AMATH 582: Homework 3

Hyun Ah Lim
February 21, 2020

Abstract

Singular Value Decomposition (SVD) was performed on multiple data sets that were generated from video recorded by three devices in four separate test cases. Each test case involved tracking the oscillatory movement of a paint can attached to a spring. The test cases included scenarios to determine the impacts of noise and added dimensions of movement. Using the results of SVD, a Principal Component Analysis was performed to evaluate the dimensionality of the system (how many axes of motion did the object really have?). These results were compared to what we knew was occurring in the system in each case and showed that SVD was an effective method for Principal Component Analysis. However, it was also found that that factors such as the quality of data collection (i.e. noise and the accuracy of location tracking based on the images) could have significant negative impacts.

1 Introduction and Overview

A paint can with a bright light (flashlight) was attached to the end of a spring and four different scenarios of movement were recorded. In each scenario three cameras from three different vantage points recorded the event in order to track the motion of the paint can.

- Ideal: The paint can was pulled straight downward. The recording devices were held as still as possible.
- Noisy: Identical to 'Ideal' except of the recording devices were intentionally shaken to introduce 'noise'
- Horizontal displacement: The paint can is pulled down and off center and released
- Horizonal displacement plus rotation: The paint can is pulled down, off center and rotation of the can is introduced.

In each case we will use SVD to perform a Principal Component Analysis on the collected data and identify the significant dimensions of movement in each of the scenarios. This exercise will also evaluate the impacts of less than perfect data collection.

2 Theoretical Background

Matrix transformation has two effects; it rotates and scales (stretch or compress) a subject vector. Any matrix can be broken into components that reflect these two actions. Figure 1 below, taken from the referenced textbook shows two vectors, v_1 and v_2 being transformed by matrix A.

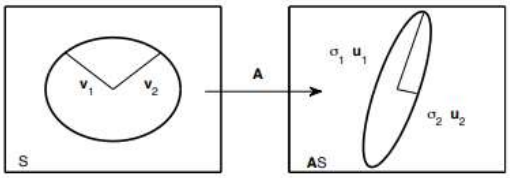

FIGURE 1. EXAMPLE WHERE VECTORS THAT COMPRISE S, ARE HIT WITH A MATRIX A, AND THE RESULTING IMPACT (TEXTBOOK[1])

The stretch factors are represented by σ_1 and σ_2, the resulting rotation by new vectors u_1 and u_2. With this initial concept, we can discuss Singular Value Decomposition (SVD), which can diagonalize and decompose any matrix into component parts, depicted in the equivalent equation 1 and figure 2:

$$A\,V = U\,\Sigma \qquad \text{(eq 1)}$$

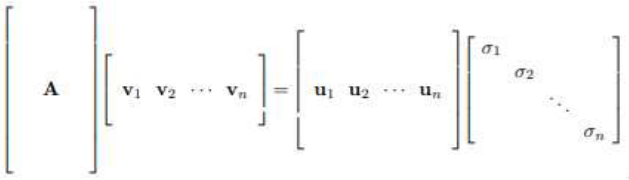

FIGURE 2. MATRIX REPRESENTATION OF THE SINGULAR VALUE DECOMPOSITION ELEMENTS OF MATRIX A (TEXTBOOK [1])

Then we can then solve for A, and we get the full SVD (eq 2):

$$A = U\,\Sigma\,V^T \tag{eq 2}$$

U $\mathbb{C}^{m\times m}$ is unitary
V $\mathbb{C}^{m\times n}$ is unitary
$\Sigma \epsilon\ \mathbb{R}^{m\times n}$ is diagonal

The relationship between SVD and eigenvectors and eigenvalues follows: Given matrix A and the SVD components above, U and V give you the eigenvectors and eigenvalues of the covariance matrix of A. The formula for a covariance matrix of A is shown below (eq 3).

$$C_A = \frac{1}{(\mathrm{n}-1)} AA^T \tag{eq 3}$$

A covariance matrix identifies redundant data by comparing the correlation/variance between data sets, thus we can see how SVD, can do the same. It determines the rank of a matrix by identifying the number of non-zero σ values in our diagonal matrix Σ. A theorem states if the rank of A is r, then there are r non-zero σ values. This method enables us to remove redundancy in a data set by identify those signals with zero variance. In addition, another theorem states that A is the sum of r rank one matrices and they are ordered by size. Therefore, if a matrix has a rank of 3, then you just need the first three rank dimensions to describe the entire system. Another theorem states that this will always provide the best approximation (in an L2 norm sense). Therefore in the real world, where full rank is the norm due to noise, etc. you can use the singular values for each mode to best approximate a system. This 'rank approximation', allows you to determine the important modes (rank dimensions) that are the principal components. That means reducing the dimensionality of a matrix and establishing its important features space.

In addition, one of the powerful attributes of SVD is that it can be performed on any matrix regardless of its properties, if performed using two bases (U and V as shown in the eq 3). A theorem states that every matrix $A \epsilon\ \mathbb{C}^{m\times n}$ has a singular value decomposition. This is superior to eigenvector decomposition, in the sense that, eigenvector decomposition can only be performed on diagonalizable matrices (square and symmetric); SVD has no such constraint, which makes it much more useful in real world applications where data sets do not come symmetric and square.

3 Algorithm Implementation and Development

The algorithm for all four scenarios were performed in a similar fashion and are outlined below:

3.1 Supervised Filtering around area of activity

After each video was loaded, it was visually reviewed and the area of motion of the paint can for the entire duration of the video was estimated. A step filter along the x and y axes were combined to create a box filter around the scoped area of motion. Each frame was then converted to grayscale, converted to a double format and multiplied by the box filter to reduce all values outside the boxed area to zero.

3.2 Motion detection using Pixel value

The values in each frame were then normalized so that the brightest shade equaled one. Then based on the results, either values equal to or close to one were used to approximate the location of the bright light set on

top of the tracked object (paint can). Then, also based on what provided the best results (based on what was expected, either the minimum or mean values of these identified pixel positions were extracted for the x and y locations of the paint.

3.3 Synchronize the video data

Each x and y pixel location vector created in section 3.2 had leading and/or ending elements (i.e. frames) removed to synchronize video and equalize frame count across all vectors.

3.4 Center the x and y location values

The displacement of the paint can was calculated by subtracting from each element, the mean value of its row.

3.5 Construct matrix A

Use the repelem() function to stack the processed x and y vectors from the previous step.

3.6 Execute SVD function on A' matrix

Use the svd() function on A' to produce the U, Σ and V matrices. Note that A is transposed for six columns (x and y for each of three cameras) and the number of rows based on the frame count used for each case.

3.7 Sigma values

Extract the sigma values from the diagonal Σ matrix (using the diag() function). Plot and calculate their component energy to determine the significance of each mode (1-6) and identify principal components.

3.8 U and V columns plotted for Principal Components

Plot the significant modes (columns) of both the left (U) and right (V) unitary matrices.

4 Computation Results

4.1 Object (paint can) displacement results

After supervised filtering for filtering and pixel value detection, the following x and y displacement measurements were made. The first two test cases are show in figure 2 and the last two, in figure 3.

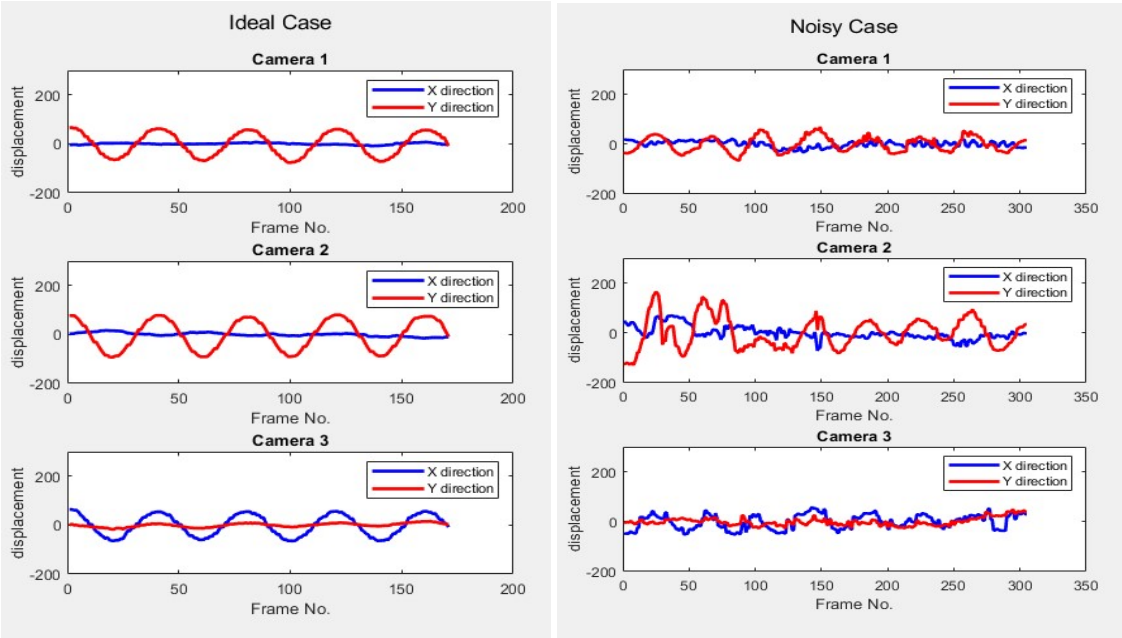

FIGURE 2. THE IDEAL CASE (LEFT). THE X AND Y DISPLACEMENT OF THE PAINT CAN FROM THE PERSPECTIVE OF THE THREE RECORDING DEVICES. OSCILLATIONS CAN BE SEEN IN A SINGLE DIRECTION FOR EACH CAMERA. THE NOISY CASE (RIGHT) ALSO SHOWS OSCILLATIONS IN THE SAME CORRESPONDING AXES FOR EACH CAMERA HOWEVER THE 'NOISE' PRODUCED BY THE SHAKING OF THE CAMERA IS EVIDENT.

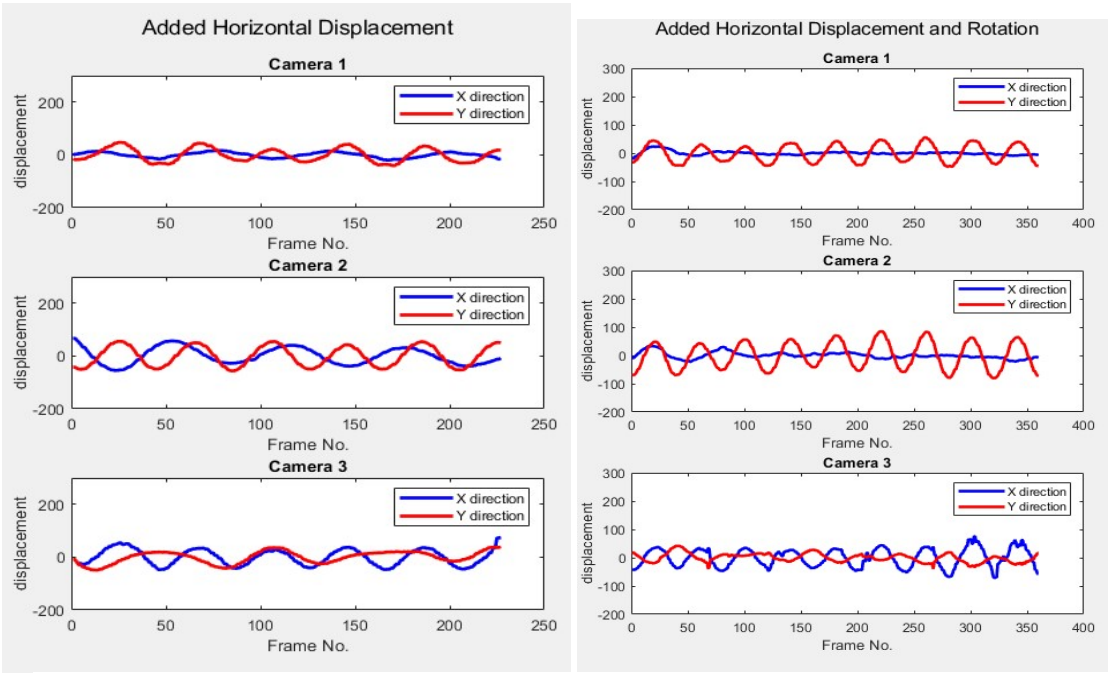

FIGURE 3. ADDED HORIZONAL DISPLACEMENT (LEFT). THE X AND Y DISPLACEMENT FROM THE PERSPECTIVE OF THE THREE RECORDING DEVICES WHEN HORIZONTAL (PENDULUM) MOTION IS ADDED TO THE VERTICAL DISPLACEMENT OF THE IDEAL CASE. ON THE RIGHT, ROTATION IS ADDED, IN ADDITION TO THE VERTICAL AND PENDULUM MOTION.

4.2 Principal Component Analysis via Singular Value Decomposition ($A = U\Sigma V^T$)

SVD was performed on matrix A in all four test cases. Below are the explicit sigma values from the diagonal Σ matrix show in tabular format in each of the four test cases (tables 1-4). Included in are the corresponding portion of energy they comprise in the system.

TABLE 1: IDEAL CASE

	Value	Energy %	Cumulative Energy %
σ_1	1132	78%	78%
σ_2	119	8%	87%
σ_3	67	5%	91%
σ_4	54	4%	95%
σ_5	44	3%	98%
σ_6	28	2%	100%

TABLE 2: NOISY CASE

	Value	Energy %	Cumulative Energy %
σ_1	1189	42%	42%
σ_2	511	18%	60%
σ_3	408	14%	75%
σ_4	316	11%	86%
σ_5	216	8%	93%
σ_6	185	7%	100%

TABLE 3: ADDED HORIZONTAL

	Value	Energy %	Cumulative Energy %
σ_1	796	43%	43%
σ_2	511	27%	70%
σ_3	305	16%	86%
σ_4	120	6%	93%
σ_5	76	4%	97%
σ_6	59	3%	100%

TABLE 4: ADDED HORIZONTAL & ROTATION

	Value	Energy %	Cumulative Energy %
σ_1	1201	58%	58%
σ_2	260	13%	71%
σ_3	235	11%	83%
σ_4	181	9%	91%
σ_5	107	5%	97%
σ_6	70	3%	100%

And then we can plot these same sigma values below along with what we consider the principal components of U and V based on these values (figures 4).

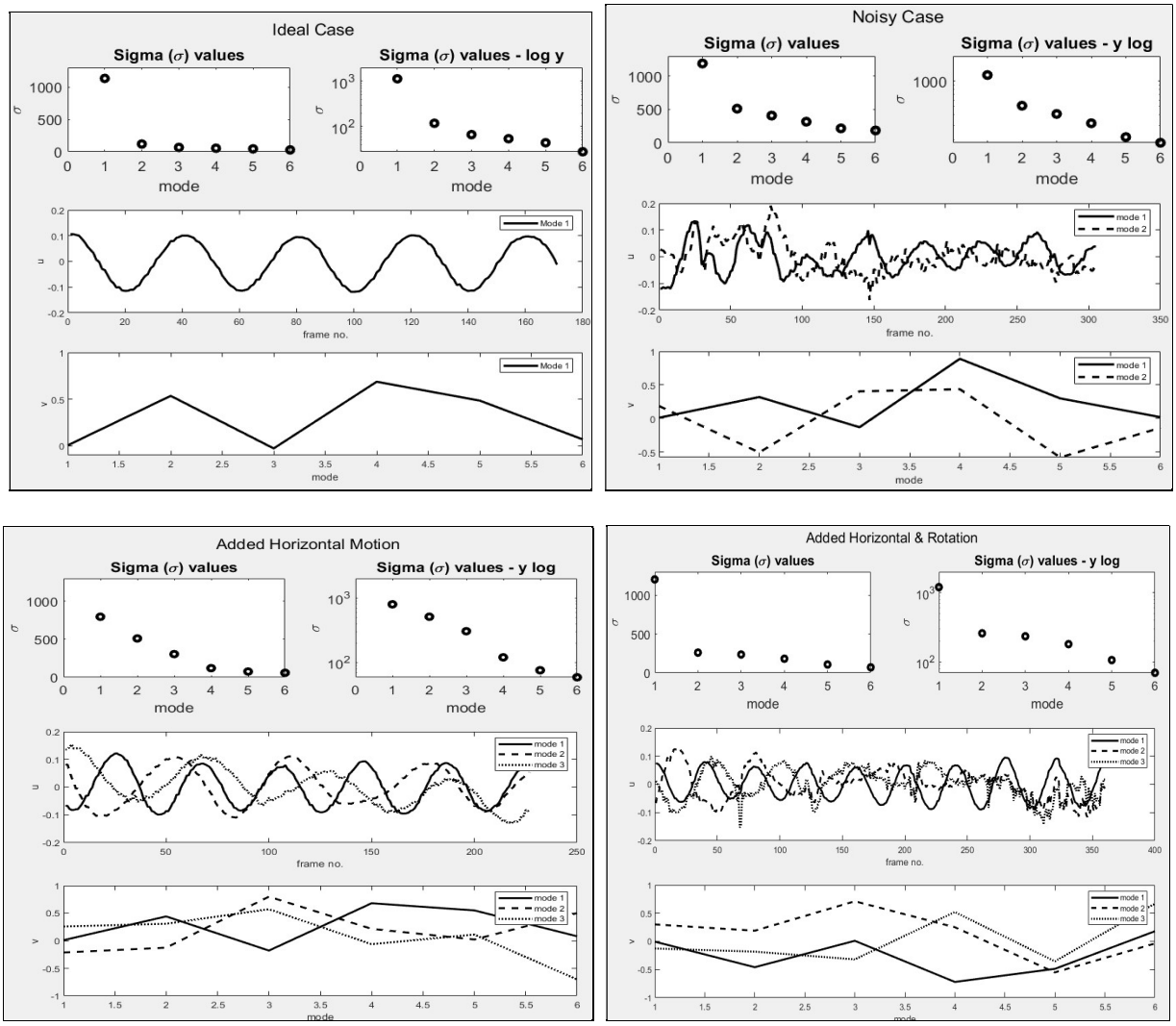

FIGURE 4: THE SIGMA VALUES ARE PLOTTED IN THE TOP ROW OF EACH CASE SECTION. THE SECOND AND THIRD ROWS IN EACH ARE THE PLOTS FOR THE PRINCIPAL COMPONENTS (MODES/COLUMNS) FOR U (DISPLACEMENT IN TIME) AND V (THE ORTHONORMAL BASIS).

In the plots above (figure 5) we attempted to use the sigma values to determine the principal components of this matrix. In each case, the bottom two plots represent the modes (columns) in the matrices U and V, respectively, that correspond to sigma values that indicate significance in this system. In the Ideal case, we reduce the six modes, to a single mode, that corresponds to 78% of the energy in the system that represents the dominant displacement at the Z axis. The Noisy shows a single sigma value that is larger, but the noise introduced by shaking does reduce its dominance. In the horizontal and horizontal plus rotation cases, we see additional modes becoming significant due to the additional horizontal (pendulum) and rotation (spinning) of the paint can. Although we would expect only two significant modes in the Horizontal Motion case, we see three possibly due to a slight rotation of the paint can and its light.

5 Summary and Conclusions

The high dimensionality of some data and the quality of its collection has a large impact on the ability to efficiency analyze a system. Therefore, methods that act to diagonalize such large and noisy datasets are effective tools in not only recognizing redundant data and enabling one to reduce a system's dimensionality but also in being able to recognize, even when disrupted by noise, which dimension or dimensions are most important in understanding the behavior of that system. By evaluating the singular values that result from SVD, the importance of each mode can be quantified (σ) and described (U, V). This enables the observer to

determine how much data is really required to understand a system at the desired level. This is what was performed in each test case and the results that were found:

- Ideal: A single dominant mode was found to describe 78% of the energy in the system. Energy found in other modes was likely due to imperfect motion of the can and less than perfect method of the data collection (using pixel strength and mean values).
- Noisy: Although we found that the energy in other dimensions were increased, the one mode that contained the oscillating displacement was mode 1 when plotted. So although noise, masked the true significance of the vertical movement, SVD allowed us to isolate it as the top behavior. (40% of the energy).
- Horizontal displacement: In this case, I expected only two principal components (motion in z axis and horizontal displacement), however based on the σ values, three were found. Again likely due to poor data collection (pixel strength tracking and syncing of video from three sources). Also the motion was likely not perfectly in two dimensions only. Three modes encompassed 93% of the systems energy.
- Horizontal displacement plus rotation: In this case we found that the vertical (z axis) displacement was even more significant than the case above. Three modes were plotted which encompassed 91% of the system's behavior.

References

[1] J. Nathan Kutz 2013, Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data, Oxford University Press

Appendix A: Relevant MATLAB functions

1. repelem(): replicates elements of an array. This function was used to construct both the x and y components of the box filter for the image.

2. horzcat(): Horizontally concatenates. This was used to stack the truncated x and y vectors to build the A matrix.

3. svd(): This function was is used to move the 0 values back to the center after the fftn() function was used. Does not affect processing but allows us to better visualize the frequency domain

4. diag(): This is the N dimensional Inverse discrete fast Fourier Transform function. This was used to transform data back from the spectral domain to the spatial domain.

5. ind2sub(): This reverses the fftshift() that was performed in the spectral domain, back to the original frequency coordinate order where the 0 values are at the outer edges. This should be performed prior to executing the ifftn() to transform back to the spatial domain.

6. find(): function used to plot a matrix using color. This function was used to generate the spectrogram.

7. min()/mean(): calculates minimum and mean values, was used for various calculations in the algorithm.

8. repmat(): replicate and tile and array. This was used to center the x and y location around the mean and set it to zero to effectively convert the measurement to a displacement around the mean.

9. length(): provides length of a vector. Used this to count frames from the videos

10. find(): find the index values of a specific non-zero value in an array

11. 'for loop' (honorable mention): This isn't a function but the 'for loop' allowed us to perform iterations across all 20 measurements with minimal coding.

Appendix B: MATLAB Code

```
clear all; close all; clc

load cam1_1.mat
load cam2_1.mat
load cam3_1.mat

xs = [1: 640];
ys =[1:480];

%% Cam 1
cam1 = vidFrames1_1;
f1x =repelem((abs(xs-322) <= 32), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 290) <= 90, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam1(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9)
for jj = 1: frame_count
   frame = double(rgb2gray(cam1(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam1_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c1 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam1_fn(:,:,:,kk);
   I = find(frame == 1);
   [y,x] = ind2sub(size(frame), I);
   y = min(y);
   x = mean(x);
  c1(kk, :) = [-x,-y]';
end

%% Cam 2
cam2= vidFrames2_1;
f1x =repelem((abs(xs-305) <= 35), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 200) <= 100, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam2(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam2(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam2_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c2 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam2_fn(:,:,:,kk);
   I = find(frame > .97);
   [y,x] = ind2sub(size(frame), I);
   y = min(y);
```

```
   x = median(x);
  c2(kk, :) = [-x,-y]';
end

%% Cam 3
cam3= vidFrames3_1;
f1x =repelem((abs(xs-350) <= 80), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 270) <= 25, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam3(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam3(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam3_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c3= zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam3_fn(:,:,:,kk);
   I = find(frame == 1);
   [y,x] = ind2sub(size(frame), I);
   y = min(y);
   x = min(x);
  c3(kk, :) = [-x,-y]';
end

%% truncate x,y from each cam to match (4.5 cycles)
x1 = c1(10:180, 1) ; y1 = c1(10:180, 2); %cam1  30-170
x2 = c2(20:190, 1) ; y2 = c2(20:190, 2); %cam2  40-180
x3 = c3(10:180, 1) ; y3 = c3(10:180, 2); %cam2 30-170
A = horzcat(x1, y1, x2, y2, x3, y3)';  % Matrix for SVD
%% calculate displacement
[m,n]=size(A); %compute data size
mn=mean(A,2); %compute mean for each row
A=A-repmat(mn, 1, n); % subtract mean
fs = 1:length(A);
r1 = rank(A); %full rank due to noise
covA = cov(A); % covariance matrix
%
figure(1)
sgtitle('Ideal Case')
subplot(3,1,1); plot(fs', A(1,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 1')
plot(fs', A(2,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,2); plot(fs', A(3,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 2')
plot(fs', A(4,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
```

```
ylim([-200 300])
subplot(3,1,3); plot(fs', A(5,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 3')
plot(fs', A(6,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])

%%
[u,s,v]=svd(A');
sig = diag(s);
lambda = sig.^2;
Y = u*A';

energy1 = sig(1)/sum(sig);
energy2 = sum(sig(1:2))/sum(sig);
energy3 = sum(sig(1:3))/sum(sig);
energy4 = sum(sig(1:4))/sum(sig);
energy5 = sum(sig(1:5))/sum(sig);
energy6 = sum(sig(:))/sum(sig);
modes =1:6;
%%
figure(2)
subplot(3,2,1), plot(sig,'ko','Linewidth',[3])
sgtitle('Ideal Case', 'Fontsize', [15]);
title('Sigma (\sigma) values' ,'Fontsize', [10]);
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6 7])
ylim([0 1300]);
xlabel('mode')
ylabel('\sigma')

subplot(3,2,2), semilogy(sig,'ko','Linewidth',[3])
title('Sigma (\sigma) values - log y', 'Fontsize', [10]);
set(gca,'Fontsize',[13],'Ytick',[10^(0) 10^(1) 10^(2) 10^3 10^4 10^5 10^6 10^7],...
  'Xtick',[0 1 2 3 4 5 6]);
ylim([0 2000]);
xlabel('mode')
ylabel('\sigma')
subplot(3,1,2)
plot(fs,u(:,1),'k','Linewidth',[2])
ylim([-0.2 0.2])
ylabel('u');
xlabel('frame no.')
legend('Mode 1')
subplot(3,1,3)
plot(modes,v(:,1),'k','Linewidth',[2])
ylabel('v');
xlabel('mode')
legend('Mode 1')
ylim([-0.1 1])

 %Noisy
clear all; close all; clc
load cam1_2.mat
load cam2_2.mat
load cam3_2.mat
```

```
xs = [1: 640];
ys =[1:480];

%% Cam 1
cam1 = vidFrames1_2;
f1x =repelem((abs(xs-350) <= 40), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 320) <= 100, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam1(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9)
for jj = 1: frame_count
   frame = double(rgb2gray(cam1(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam1_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c1 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam1_fn(:,:,:,kk);
   I = find(frame >.96);
   [y,x] = ind2sub(size(frame), I);
   y = mean(y);
   x = mean(x);
  c1(kk, :) = [-x,-y]';
end

%% Cam 2
cam2= vidFrames2_2;
f1x =repelem((abs(xs-300) <= 75), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 270) <= 250, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam2(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam2(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam2_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c2 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam2_fn(:,:,:,kk);
   I = find(frame>.98);
   [y,x] = ind2sub(size(frame), I);
   y = mean(y);
   x = mean(x);
  c2(kk, :) = [-x,-y]';
end
%%
% Cam 3
cam3= vidFrames3_2;
f1x =repelem((abs(xs-350) <= 100), 480, 1) ; %create filter along vert axis
```

```
f1y = repelem(abs(ys- 250) <= 40, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam3(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam3(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam3_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c3= zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam3_fn(:,:,:,kk);
   I = find(frame>.97);
   [y,x] = ind2sub(size(frame), I);
    y = mean(y);
   x = mean(x);
  c3(kk, :) = [-x,-y]';
end
% truncate x,y from each cam to match (4.5 cycles)
x1 = c1(10:314, 1) ; y1 = c1(10:314, 2); %cam1  13-314
x2 = c2(36:340, 1) ; y2 = c2(36:340, 2); %cam2  14-34
x3 = c3(15:319, 1) ; y3 = c3(15:319, 2); %cam2 20-327
A = horzcat(x1, y1, x2, y2, x3, y3)';  % Matrix for SVD
% calculate displacement
[m,n]=size(A); %compute data size
mn=mean(A,2); %compute mean for each row
A=A-repmat(mn, 1, n); % subtract mean
fs = 1:length(A);
r1 = rank(A); %full rank due to noise
covA = cov(A); % covariance matrix

figure(1)
sgtitle('Noisy Case')
subplot(3,1,1); plot(fs', A(1,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 1')
plot(fs', A(2,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,2); plot(fs', A(3,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 2')
plot(fs', A(4,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,3); plot(fs', A(5,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 3')
plot(fs', A(6,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
```

```
%
[u,s,v]=svd(A');
sig = diag(s);
lambda = sig.^2;
Y = u*A';

energy1 = sig(1)/sum(sig);
energy2 = sum(sig(1:2))/sum(sig);
energy3 = sum(sig(1:3))/sum(sig);
energy4 = sum(sig(1:4))/sum(sig);
energy5 = sum(sig(1:5))/sum(sig);
energy6 = sum(sig(:))/sum(sig);modes =1:6;
%%
figure(2)
subplot(3,2,1), plot(sig,'ko','Linewidth',[3])
sgtitle('Noisy Case', 'Fontsize', [15]);
title('Sigma (\sigma) values');
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6 7])
ylim([0 1300]);
xlabel('mode')
ylabel('\sigma')


subplot(3,2,2), semilogy(sig,'ko','Linewidth',[3])
title('Sigma (\sigma) values - y log');
set(gca,'Fontsize',[13],'Ytick',[10^(0) 10^(1) 10^(2) 10^3 10^4 10^5 10^6 10^7],...
  'Xtick',[0 1 2 3 4 5 6]);
ylim([0 2000]);
xlabel('mode')
ylabel('\sigma')
subplot(3,1,2)
plot(fs,u(:,1),'k','Linewidth',[2]); hold on;
plot(fs,u(:,2),'k--','Linewidth',[2]);
ylim([-0.2 0.2])
ylabel('u');
xlabel('frame no.')
legend('mode 1', 'mode 2')
subplot(3,1,3)
plot(modes,v(:,1),'k','Linewidth',[2]); hold on;
plot(modes,v(:,2),'k--','Linewidth',[2]);
ylabel('v');
xlabel('mode')
legend('mode 1', 'mode 2')
```

%Horizonal Displacement

```
clear all; close all; clc
load cam1_3.mat
load cam2_3.mat
load cam3_3.mat

xs = [1: 640];
ys =[1:480];

%% Cam 1
cam1 = vidFrames1_3;
```

```
f1x =repelem((abs(xs-320) <= 50), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 295) <= 90, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam1(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9)
for jj = 1: frame_count
   frame = double(rgb2gray(cam1(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam1_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c1 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam1_fn(:,:,:,kk);
   I = find(frame >.97);
   [y,x] = ind2sub(size(frame), I);
   y = mean(y);
   x = mean(x);
  c1(kk, :) = [-x,-y]';
end

%% Cam 2
cam2= vidFrames2_3;
f1x =repelem((abs(xs-290) <= 100), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 255) <= 145, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam2(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam2(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam2_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c2 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam2_fn(:,:,:,kk);
   I = find(frame>.97);
   [y,x] = ind2sub(size(frame), I);
   y = mean(y);
   x = mean(x);
  c2(kk, :) = [-x,-y]';
end
%%
% Cam 3
cam3= vidFrames3_3;
f1x =repelem((abs(xs-350) <= 100), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 260) <= 55, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam3(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
```

```
    frame = double(rgb2gray(cam3(:,:,:,jj)));
    frame_f = filter1.*frame;
    frame_fn = frame_f/max(max(frame_f));
    cam3_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c3= zeros(2, frame_count).';
for kk = 1:frame_count
    frame = cam3_fn(:,:,:,kk);
    I = find(frame >.97);
    [y,x] = ind2sub(size(frame), I);
     y = mean(y);
    x = mean(x);
   c3(kk, :) = [-x,-y]';
end
%% truncate x,y from each cam to match (4.5 cycles)
x1 = c1(13:239, 1) ; y1 = c1(13:239, 2); %cam1  13-314
x2 = c2(1:227, 1) ; y2 = c2(1:227, 2); %cam2  14-34
x3 = c3(5:231, 1) ; y3 = c3(5:231, 2); %cam2 20-327
A = horzcat(x1, y1, x2, y2, x3, y3)';  % Matrix for SVD
% calculate displacement
[m,n]=size(A); %compute data size
mn=mean(A,2); %compute mean for each row
A=A-repmat(mn, 1, n); % subtract mean
fs = 1:length(A);
r1 = rank(A); %full rank due to noise
covA = cov(A); % covariance matrix

figure(1)
sgtitle('Added Horizontal Displacement')
subplot(3,1,1); plot(fs', A(1,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 1')
plot(fs', A(2,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,2); plot(fs', A(3,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 2')
plot(fs', A(4,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,3); plot(fs', A(5,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 3')
plot(fs', A(6,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])

%
[u,s,v]=svd(A');
sig = diag(s);
lambda = sig.^2;
Y = u*A';
```

```
energy1 = sig(1)/sum(sig);
energy2 = sum(sig(1:2))/sum(sig);
energy3 = sum(sig(1:3))/sum(sig);
energy4 = sum(sig(1:4))/sum(sig);
energy5 = sum(sig(1:5))/sum(sig);
energy6 = sum(sig(:))/sum(sig);
modes =1:6;
%%
figure(2)
subplot(3,2,1), plot(sig,'ko','Linewidth',[3])
sgtitle('Added Horizontal Motion', 'Fontsize', [15]);
title('Sigma (\sigma) values');
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6 7])
ylim([0 1300]);
xlabel('mode')
ylabel('\sigma')


subplot(3,2,2), semilogy(sig,'ko','Linewidth',[3])
title('Sigma (\sigma) values - y log');
set(gca,'Fontsize',[13],'Ytick',[10^(0) 10^(1) 10^(2) 10^3 10^4 10^5 10^6 10^7],...
  'Xtick',[0 1 2 3 4 5 6]);
ylim([0 2000]);
xlabel('mode')
ylabel('\sigma')
subplot(3,1,2)
plot(fs,u(:,1),'k','Linewidth',[2]); hold on;
plot(fs,u(:,2),'k--','Linewidth',[2]);
plot(fs,u(:,3),'k:','Linewidth',[2]);
ylim([-0.2 0.2])
ylabel('u');
xlabel('frame no.')
legend('mode 1', 'mode 2', 'mode 3')
subplot(3,1,3)
plot(modes,v(:,1),'k','Linewidth',[2]); hold on;
plot(modes,v(:,2),'k--','Linewidth',[2]);
plot(modes,v(:,3),'k:','Linewidth',[2]);
ylabel('v');
xlabel('mode')
legend('mode 1', 'mode 2', 'mode 3')
 %Horizonal Displacement and Rotation

clear all; close all; clc
load cam1_4.mat
load cam2_4.mat
load cam3_4.mat

xs = [1: 640];
ys =[1:480];

%% Cam 1
cam1 = vidFrames1_4;
f1x =repelem((abs(xs-390) <= 65), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 285) <= 110, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam1(1,1,1,:));
```

```
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9)
for jj = 1: frame_count
   frame = double(rgb2gray(cam1(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam1_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c1 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam1_fn(:,:,:,kk);
   I = find(frame >.97);
   [y,x] = ind2sub(size(frame), I);
   y = mean(y);
   x = mean(x);
  c1(kk, :) = [-x,-y]';
end

%% Cam 2
cam2= vidFrames2_4;
f1x =repelem((abs(xs-310) <= 100), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 230) <= 145, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam2(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam2(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam2_fn(:,:,:,jj) = frame_fn;
end
% find max value in each frame (end with 1x226 plot)
c2 = zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam2_fn(:,:,:,kk);
   I = find(frame>.97);
   [y,x] = ind2sub(size(frame), I);
   y = mean(y);
   x = mean(x);
  c2(kk, :) = [-x,-y]';
end
%%
% Cam 3
cam3= vidFrames3_4;
f1x =repelem((abs(xs-390) <= 100), 480, 1) ; %create filter along vert axis
f1y = repelem(abs(ys- 215) <= 70, 640, 1).'; %create filter along horiz axis
filter1 = f1x.*f1y; %final filter
frame_count = length(cam3(1,1,1,:));
ts = [1:frame_count];
% filter all frames (movement starts at 8 or 9
for jj = 1: frame_count
   frame = double(rgb2gray(cam3(:,:,:,jj)));
   frame_f = filter1.*frame;
   frame_fn = frame_f/max(max(frame_f));
   cam3_fn(:,:,:,jj) = frame_fn;
```

```
end
% find max value in each frame (end with 1x226 plot)
c3= zeros(2, frame_count).';
for kk = 1:frame_count
   frame = cam3_fn(:,:,:,kk);
   I = find(frame >.96);
   [y,x] = ind2sub(size(frame), I);
    y = mean(y);
   x = mean(x);
  c3(kk, :) = [-x,-y]';
end
%% truncate x,y from each cam to match (4.5 cycles)
x1 = c1(33:392, 1) ; y1 = c1(33:392, 2); %cam1  13-314
x2 = c2(40:399, 1) ; y2 = c2(40:399, 2); %cam2  14-34
x3 = c3(32:391, 1) ; y3 = c3(32:391, 2); %cam2 20-327
A = horzcat(x1, y1, x2, y2, x3, y3)';  % Matrix for SVD
% calculate displacement
[m,n]=size(A); %compute data size
mn=mean(A,2); %compute mean for each row
A=A-repmat(mn, 1, n); % subtract mean
fs = 1:length(A);

figure(1)
sgtitle('Added Horizontal Displacement and Rotation')
subplot(3,1,1); plot(fs', A(1,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 1')
plot(fs', A(2,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,2); plot(fs', A(3,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 2')
plot(fs', A(4,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
subplot(3,1,3); plot(fs', A(5,:), 'b', 'Linewidth', [2]);  hold on;
title('Camera 3')
plot(fs', A(6,:), 'r', 'Linewidth', [2])
legend('X direction','Y direction', 'Location', 'NorthEast')
xlabel('Frame No.')
ylabel('displacement');
ylim([-200 300])
%
[u,s,v]=svd(A');
sig = diag(s);
lambda = sig.^2;
Y = u*A';
energy1 = sig(1)/sum(sig);
energy2 = sum(sig(1:2))/sum(sig);
energy3 = sum(sig(1:3))/sum(sig);
energy4 = sum(sig(1:4))/sum(sig);
energy5 = sum(sig(1:5))/sum(sig);
energy6 = sum(sig(:))/sum(sig);
modes = 1:6;
```

```
%%
figure(2)
subplot(3,2,1), plot(sig,'ko','Linewidth',[3])
sgtitle('Added Horizontal & Rotation ', 'Fontsize', [15]);
title('Sigma (\sigma) values');
set(gca,'Fontsize',[13],'Xtick',[0 1 2 3 4 5 6 7])
ylim([0 1300]);
xlabel('mode')
ylabel('\sigma')


subplot(3,2,2), semilogy(sig,'ko','Linewidth',[3])
title('Sigma (\sigma) values - y log');
set(gca,'Fontsize',[13],'Ytick',[10^(0) 10^(1) 10^(2) 10^3 10^4 10^5 10^6 10^7],...
  'Xtick',[0 1 2 3 4 5 6]);
ylim([0 2000]);
xlabel('mode')
ylabel('\sigma')
subplot(3,1,2)
plot(fs,u(:,1),'k','Linewidth',[2]); hold on;
plot(fs,u(:,2),'k--','Linewidth',[2]);
plot(fs,u(:,3),'k:','Linewidth',[2]);
ylim([-0.2 0.2])
ylabel('u');
xlabel('frame no.')
legend('mode 1', 'mode 2', 'mode 3')
subplot(3,1,3)
plot(modes,v(:,1),'k','Linewidth',[2]); hold on;
plot(modes,v(:,2),'k--','Linewidth',[2]);
plot(modes,v(:,3),'k:','Linewidth',[2]);
ylabel('v');
xlabel('mode')
legend('mode 1', 'mode 2', 'mode 3')
```