The noise was mostly pulled out except for some effect on the modes over time, where the oscillations are not as smooth. Similarly to the first case, the third mode appears to be redundant.

4.3 Part III Horizontal Displacement

A quick glance at the covariances seen in Figure 4a shows a large variance in measurements 5 and 6. These are overshadowing the other measurements but upon further inspection the other measurements showed a similar pattern to earlier. The large variance for the third camera is most likely from the difficulty of tracking the paint can in this scenario. When comparing the values in Figure 5 to the previous two cases, a

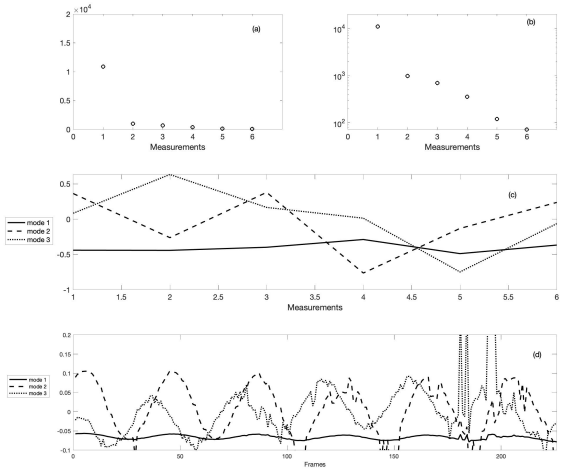


Figure 2: PCA Modes of the Ideal Case. The singular values are shown in (a) and a log plot (b). Panel (c) shows the first three modes in relation to the measurements, while panel (d) shows the same modes over time.

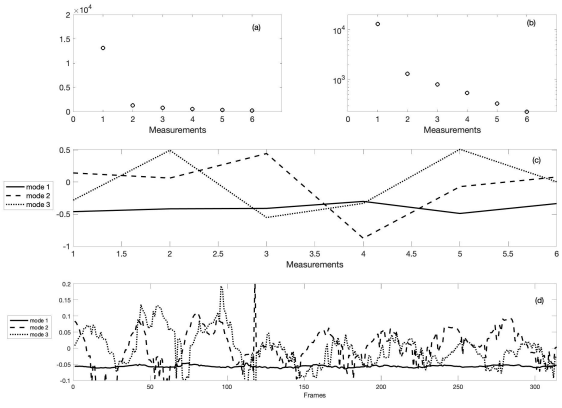
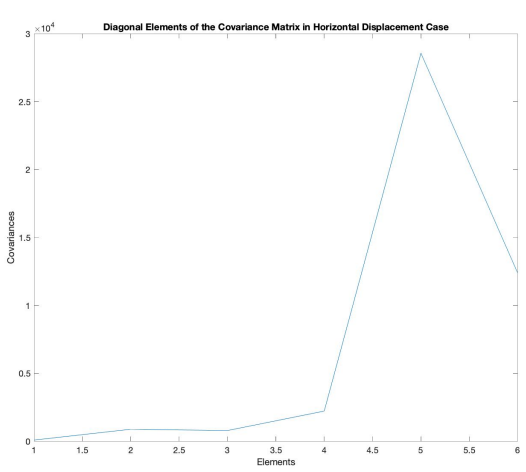
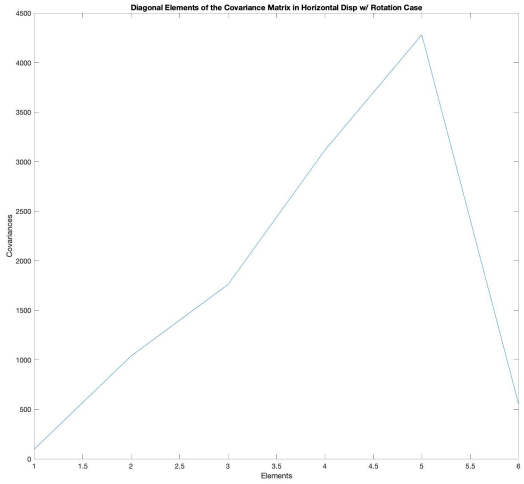


Figure 3: PCA Modes of the Noisy Case. The singular values are shown in (a) and a log plot (b). Panel (c) shows the first three modes in relation to the measurements, while panel (d) shows the same modes over time.



(a) Horizontal Displacement



(b) Horizontal Displacement with Rotation

Figure 4: Graph of the Diagonal Covariances for the Horizontal Displacement and Horizontal Displacement with Rotation Cases

lot of similarities are found. The singular values are largely the same, with just the second value being more prominent than before. The largest difference is seen graphing the principal modes over time. Instead of the second and third mode showing similar oscillations the second mode shows a smaller and less consistent oscillation, while the third is showing the main oscillation in the y-axis. The horizontal displacement appears to be much more difficult to track and appears to dampen or stop halfway through the videos. From panel (c), we see that camera 2 (measurement 4) was largely responsible for capturing the oscillation, while camera 3 (measurement 5) captured the horizontal displacement.

4.4 Part IV Horizontal Displacement and Rotation

The final case is again similar to the rest. Here the covariances are large for all but the first and last measurement, Figure 4b. This telling that there was more information coming from each camera and movement across almost every directions. Since this scenario has several degrees of movement, the principal modes become less clear. In Figure 6, we see how even with the extra movements the singular values appear largely the same as the first two cases, and the second value is not as prominent as the previous case. When comparing the modes, the second mode looks like it captured two separate movements. The graph is largely oscillatory but there are several sections where the oscillations could be interpreted as the rotations, especially clear at frames 250-275. Panel (c) shows that the rotation and horizontal displacement came almost entirely from measurement 5, the third camera.

5 Summary and Conclusions

This analysis shows how the SVD can be used to analyze a system without knowing the underlying forces. The ideal case shows how a simple oscillation is easily captured by separating the data matrix into its principal components. With noise added in, the SVD was largely unaffected. The noise was captured in the principal modes by making them less smooth. Once extra movement was added in, this started to affect some of the values in the SVD. The second and third singular values started to increase and become more important in defining the behaviour. When looking at the principal modes, the second mode changed from being an extra oscillatory measurement to defining the new movement. In the horizontal displacement it looks similar to a slight oscillation but much less pronounced and disappears halfway through the video. In

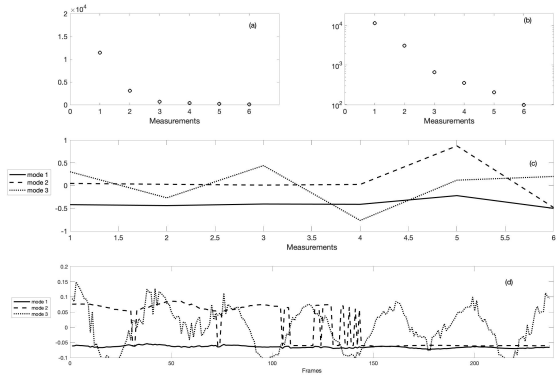


Figure 5: PCA Modes of the Horizontal Displacement Case. The singular values are shown in (a) and a log plot (b). Panel (c) shows the first three modes in relation to the measurements, while panel (d) shows the same modes over time.

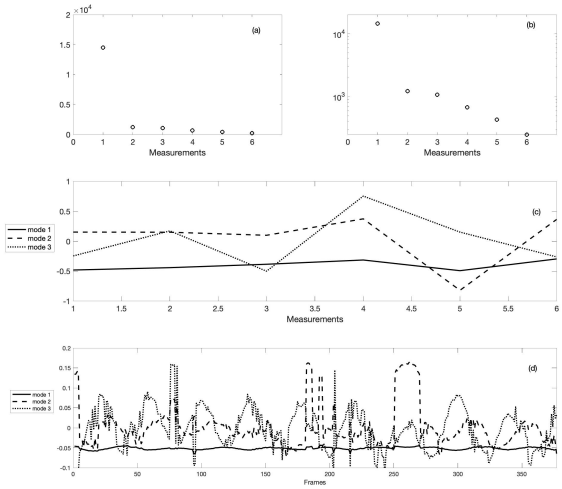


Figure 6: PCA Modes of the Horizontal Displacement with Rotation Case. The singular values are shown in (a) and a log plot (b). Panel (c) shows the first three modes in relation to the measurements, while panel (d) shows the same modes over time.

the final case with the rotation, the second mode captured the horizontal displacement as an oscillation but also captured part of the rotations as well. These rotations were difficult to accurately track since the light was out of sight during half the rotation.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

`https://github.com/ConnorSch/PCA_Analysis`

Appendix A MATLAB Functions

- `cov(X)` computes the covariance of the input matrix `X`. If x is `n x m` then the covariance matrix will be `m x m`
- `svd(X)` computes the full Singular Value Decomposition of matrix X. Defined as `X = USV*`

Appendix B MATLAB Code

```
%% Ideal Case for Cameras
clear all; close all; clc;

load('camera_data/cam1_1.mat');
cam1 = double(vidFrames1_1);

load('camera_data/cam2_1.mat');
cam2 = double(vidFrames2_1);

load('camera_data/cam3_1.mat');
cam3 = double(vidFrames3_1);

numFrames1 = size(cam1,4);
numFrames2 = size(cam2,4);
numFrames3 = size(cam3,4);

frames = [numFrames1, numFrames2, numFrames3];

xdim = size(cam1,2);
x = linspace(1,xdim,xdim);
ydim = size(cam1,1);
y = linspace(1,ydim,ydim);

cam1g = zeros(ydim, xdim, numFrames1);
cam2g = zeros(ydim, xdim, numFrames2);
cam3g = zeros(ydim, xdim, numFrames3);

for k = 1:numFrames1
    mov1(k).cdata = vidFrames1_1(:,:,:,k);
    mov1(k).colormap = [];
    cam1g(:,:,k) = rgb2gray(vidFrames1_1(:,:,:,k));
end
```

```
for k = 1:numFrames2
    mov2(k).cdata = vidFrames2_1(:,:,:,k);
    mov2(k).colormap = [];
    cam2g(:,:,k) = rgb2gray(vidFrames2_1(:,:,:,k));
end
for k = 1:numFrames3
    mov3(k).cdata = vidFrames3_1(:,:,:,k);
    mov3(k).colormap = [];
    cam3g(:,:,k) = rgb2gray(vidFrames3_1(:,:,:,k));
end
%% Movies
figure(1)
for i = 1:numFrames1

    subplot(1,3,1)
    X = frame2im(mov1(i)); % movie one starts off with can at top
    imshow(X);
    drawnow

    subplot(1,3,2)
    X2 = frame2im(mov2(i+14)); % i = 14 when can is at top
    imshow(X2);
    drawnow

    subplot(1,3,3)
    X3 = frame2im(mov3(i+6)); % i = 6 when can is at top
    imshow(X3);
    drawnow
end

%% Synchronizing the movie files
% reset two of the movie files so they are all synchronized
mov2_disp = 14 + 1;
mov3_disp = 6 + 1;
cam2g = cam2g(:,:,mov2_disp:end);
cam3g = cam3g(:,:,mov3_disp:end);

numFrames2 = size(cam2g,3);
numFrames3 = size(cam3g,3);

frames = [numFrames1, numFrames2, numFrames3];

%% getting the positions for the first camera
figure()
x1 = zeros(1,numFrames1);
y1 = zeros(1,numFrames1);
for i = 1:numFrames1
    frame = cam1g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    maxf = max(frame(:));
    width = 40;
    filt = zeros(ydim,xdim);
```

```
    filt(:,300:300+width) = frame(:,300:300+width);
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y1(i),x1(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the second camera
figure()
x2 = zeros(1,numFrames2);
y2 = zeros(1,numFrames2);
for i = 1:numFrames2
    frame = cam2g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    maxf = max(frame(:));
    filt = frame;
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y2(i),x2(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the third camera
figure()
x3 = zeros(1,numFrames3);
y3 = zeros(1,numFrames3);
for i = 1:numFrames3
    frame = cam3g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame));
    subplot(1,2,2)
    filt = frame;
    width = 240;
    filt = zeros(ydim,xdim);
    filt(width:350,:) = frame(width:350,:);
    maxf = max(filt(:));
    filt(filt<maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y3(i),x3(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% Prep the matrix for the SVD
x2 = x2(1:numFrames1);
y2 = y2(1:numFrames1);
x3 = x3(1:numFrames1);
y3 = y3(1:numFrames1);

X = [x1;y1;x2;y2;x3;y3];
[u,s,v] = svd(X);
lambda = diag(s).^2;
Y = u'*X;
```

```
Cx = cov(X');

figure()
plot(diag(Cx))
xlabel('Elements')
ylabel('Covariances')
title('Diagonal Elements of the Covariance Matrix in Ideal Case')

%% Playing with the SVD
figure()
sig = diag(s);

title('Singular Values and PCA Modes for Ideal Case')
sig=diag(s);
subplot(3,2,1), plot(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6])
xlabel('Measurements')
text(6,1.75*10^4,'(a)','Fontsize',[13])

subplot(3,2,2), semilogy(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Ytick',[10^0 10^2 10^3 10^4 10^5],...
    'Xtick',[0 1 2 3 4 5 6]);
text(6,1.4*10^4,'(b)','Fontsize',[13])
xlabel('Measurements')

xtest = linspace(1,6,6);
subplot(3,1,2)
plot(xtest,u(:,1),'k',xtest,u(:,2),'k--',xtest,u(:,3),'k:','Linewidth',[2])
set(gca,'Fontsize',[13])
xlabel('Measurements')
legend('mode 1','mode 2','mode 3','Location','NorthWest')
text(5.5,0.35,'(c)','Fontsize',[13])

subplot(3,1,3)
t = linspace(1,numFrames1,numFrames1);
plot(t, v(:,1),'k',t, v(:,2),'k--',t, v(:,3),'k:','Linewidth',[2])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
xlabel('Frames')
axis([0 numFrames1 -0.1 0.2])
text(215,0.15,'(d)','Fontsize',[13])

%% Ideal Case for Cameras
clear all; close all; clc;

load('camera_data/cam1_2.mat');
cam1 = double(vidFrames1_2);

load('camera_data/cam2_2.mat');
cam2 = double(vidFrames2_2);

load('camera_data/cam3_2.mat');
```

```
cam3 = double(vidFrames3_2);

numFrames1 = size(cam1,4);
numFrames2 = size(cam2,4);
numFrames3 = size(cam3,4);

frames = [numFrames1, numFrames2, numFrames3];

xdim = size(cam1,2);
x = linspace(1,xdim,xdim);
ydim = size(cam1,1);
y = linspace(1,ydim,ydim);

cam1g = zeros(ydim, xdim, numFrames1);
cam2g = zeros(ydim, xdim, numFrames2);
cam3g = zeros(ydim, xdim, numFrames3);

for k = 1:numFrames1
    mov1(k).cdata = vidFrames1_2(:,:,:,k);
    mov1(k).colormap = [];
    cam1g(:,:,k) = rgb2gray(vidFrames1_2(:,:,:,k));
end
for k = 1:numFrames2
    mov2(k).cdata = vidFrames2_2(:,:,:,k);
    mov2(k).colormap = [];
    cam2g(:,:,k) = rgb2gray(vidFrames2_2(:,:,:,k));
end
for k = 1:numFrames3
    mov3(k).cdata = vidFrames3_2(:,:,:,k);
    mov3(k).colormap = [];
    cam3g(:,:,k) = rgb2gray(vidFrames3_2(:,:,:,k));
end
%% Creating movies

figure(1)
for i = 1:numFrames1
    subplot(1,3,1)
    X = frame2im(mov1(i));
    imshow(X);
    drawnow

    subplot(1,3,2) % i = 16 when can is at top
    X2 = frame2im(mov2(i));
    imshow(X2);
    drawnow

    subplot(1,3,3)
    X3 = frame2im(mov3(i));
    imshow(X3);
    drawnow
end

%% Synchronizing the movie files
% reset movie files so they are all synchronized
```

```
mov2_disp = 16 + 1;
cam2g = cam2g(:,:,mov2_disp:end);

numFrames2 = size(cam2g,3);

frames = [numFrames1, numFrames2, numFrames3];

%% getting the positions for the first camera
figure()
x1 = zeros(1,numFrames1);
y1 = zeros(1,numFrames1);
for i = 1:numFrames1
    frame = cam1g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    width = 90;
    filt = zeros(ydim,xdim);
    filt(:,300:300+width) = frame(:,300:300+width);
    maxf = max(filt(:));
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y1(i),x1(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the second camera
figure()
x2 = zeros(1,numFrames2);
y2 = zeros(1,numFrames2);
for i = 1:numFrames2
    frame = cam2g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    filt = frame;
    maxf = max(filt(:));
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y2(i),x2(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the third camera
figure()
x3 = zeros(1,numFrames3);
y3 = zeros(1,numFrames3);
for i = 1:numFrames3
    frame = cam3g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame));
    subplot(1,2,2)
    width = 200;
    filt = zeros(ydim,xdim);
```

```
    filt(width:350,:) = frame(width:350,:);
    maxf = max(filt(:));
    filt(filt<maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y3(i),x3(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% Prep the matrix for the SVD
x2 = x2(1:numFrames1);
y2 = y2(1:numFrames1);
x3 = x3(1:numFrames1);
y3 = y3(1:numFrames1);

X = [x1;y1;x2;y2;x3;y3];
[u,s,v] = svd(X);
lambda = diag(s).^2;
Y = u'*X;

Cx = cov(X');

figure()
plot(diag(Cx))
xlabel('Elements')
ylabel('Covariances')
title('Diagonal Elements of the Covariance Matrix in Noisy Case')
%% Playing with the SVD
figure()
sig = diag(s);

title('Singular Values and PCA Modes for Noisy Case')
sig=diag(s);
subplot(3,2,1), plot(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6])
text(6,1.75*10^4,'(a)','Fontsize',[13])
xlabel('Measurements')

subplot(3,2,2), semilogy(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Ytick',[10^0 10^2 10^3 10^4 10^5],...
    'Xtick',[0 1 2 3 4 5 6]);
text(6,1.4*10^4,'(b)','Fontsize',[13])
xlabel('Measurements')

xtest = linspace(1,6,6);
subplot(3,1,2)
plot(xtest,u(:,1),'k',xtest,u(:,2),'k--',xtest,u(:,3),'k:','Linewidth',[2])
set(gca,'Fontsize',[13])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
text(5.75,0.35,'(c)','Fontsize',[13])
xlabel('Measurements')

subplot(3,1,3)
```

```
t = linspace(1,numFrames1,numFrames1);
plot(t, v(:,1),'k',t, v(:,2),'k--',t, v(:,3),'k:','Linewidth',[2])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
axis([0 numFrames1 -0.1 0.2])
text(300,0.15,'(d)','Fontsize',[13])
xlabel('Frames')

%% Horizontal Displacement
clear all; close all; clc;

load('camera_data/cam1_3.mat');
cam1 = double(vidFrames1_3);

load('camera_data/cam2_3.mat');
cam2 = double(vidFrames2_3);

load('camera_data/cam3_3.mat');
cam3 = double(vidFrames3_3);

numFrames1 = size(cam1,4);
numFrames2 = size(cam2,4);
numFrames3 = size(cam3,4);

frames = [numFrames1, numFrames2, numFrames3];

xdim = size(cam1,2);
x = linspace(1,xdim,xdim);
ydim = size(cam1,1);
y = linspace(1,ydim,ydim);

cam1g = zeros(ydim, xdim, numFrames1);
cam2g = zeros(ydim, xdim, numFrames2);
cam3g = zeros(ydim, xdim, numFrames3);

for k = 1:numFrames1
    mov1(k).cdata = vidFrames1_3(:,:,:,k);
    mov1(k).colormap = [];
    cam1g(:,:,k) = rgb2gray(vidFrames1_3(:,:,:,k));
end
for k = 1:numFrames2
    mov2(k).cdata = vidFrames2_3(:,:,:,k);
    mov2(k).colormap = [];
    cam2g(:,:,k) = rgb2gray(vidFrames2_3(:,:,:,k));
end
for k = 1:numFrames3
    mov3(k).cdata = vidFrames3_3(:,:,:,k);
    mov3(k).colormap = [];
    cam3g(:,:,k) = rgb2gray(vidFrames3_3(:,:,:,k));
end

%% Creating movies

figure(1)
for i = 1:numFrames3
```

```
    subplot(1,3,1)
    F1 = frame2im(mov1(i));
    imshow(F1);
    drawnow

    subplot(1,3,2) % i = 28 when at top
    F2 = frame2im(mov2(i+28));
    imshow(F2);
    drawnow

    subplot(1,3,3)
    F3 = frame2im(mov3(i));
    imshow(F3);
    drawnow
end

%% Synchronizing the movie files
% reset two of the movie files so they are all synchronized
mov2_disp = 28 + 1;
cam2g = cam2g(:,:,mov2_disp:end);

numFrames2 = size(cam2g,3);

frames = [numFrames1, numFrames2, numFrames3];

%% getting the positions for the first camera
figure()
x1 = zeros(1,numFrames1);
y1 = zeros(1,numFrames1);
for i = 1:numFrames1
    frame = cam1g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    width = 40;
    filt = zeros(ydim,xdim);
    filt(:,300:300+width) = frame(:,300:300+width);
    maxf = max(filt(:));
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y1(i),x1(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the second camera
figure()
x2 = zeros(1,numFrames2);
y2 = zeros(1,numFrames2);
for i = 1:numFrames2
    frame = cam2g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    filt = frame;
```

```
    maxf = max(filt(:));
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y2(i),x2(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the third camera
figure()
x3 = zeros(1,numFrames3);
y3 = zeros(1,numFrames3);
for i = 1:numFrames3
    frame = cam3g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame));
    subplot(1,2,2)
    filt = frame;
    maxf = max(filt(:));
    filt(filt<maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y3(i),x3(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% Prep the matrix for the SVD
shortest = min(frames);

x1 = x1(1:shortest);
y1 = y1(1:shortest);
x2 = x2(1:shortest);
y2 = y2(1:shortest);
x3 = x3(1:shortest);
y3 = y3(1:shortest);

X = [x1;y1;x2;y2;x3;y3];
[u,s,v] = svd(X);
lambda = diag(s).^2;
Y = u'*X;

Cx = cov(X');

figure()
plot(diag(Cx))
xlabel('Elements')
ylabel('Covariances')
title('Diagonal Elements of the Covariance Matrix in Horizontal Displacement Case')

%% Playing with the SVD
figure()
sig = diag(s);

title('Singular Values and PCA Modes for Hoizontal Displacement Case')
sig=diag(s);
subplot(3,2,1), plot(sig,'ko','Linewidth',[1.5])
```

```
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6])
text(6,1.75*10^4,'(a)','Fontsize',[13])
xlabel('Measurements')

subplot(3,2,2), semilogy(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Ytick',[10^0 10^2 10^3 10^4 10^5],...
    'Xtick',[0 1 2 3 4 5 6]);
text(6,1.4*10^4,'(b)','Fontsize',[13])
xlabel('Measurements')

xtest = linspace(1,6,6);
subplot(3,1,2)
plot(xtest,u(:,1),'k',xtest,u(:,2),'k--',xtest,u(:,3),'k:','Linewidth',[2])
set(gca,'Fontsize',[13])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
text(5.75,0.5,'(c)','Fontsize',[13])
xlabel('Measurements')

subplot(3,1,3)
t = linspace(1,shortest,shortest);
plot(t, v(:,1),'k',t, v(:,2),'k--',t, v(:,3),'k:','Linewidth',[2])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
axis([0 numFrames1 -0.1 0.2])
text(215,0.15,'(d)','Fontsize',[13])
xlabel('Frames')

%% Horizontal Displacement with Rotation
clear all; close all; clc;

load('camera_data/cam1_4.mat');
cam1 = double(vidFrames1_4);

load('camera_data/cam2_4.mat');
cam2 = double(vidFrames2_4);

load('camera_data/cam3_4.mat');
cam3 = double(vidFrames3_4);

numFrames1 = size(cam1,4);
numFrames2 = size(cam2,4);
numFrames3 = size(cam3,4);

frames = [numFrames1, numFrames2, numFrames3];

xdim = size(cam1,2);
x = linspace(1,xdim,xdim);
ydim = size(cam1,1);
y = linspace(1,ydim,ydim);

cam1g = zeros(ydim, xdim, numFrames1);
cam2g = zeros(ydim, xdim, numFrames2);
cam3g = zeros(ydim, xdim, numFrames3);
```

```
for k = 1:numFrames1
    mov1(k).cdata = vidFrames1_4(:,:,:,k);
    mov1(k).colormap = [];
    cam1g(:,:,k) = rgb2gray(vidFrames1_4(:,:,:,k));
end
for k = 1:numFrames2
    mov2(k).cdata = vidFrames2_4(:,:,:,k);
    mov2(k).colormap = [];
    cam2g(:,:,k) = rgb2gray(vidFrames2_4(:,:,:,k));
end
for k = 1:numFrames3
    mov3(k).cdata = vidFrames3_4(:,:,:,k);
    mov3(k).colormap = [];
    cam3g(:,:,k) = rgb2gray(vidFrames3_4(:,:,:,k));
end

%% Creating movies

figure(1)
for i = 1:min(frames)
    subplot(1,3,1) % i = 15 at peak
    F1 = frame2im(mov1(i));
    imshow(F1);
    drawnow

    subplot(1,3,2) % i = 19 at peak
    F2 = frame2im(mov2(i));
    imshow(F2);
    drawnow

    subplot(1,3,3) % i = 16 at peak
    F3 = frame2im(mov3(i));
    imshow(F3);
    drawnow
end

%% Synchronizing the movie files
% reset movie files so they are all synchronized
mov1_disp = 15 + 1;
mov2_disp = 19 + 1;
mov3_disp = 16 + 1;

cam1g = cam1g(:,:,mov1_disp:end);
cam2g = cam2g(:,:,mov2_disp:end);
cam3g = cam3g(:,:,mov3_disp:end);

numFrames1 = size(cam1g,3);
numFrames2 = size(cam2g,3);
numFrames3 = size(cam3g,3);

frames = [numFrames1, numFrames2, numFrames3];
shortest = min(frames);
```

```
%% getting the positions for the first camera
figure()
x1 = zeros(1,numFrames1);
y1 = zeros(1,numFrames1);
for i = 1:numFrames1
    frame = cam1g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    width = 150;
    filt = zeros(ydim,xdim);
    filt(150:end,300:300+width) = frame(150:end,300:300+width);
    maxf = max(filt(:));
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y1(i),x1(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the second camera
figure()
x2 = zeros(1,numFrames2);
y2 = zeros(1,numFrames2);
for i = 1:numFrames2
    frame = cam2g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame))
    subplot(1,2,2)
    filt = frame;
    maxf = max(filt(:));
    filt(filt < maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y2(i),x2(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% getting the positions for the third camera
figure()
x3 = zeros(1,numFrames3);
y3 = zeros(1,numFrames3);
for i = 1:numFrames3
    frame = cam3g(:,:,i);
    subplot(1,2,1)
    imshow(uint8(frame));
    subplot(1,2,2)
    filt = frame;
    maxf = max(filt(:));
    filt(filt<maxf) = 0;
    imshow(uint8(filt))
    drawnow
    [y3(i),x3(i)] = ind2sub(size(filt),find(filt == maxf,1));
end

%% Prep the matrix for the SVD
```

```
x1 = x1(1:shortest);
y1 = y1(1:shortest);
x2 = x2(1:shortest);
y2 = y2(1:shortest);
x3 = x3(1:shortest);
y3 = y3(1:shortest);

X = [x1;y1;x2;y2;x3;y3];
[u,s,v] = svd(X);
lambda = diag(s).^2;
Y = u'*X;

Cx = cov(X');

figure()
plot(diag(Cx))
xlabel('Elements')
ylabel('Covariances')
title('Diagonal Elements of the Covariance Matrix in Horizontal Disp w/ Rotation Case')

%% Playing with the SVD
figure()
sig = diag(s);

title('Singular Values and PCA Modes for Hoizontal Disp w/ Rot Case')
sig=diag(s);
subplot(3,2,1), plot(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6])
text(6,1.75*10^4,'(a)','Fontsize',[13])
xlabel('Measurements')

subplot(3,2,2), semilogy(sig,'ko','Linewidth',[1.5])
axis([0 7 0 2*10^4])
set(gca,'Fontsize',[13],'Ytick',[10^0 10^2 10^3 10^4 10^5],...
    'Xtick',[0 1 2 3 4 5 6]);
text(6,1.4*10^4,'(b)','Fontsize',[13])
xlabel('Measurements')

xtest = linspace(1,6,6);
subplot(3,1,2)
plot(xtest,u(:,1),'k',xtest,u(:,2),'k--',xtest,u(:,3),'k:','Linewidth',[2])
set(gca,'Fontsize',[13])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
text(5.75,0.5,'(c)','Fontsize',[13])
xlabel('Measurements')

subplot(3,1,3)
t = linspace(1,shortest,shortest);
plot(t, v(:,1),'k',t, v(:,2),'k--',t, v(:,3),'k:','Linewidth',[2])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
axis([0 numFrames1 -0.1 0.2])
text(355,0.15,'(d)','Fontsize',[13])
xlabel('Frames')
```