John Curtis

Assignment 3: PCA and a Spring-Mass System

Abstract

This paper explores using the singular value decomposition and principal component analysis to determine the oscillation behaviour of a spring mass system. There were 4 different test cases: ideal linear oscillation case, linear oscillation with noise case, added horizontal displacement resulting in multi-dimensional oscillation, multi-dimensional oscillation with rotation. The PCA was used to find that for all 4 cases, the rank 1 approximation contained most of the information. This was particularly interesting for cases 3 and 4 where a higher rank approximation was expected due to the multi-dimensional oscillation.

Introduction and Overview

Principal component analysis (PCA) utilises the singular value decomposition (SVD) to break down some data into the main orthonormal components it is comprised of. PCA can be used to determine behaviour from large sets of data and to reduce the information to just that of the principal components. Here we will take position data of an oscillating mass on a spring from 3 cameras in different scenarios, and then use the PCA to find the underlying behaviour of the spring mass system without any prior physics knowledge.

Theoretical Background

According to Hooke's Law, a spring mass system is governed by the equation:

$$z(t) = A\cos(\omega t + \varphi) \quad Eq.1$$

where z is the position of the mass, A is the amplitude of oscillation, ω is the frequency of the oscillation and φ is the phase. We will use this to check our results later but suppose we do not know this and we want to find the behaviour of the spring mass system, we can then use PCA to determine the behaviour.

Firstly, any matrix $\boldsymbol{A}$ can be decomposed using the SVD into three matrices $\boldsymbol{A} = \boldsymbol{U\Sigma V}^*$ where $\boldsymbol{\Sigma}$ is a diagonal matrix of values σ called singular values and $\boldsymbol{U}$ and $\boldsymbol{V}$ are matrices where the columns make up the left and right singular vectors respectively. These singular vectors $\boldsymbol{u}$ and $\boldsymbol{j}$ are orthogonal and can be used to create a low rank approximation of $\boldsymbol{A}$, the equation for which is:

$$\boldsymbol{A}_N = \sum_{j=1}^{N} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_v^* \quad Eq.2$$

where $1 < N < rank(\boldsymbol{A})$ is the rank of the approximation.

PCA utilises the SVD to find the orthogonal/independent components that the data varies along. Using the low rank approximation, PCA can determine the bases, represented by the singular vectors, for the variation in the data and in that basis. Assuming the data is a discrete representation of some underlying function, these bases correspond to the proper orthogonal modes of the

function, and expanding the function using these modes is what is called the proper orthogonal decomposition (POD).

To determine which of these principal component bases corresponds to the greatest and primary variation in the data of $\boldsymbol{A}$, we can calculate the energy of the rank N approximation as follows:

$$energy_N = \frac{\sum_{i=1}^{N} \sigma_i^2}{\sum_{j=1}^{r} \sigma_j^2} \quad Eq. 3$$

where N is the rank of the approximation, σ is the singular value and r is the rank of $\boldsymbol{A}$. The greater the energy of the approximation in the basis, the more information is captured in that basis and the better the approximation is. Thus the principal components of the data where most of the variation lies are the singular bases corresponding to the highest energy approximations. Conveniently in the SVD, the singular values and vectors are organised in order of highest to lowest energy, making it easier to find the principal components.

Algorithm Implementation and Development

I began by converting 3 the videos in each test scenario to matrices and truncating them so the position of the mass lined up for all 3 cameras at the same time.

Then to track the position of the mass I first created a gaussian filter function centred at the approximate location of the mass throughout the oscillation to reduce the brightness of pixels other than the pixels of the bright light on the mass. The filter was made to be relatively wide so it does not reduce the brightness of the light on the mass.

Then I iterated through each camera's data, and then all of the frames for the data. At each iteration I converted the matrix representing the frame to grayscale, multiplied it by the filter and then performed a matrix convolution with a 15×15 matrix with each value being $\frac{1}{15^2}$ so that each value in the new matrix represents a rough average of the 15 values/pixels around it. The number 15 was used as it was found to be the approximate size of the light on the mass in pixels. This made it so that the matrix has a greater value if it is in a cluster of bright pixels with large values, making it easier to find the light on the mass without much noise from bright pixels in the background.

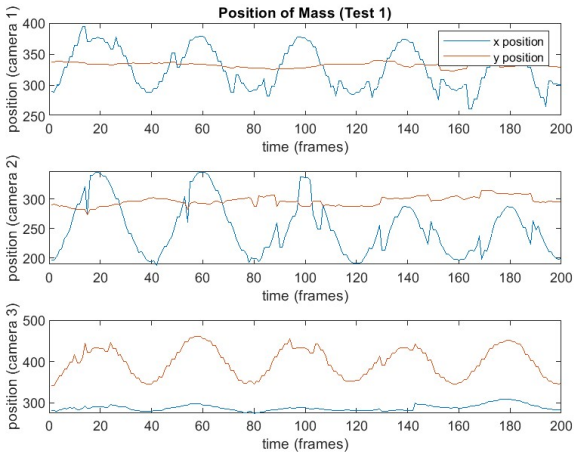
I took the maximum value of this resulting matrix to find the position of the mass in that frame. I also checked whether the brightness value was above a threshold and if it wasn't, I set the position value to be the same as the previous frame to again reduce the effect of the background. Then after iterating through each frame an camera data, I ended up with a $6 \times (number\ of\ frames)$ matrix where the rows represent the x and y positions of the mass for each camera, and the columns represent the time in frames.

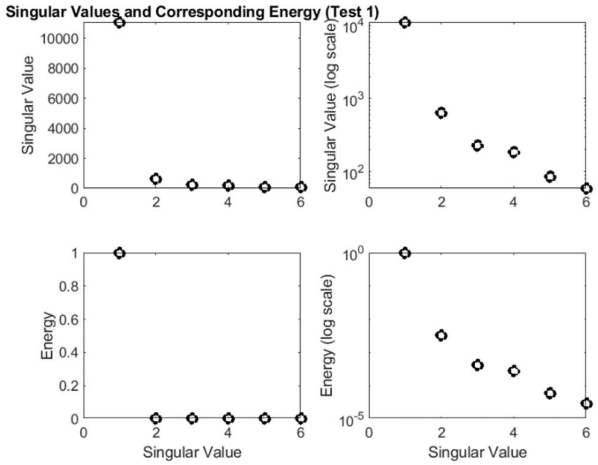
After that I performed the SVD on this position matrix and calculated the energy of the singular values using equation 3. Then I found the first few POD modes by plotting the singular vectors against time in frames. Then using the information of the energy, I calculated the most appropriate rank approximation of the data using equation 2.

I repeated the above steps for all 4 of the test scenarios (ideal case, noisy case, horizontal displacement, horizontal displacement and rotation). Allowing me to analyse the principal components in each situation.

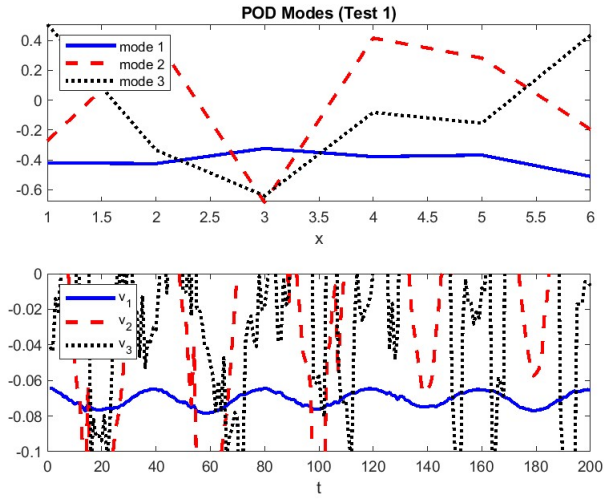
Computational Results

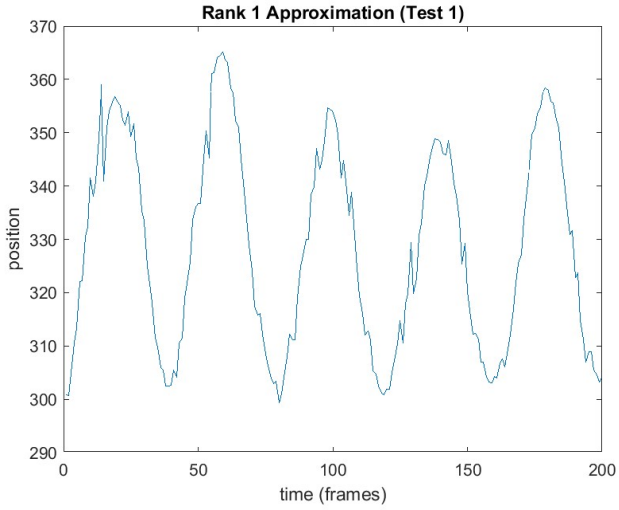
Test 1: Ideal case





From the singular values and the corresponding energy, we can see that most of the information is contained in the rank 1 approximation and so the principal component represented by the first singular vectors captures the main information of the mass's position. However, the other singular values and energy are not insignificant, which suggests that there is still information in these rank approximations, possibly due to noise.



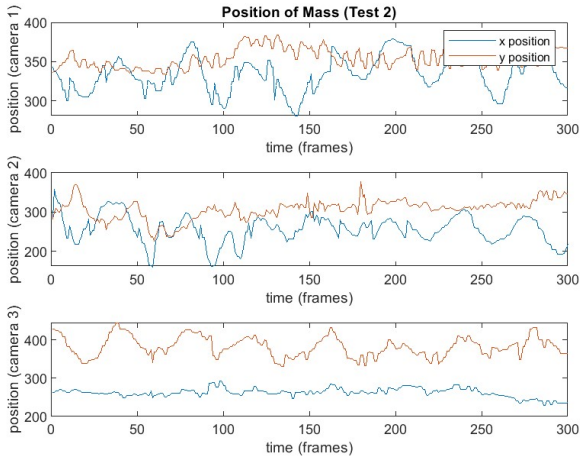


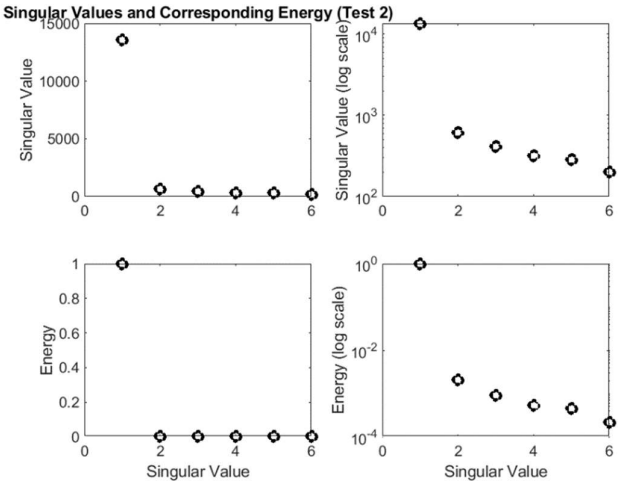
(Note that the time variation plot of the modes was zoomed in to better see the behaviour of the first mode)

We can see that the modes themselves are not as important as how they vary in time, which makes sense as we are looking for the behaviour of the mass depending on time. We can see that the first mode varies sinusoidally which while the others are much more random.

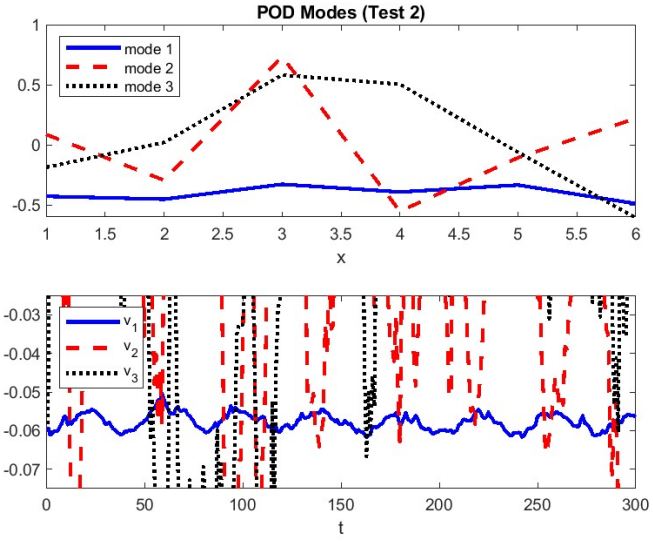
We also see that the rank 1 approximation leads to a sinusoidal oscillation of the position, which is what is expected for the behaviour of a spring mass system as governed by equation 1.

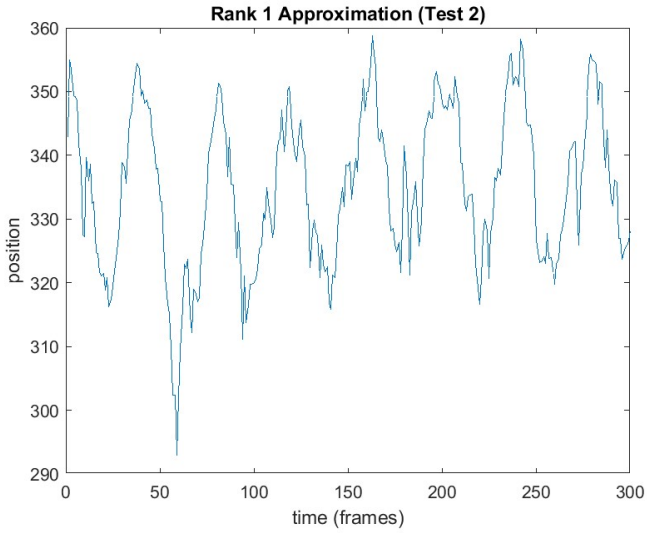
Test 2: Noisy Case





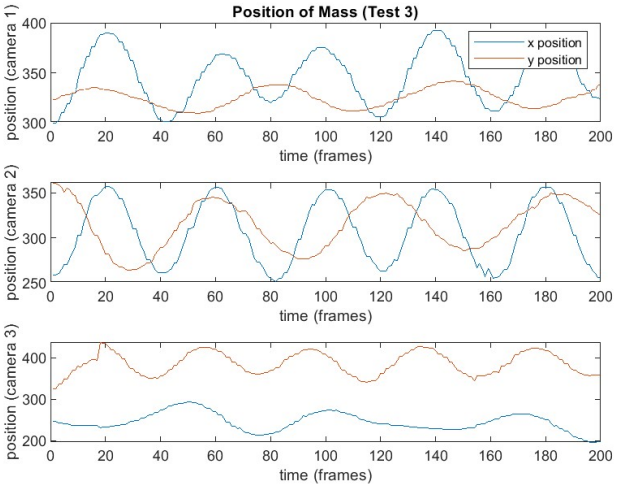
Though the position extracted from the video is much noisier in this case, most of the information is still captured in the rank 1 approximation like in test 1.

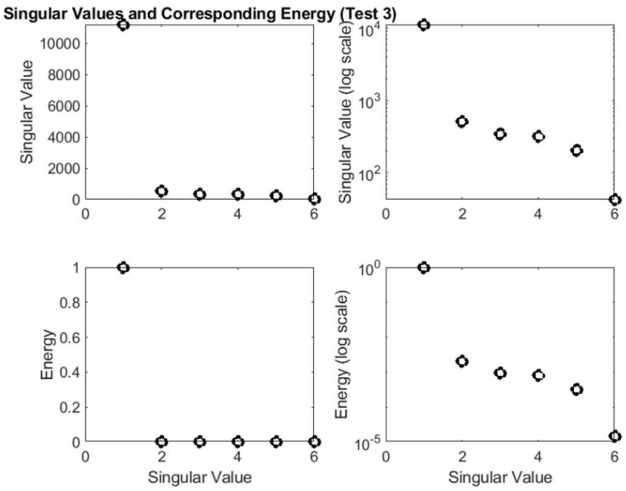




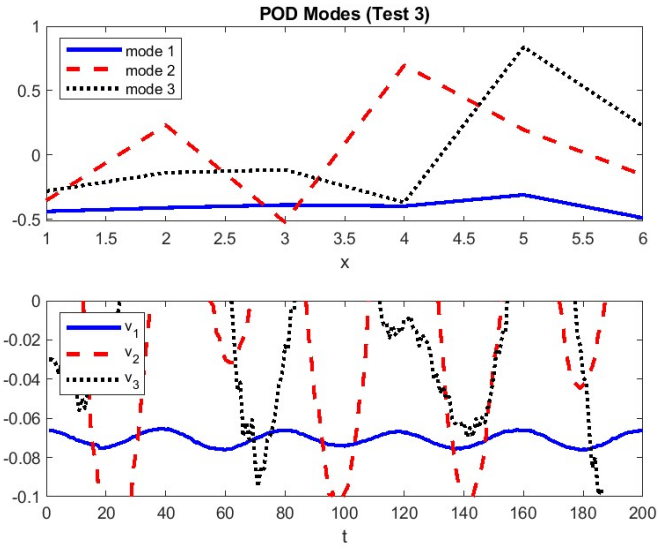
We can still see that the time variation of the first mode as well as the rank 1 approximation is roughly sinusoidal like in test 1, though with more noise.

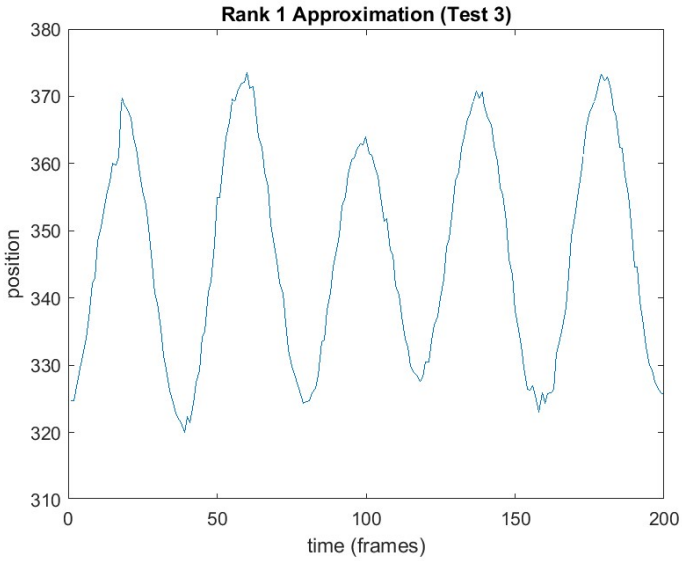
Test 3: Horizontal displacement





Surprisingly, the plot of singular values and energy is very similar to tests 1 and 2. The first singular value and energy is much greater than the rest which are still not insignificant but are just as small compared to the first as in tests 1 and 2. This suggests that most of the position information is contained in the rank 1 approximation, even though now there is added motion along different axes.

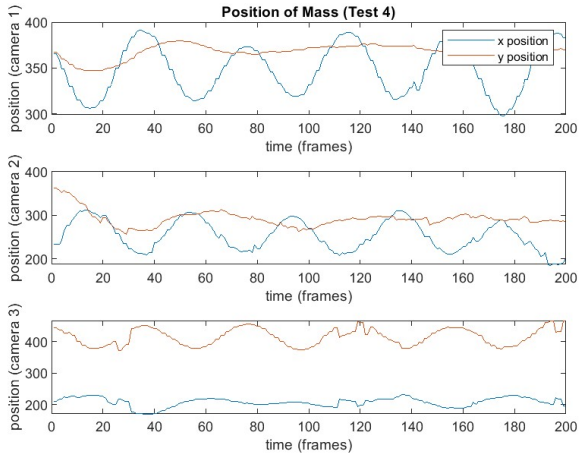


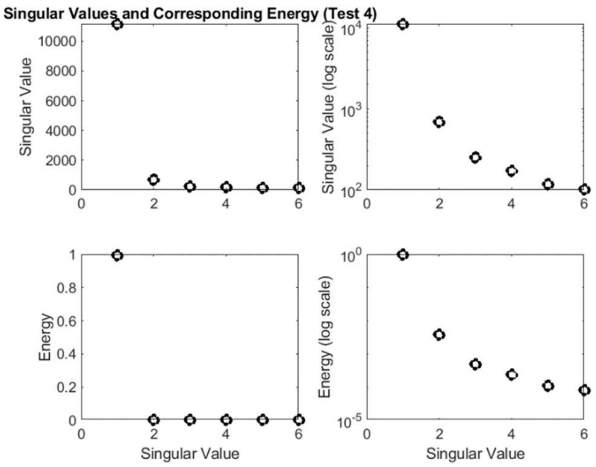


We see again that there is sinusoidal behaviour in both the time variance of the first mode and in the rank 1 projection. There is also some smooth oscillation of the next 2 modes, however these are not nearly as significant since the corresponding energy is much smaller. This oscillation of the next modes may correspond to the secondary horizontal oscillation, but it is hard to tell since the singular values and energies are about the same size as tests 1 and 2 where they were attributed to noise.

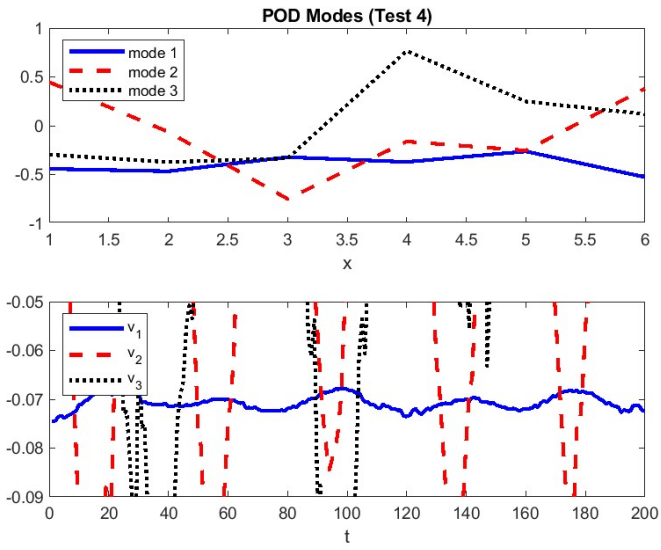
The fact that most of the information is captured in the rank 1 approximation might have multiple reasons. The added degree of oscillation may be proportionally related to the up and down oscillation from tests 1 and 2 such that both oscillations combined can be represented using a single principal component. Or since these secondary oscillations are much smaller, they are not represented by the PCA as much, possibly because they are drowned out by the noise in the background.

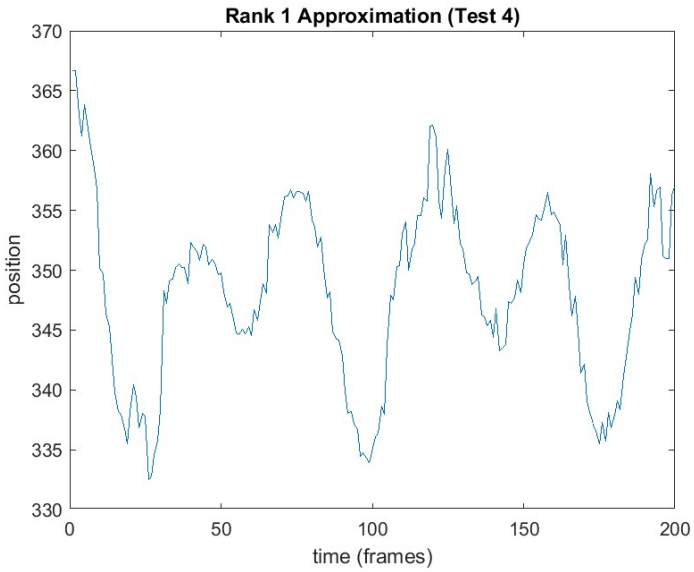
Test 4: Horizontal Displacement and Rotation





Similar to all other tests, we can see that most of the information is contained in the rank 1 approximation.





Like all other test cases, there is some sinusoidal behaviour in the first mode's time dependence. Also like in test 3 we see that the next 2 modes also have some relatively smooth oscillatory behaviour, but it is not as important due to the smaller singular values and energy.

We see that the rank 1 approximation oscillates but there is significantly more noise and isn't cleanly sinusoidal like in tests 1, 2 and 3. This may because at some points in time, the light used to identify the mass is out of view making it harder to track the position.

Summary and Conclusion

We were able to determine the principal components and the behaviour along them for each of the 4 test cases as described above. However the behaviour for tests 3 and 4 where the singular values and energy suggest most of the information is in the rank 1 approximation, despite a multi-dimensional oscillation is strange. This goes to show that though SVD and PCA are very useful for extracting just the useful information, they sometimes can be like a black-box making it difficult to understand unusual cases like this.

Appendix A: MatLab Functions Used

- **Y = zeros(X)** – Returns an n-by-n matrix of zeros.
  - https://au.mathworks.com/help/matlab/ref/zeros.html
- **Y = ones(X)** – Returns an n-by-n matrix of ones.
  - https://au.mathworks.com/help/matlab/ref/ones.html
- **Y = exp(X)** – Returns the exponential e^x for each element in array X.
  - https://au.mathworks.com/help/matlab/ref/exp.html
- **I = rgb2gray(RGB)** – Converts the truecolor image RGB to the grayscale image I.
  - https://au.mathworks.com/help/matlab/ref/rgb2gray.html
- **I2 = im2double(I)** – Converts the image I to double precision.
  - https://au.mathworks.com/help/matlab/ref/im2double.html
- **C = conv2(A,B)** – Returns the two-dimensional convolution of matrices A and B.
  - https://au.mathworks.com/help/matlab/ref/conv2.html
- **[M,I] = max(A,[],'all','linear')** – Returns the maximum value in A as well as the linear index into A that corresponds to the maximum value in A.
  - https://au.mathworks.com/help/matlab/ref/max.html
- **[I1,I2,...,In] = ind2sub(sz,ind)** – Returns the equivalent multidimensional subscripts corresponding to the linear indices ind for a multidimensional array of size sz.
  - https://au.mathworks.com/help/matlab/ref/ind2sub.html
- **[U,S,V] = svd(A,'econ')** – Performs a reduced singular value decomposition of matrix A
  - https://au.mathworks.com/help/matlab/ref/double.svd.html
- **plot(X,Y,LineSpec)** – Creates a 2-D line plot of the data in Y versus the corresponding values in X.
  - https://au.mathworks.com/help/matlab/ref/plot.html
- **semilogy(X,Y,LineSpec)** – Plots x- and y-coordinates using a linear scale on the x-axis and a base-10 logarithmic scale on the y-axis.
  - https://au.mathworks.com/help/matlab/ref/semilogy.html
- **print(filename,formattype)** – Saves the current figure to a file using the specified file format.
  - https://au.mathworks.com/help/matlab/ref/print.html

Appendix B: MatLab Code

```
%% Load Test 1
clear; close all; clc;
load('cam1_1.mat');
load('cam2_1.mat');
load('cam3_1.mat');

vids = cell(1,3);
vids{1} = vidFrames1_1(:,:,:,11:211);
vids{2} = vidFrames2_1(:,:,:,20:220);
vids{3} = vidFrames3_1(:,:,:,11:211);
numFrames = size(vids{1},4)-1;

%% Analyse Test 1
pos = zeros(6,numFrames);
x = 1:640;
y = 1:480;
filter = exp(-0.000006*(y' - 240).^2)*exp(-0.000006*(x -
320).^2);

for i = 1:3
    for j = 1:numFrames
        X = im2double(rgb2gray(vids{i}(:,:,:,j+1)));
        X = conv2(X.*filter,ones(15)*(1/15^2));
        [M,I] = max(X,[],'all','linear');
        [row,col] = ind2sub(size(X),I);
        if M > 0.3
            pos(i*2-1:i*2,j) = [row;col];
        elseif j>1
            pos(i*2-1:i*2,j) = pos(i*2-1:i*2,j-1);
        else
            pos(i*2-1:i*2,j) = [0;0];
        end
    end
end

%% Plot Test 1
figure(1)
subplot(3,1,1)
plot(1:numFrames,pos(1,:));
title('Position of Mass (Test 1)');
hold on;
plot(1:numFrames,pos(2,:));
xlabel('time (frames)');
ylabel('position (camera 1)');
```

```
legend('x position','y position');
subplot(3,1,2)
plot(1:numFrames,pos(3,:));
hold on;
plot(1:numFrames,pos(4,:));
xlabel('time (frames)');
ylabel('position (camera 2)');
subplot(3,1,3)
plot(1:numFrames,pos(5,:));
hold on;
plot(1:numFrames,pos(6,:));
xlabel('time (frames)');
ylabel('position (camera 3)');
print('tracking_1','-dpng');

[U,S,V] = svd(pos,'econ');

sig = diag(S);
figure(2);
subplot(2,2,1);
plot(sig,'ko','Linewidth',2);
title('Singular Values and Corresponding Energy (Test
1)');
ylabel('Singular Value');
subplot(2,2,2);
semilogy(sig,'ko','Linewidth',2)
ylabel('Singular Value (log scale)');
subplot(2,2,3);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2);
xlabel('Singular Value');
ylabel('Energy');
subplot(2,2,4);
semilogy(sig.^2/sum(sig.^2),'ko','Linewidth',2);
ylabel('Energy (log scale)');
xlabel('Singular Value');
print('svals_1','-dpng');

f = 1:6;
t = 1:numFrames;
figure(3);
subplot(2,1,1);
plot(f,U(:,1),'b',f,U(:,2),'--
r',f,U(:,3),':k','Linewidth',2);
title('POD Modes (Test 1)');
xlabel('x');
legend('mode 1','mode 2','mode
3','Location','northwest');
```

```
subplot(2,1,2);
plot(t,V(:,1),'b',t,V(:,2),'--
r',t,V(:,3),':k','Linewidth',2);
xlabel('t');
ylim([-0.1 0]);
legend('v_1','v_2','v_3','Location','northwest');
print('modes_1','-dpng');

f_rank1 = U(:,1)*S(1,1)*V(:,1)';
figure(4);
plot(1:numFrames,f_rank1(1,:));
title('Rank 1 Approximation (Test 1)');
xlabel('time (frames)');
ylabel('position');
print('projection_1','-dpng');

%% Load Test 2
clear; close all; clc;
load('cam1_2.mat');
load('cam2_2.mat');
load('cam3_2.mat');

vids = cell(1,3);
vids{1} = vidFrames1_2(:,:,:,14:314);
vids{2} = vidFrames2_2(:,:,:,2:302);
vids{3} = vidFrames3_2(:,:,:,18:318);
numFrames = size(vids{1},4)-1;

%% Analyse Test 2
pos = zeros(6,numFrames);
x = 1:640;
y = 1:480;
filters = cell(1,3);
filters{1} = exp(-0.00001*(y' - 300).^2)*exp(-0.00008*(x
- 350).^2);
filters{2} = exp(-0.00001*(y' - 250).^2)*exp(-0.00001*(x
- 320).^2);
filters{3} = exp(-0.00008*(y' - 250).^2)*exp(-0.00001*(x
- 300).^2);

for i = 1:3
    for j = 1:numFrames
        X = im2double(rgb2gray(vids{i}(:,:,:,j+1)));
        X = conv2(X.*filters{i},ones(15)*(1/15^2));
        [M,I] = max(X,[],'all','linear');
        [row,col] = ind2sub(size(X),I);
        if M > 0.3
```

```
            pos(i*2-1:i*2,j) = [row;col];
        elseif j>1
            pos(i*2-1:i*2,j) = pos(i*2-1:i*2,j-1);
        else
            pos(i*2-1:i*2,j) = [0;0];
        end
    end
end

%% Plot Test 2
figure(1)
subplot(3,1,1)
plot(1:numFrames,pos(1,:));
title('Position of Mass (Test 2)');
hold on;
plot(1:numFrames,pos(2,:));
xlabel('time (frames)');
ylabel('position (camera 1)');
legend('x position','y position');
subplot(3,1,2)
plot(1:numFrames,pos(3,:));
hold on;
plot(1:numFrames,pos(4,:));
xlabel('time (frames)');
ylabel('position (camera 2)');
subplot(3,1,3)
plot(1:numFrames,pos(5,:));
hold on;
plot(1:numFrames,pos(6,:));
xlabel('time (frames)');
ylabel('position (camera 3)');
print('tracking_2','-dpng');

[U,S,V] = svd(pos,'econ');

sig = diag(S);
figure(2);
subplot(2,2,1);
plot(sig,'ko','Linewidth',2);
title('Singular Values and Corresponding Energy (Test
2)');
ylabel('Singular Value');
subplot(2,2,2);
semilogy(sig,'ko','Linewidth',2)
ylabel('Singular Value (log scale)');
subplot(2,2,3);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2);
```

```
xlabel('Singular Value');
ylabel('Energy');
subplot(2,2,4);
semilogy(sig.^2/sum(sig.^2),'ko','Linewidth',2);
ylabel('Energy (log scale)');
xlabel('Singular Value');
print('svals_2','-dpng');

f = 1:6;
t = 1:numFrames;
figure(3);
subplot(2,1,1);
plot(f,U(:,1),'b',f,U(:,2),'--
r',f,U(:,3),':k','Linewidth',2);
title('POD Modes (Test 2)');
xlabel('x');
legend('mode 1','mode 2','mode
3','Location','northwest');
subplot(2,1,2);
plot(t,V(:,1),'b',t,V(:,2),'--
r',t,V(:,3),':k','Linewidth',2);
xlabel('t');
ylim([-0.075 -0.025]);
legend('v_1','v_2','v_3','Location','northwest');
print('modes_2','-dpng');

f_rank1 = U(:,1)*S(1,1)*V(:,1)';
figure(4);
plot(1:numFrames,f_rank1(1,:));
title('Rank 1 Approximation (Test 2)');
xlabel('time (frames)');
ylabel('position');
print('projection_2','-dpng');

%% Load Test 3
clear; close all; clc;
load('cam1_3.mat');
load('cam2_3.mat');
load('cam3_3.mat');

vids = cell(1,3);
vids{1} = vidFrames1_3(:,:,:,38:238);
vids{2} = vidFrames2_3(:,:,:,25:225);
vids{3} = vidFrames3_3(:,:,:,35:235);
numFrames = size(vids{1},4)-1;

%% Analyse Test 3
```

```
pos = zeros(6,numFrames);
x = 1:640;
y = 1:480;
filter = exp(-0.000005*(y' - 240).^2)*exp(-0.000005*(x -
320).^2);

for i = 1:3
    for j = 1:numFrames
        X = im2double(rgb2gray(vids{i}(:,:,:,j+1)));
        X = conv2(X.*filter,ones(15)*(1/15^2));
        [M,I] = max(X,[],'all','linear');
        [row,col] = ind2sub(size(X),I);
        if M > 0.3
            pos(i*2-1:i*2,j) = [row;col];
        elseif j>1
            pos(i*2-1:i*2,j) = pos(i*2-1:i*2,j-1);
        else
            pos(i*2-1:i*2,j) = [0;0];
        end
    end
end

%% Plot Test 3
figure(1)
subplot(3,1,1)
plot(1:numFrames,pos(1,:));
title('Position of Mass (Test 3)');
hold on;
plot(1:numFrames,pos(2,:));
xlabel('time (frames)');
ylabel('position (camera 1)');
legend('x position','y position');
subplot(3,1,2)
plot(1:numFrames,pos(3,:));
hold on;
plot(1:numFrames,pos(4,:));
xlabel('time (frames)');
ylabel('position (camera 2)');
subplot(3,1,3)
plot(1:numFrames,pos(5,:));
hold on;
plot(1:numFrames,pos(6,:));
xlabel('time (frames)');
ylabel('position (camera 3)');
print('tracking_3','-dpng');

[U,S,V] = svd(pos,'econ');
```

```
sig = diag(S);
figure(2);
subplot(2,2,1);
plot(sig,'ko','Linewidth',2);
title('Singular Values and Corresponding Energy (Test
3)');
ylabel('Singular Value');
subplot(2,2,2);
semilogy(sig,'ko','Linewidth',2)
ylabel('Singular Value (log scale)');
subplot(2,2,3);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2);
xlabel('Singular Value');
ylabel('Energy');
subplot(2,2,4);
semilogy(sig.^2/sum(sig.^2),'ko','Linewidth',2);
ylabel('Energy (log scale)');
xlabel('Singular Value');
print('svals_3','-dpng');

f = 1:6;
t = 1:numFrames;
figure(3);
subplot(2,1,1);
plot(f,U(:,1),'b',f,U(:,2),'--
r',f,U(:,3),':k','Linewidth',2);
title('POD Modes (Test 3)');
xlabel('x');
legend('mode 1','mode 2','mode
3','Location','northwest');
subplot(2,1,2);
plot(t,V(:,1),'b',t,V(:,2),'--
r',t,V(:,3),':k','Linewidth',2);
xlabel('t');
ylim([-0.1 0]);
legend('v_1','v_2','v_3','Location','northwest');
print('modes_3','-dpng');

f_rank1 = U(:,1)*S(1,1)*V(:,1)';
figure(4);
plot(1:numFrames,f_rank1(1,:));
title('Rank 1 Approximation (Test 3)');
xlabel('time (frames)');
ylabel('position');
print('projection_3','-dpng');
```

```
%% Load Test 4
clear; close all; clc;
load('cam1_4.mat');
load('cam2_4.mat');
load('cam3_4.mat');

vids = cell(1,3);
vids{1} = vidFrames1_4(:,:,:,38:238);
vids{2} = vidFrames2_4(:,:,:,25:225);
vids{3} = vidFrames3_4(:,:,:,35:235);
numFrames = size(vids{1},4)-1;

%% Analyse Test 4
pos = zeros(6,numFrames);
x = 1:640;
y = 1:480;
filter = exp(-0.000005*(y' - 240).^2)*exp(-0.000005*(x -
320).^2);

for i = 1:3
    for j = 1:numFrames
        X = im2double(rgb2gray(vids{i}(:,:,:,j+1)));
        X = conv2(X.*filter,ones(15)*(1/15^2));
        [M,I] = max(X,[],'all','linear');
        [row,col] = ind2sub(size(X),I);
        if M > 0.3
            pos(i*2-1:i*2,j) = [row;col];
        elseif j>1
            pos(i*2-1:i*2,j) = pos(i*2-1:i*2,j-1);
        else
            pos(i*2-1:i*2,j) = [0;0];
        end
    end
end

%% Plot Test 4
figure(1)
subplot(3,1,1)
plot(1:numFrames,pos(1,:));
title('Position of Mass (Test 4)');
hold on;
plot(1:numFrames,pos(2,:));
xlabel('time (frames)');
ylabel('position (camera 1)');
legend('x position','y position');
subplot(3,1,2)
plot(1:numFrames,pos(3,:));
```

```
hold on;
plot(1:numFrames,pos(4,:));
xlabel('time (frames)');
ylabel('position (camera 2)');
subplot(3,1,3)
plot(1:numFrames,pos(5,:));
hold on;
plot(1:numFrames,pos(6,:));
xlabel('time (frames)');
ylabel('position (camera 3)');
print('tracking_4','-dpng');

[U,S,V] = svd(pos,'econ');

sig = diag(S);
figure(2);
subplot(2,2,1);
plot(sig,'ko','Linewidth',2);
title('Singular Values and Corresponding Energy (Test
4)');
ylabel('Singular Value');
subplot(2,2,2);
semilogy(sig,'ko','Linewidth',2)
ylabel('Singular Value (log scale)');
subplot(2,2,3);
plot(sig.^2/sum(sig.^2),'ko','Linewidth',2);
xlabel('Singular Value');
ylabel('Energy');
subplot(2,2,4);
semilogy(sig.^2/sum(sig.^2),'ko','Linewidth',2);
ylabel('Energy (log scale)');
xlabel('Singular Value');
print('svals_4','-dpng');

f = 1:6;
t = 1:numFrames;
figure(3);
subplot(2,1,1);
plot(f,U(:,1),'b',f,U(:,2),'--
r',f,U(:,3),':k','Linewidth',2);
title('POD Modes (Test 4)');
xlabel('x');
legend('mode 1','mode 2','mode
3','Location','northwest');
subplot(2,1,2);
plot(t,V(:,1),'b',t,V(:,2),'--
r',t,V(:,3),':k','Linewidth',2);
```

```
xlabel('t');
ylim([-0.09 -0.05]);
legend('v_1','v_2','v_3','Location','northwest');
print('modes_4','-dpng');

f_rank1 = U(:,1)*S(1,1)*V(:,1)';
figure(4);
plot(1:numFrames,f_rank1(1,:));
title('Rank 1 Approximation (Test 4)');
xlabel('time (frames)');
ylabel('position');
print('projection_4','-dpng');
```