Rajat Dixit

33415118

ELEC 221 Project

Project ELEC 221

Task 1 : Frames Extraction

0. **What are the total number of frames in the video?**
   3081

   What is the frame rate?
   29.9700

1. **What is the size (number of rows, columns,..) of the data array representing each frame?**
   When the imwrite function was used between the 45th and the 56th second, 340 frames were found. In our case imwrite is called to save to img data type.
   The characteristics of img are given below.
   Size : 720x1280x3 uint8
   Its 8 bit per image color therefore it gives a total RGB bit depth of 24bits
   Each image/frame is 1280 pixels x 720 pixels
   True RGB image has 24bit image depth

Task 2 : Conversion to Grayscale

2. **What is the number of bits per pixel in a grayscale image?**
   Using the whos info function we get a bit depth of 8 bits. We expected this as instead of having RGB (3 channels, 24bits) we only have 1 monochrome channel (1 channel, 8 bits).

   Explain briefly, and possibly aided by equations, how an RGB image is converted to grayscale. (If you use a MATLAB built-in function, you still need to state which one you use and briefly explain how it works.)
   The MATLAB rgb2gray function was used.
   The rgb2gray takes a video reader object which in our case is named ‘imggrayscale’. It goes through every frame and converts it into an 8 bit image.
   In the whole conversion process it takes each frame and eliminates the hue and saturation information while retaining the luminance. This results in a frame/image that is of same brightness but instead of RGB its now grayscale

   Code :

```
w = VideoReader ('nasa_video.mp4');
w.CurrentTime = 45.4;
jj = 1;
while w.CurrentTime <= 56.7
imggrayscale = readFrame(w);
filename1 = [sprintf('%03d',jj) '.png'];
fullname1 = fullfile('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files','Images_Grayscale',filename1);
```

```
I = rgb2gray(imggrayscale);
imwrite(I,fullname1)
jj = jj+1;
end
```

3. **Use imshow function to display the image**

To display the image the imshow function was used. There are multiple ways one can call the function. The syntax we used here is
imshow('filename')
Code :

```
%% X will output a 2 digit number here
studentno = 18;
X = mod (studentno,30) + 1;
%running the mod function we get our X = 19, therefore we pull up the April%19th frame which
is frame 55 in the Images_Grayscale folder
Imshow ('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files\Images_Grayscale\055.png');
```

Output:

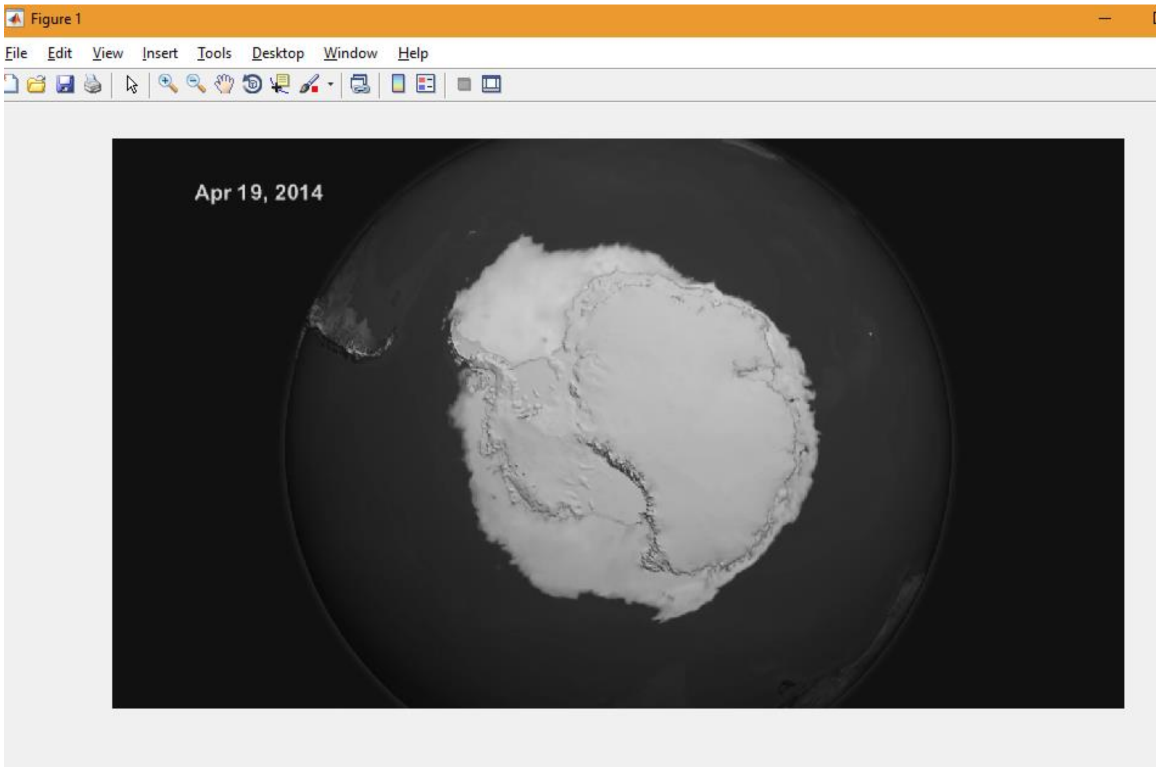
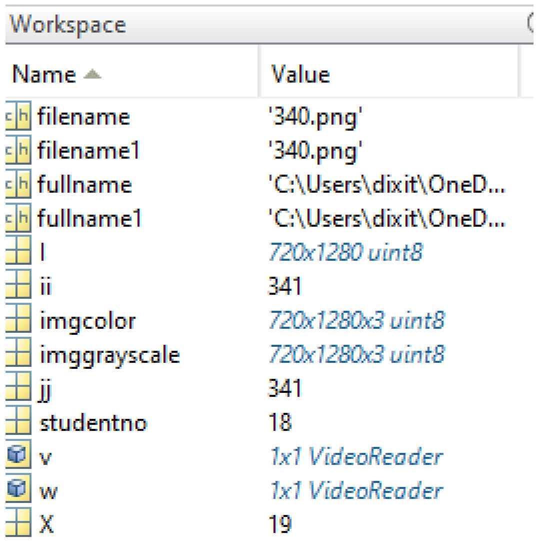


Figure 1 : Showing output of Grayscale image with X = 19, therefore Apr 19, 2014



Workspace

Name	Value
filename	'340.png'
filename1	'340.png'
fullname	'C:\Users\dixit\OneD...
fullname1	'C:\Users\dixit\OneD...
I	720x1280 uint8
ii	341
imgcolor	720x1280x3 uint8
imggrayscale	720x1280x3 uint8
jj	341
studentno	18
v	1x1 VideoReader
w	1x1 VideoReader
X	19

Figure 2 : Showing the value of X = 19 (last row) from our modulus operation.

4. **Explain why conversion to grayscale is a logical step before conversion to black and white.**
Conversion to grayscale will help reduce the bit depth of the image which in turn will be helpful and faster for the function to further convert it to black and white. We can also retain the luminance of the image by converting into grayscale first. Conversion to grayscale also allows us to be more flexible with the image as now we can mask and edit the image as we like.

Task 3 : Conversion to Black and White

5. **Explain briefly how conversion of black and white can be performed?**
The imbinarize function was used in this case. The function creates a binary image by replacing all values above a determined threshold with 1s and setting all other values to 0s. The value of threshold is specified as a scalar luminance value or a matrix of luminance values.

Code :

```
%% Convert to binary black and white

kk = 1;
grayfiles = dir('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files\Images_Grayscale\*.png');
for i=1:340
   readbw = strcat('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files\Images_Grayscale\', grayfiles(i).name);
   convert_bw = imread(readbw);
   bw = imbinarize(convert_bw, 0.5);
   filename2 = [sprintf('%03d',kk) '.png'];
   fullname2 = fullfile('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files','Images_BW',filename2);
   imwrite(bw,fullname2);
   kk = kk+1;
```

```
end
```

Use imshow function to display the black and white image

Displaying the April 19, 2014 image. (It is a coincidence that the last 2 digits of my student number also give an answer = 19)

Code:

```
%% Display black and white image from grayscale
imshow('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files\Images_BW\055.png');
```

Output:

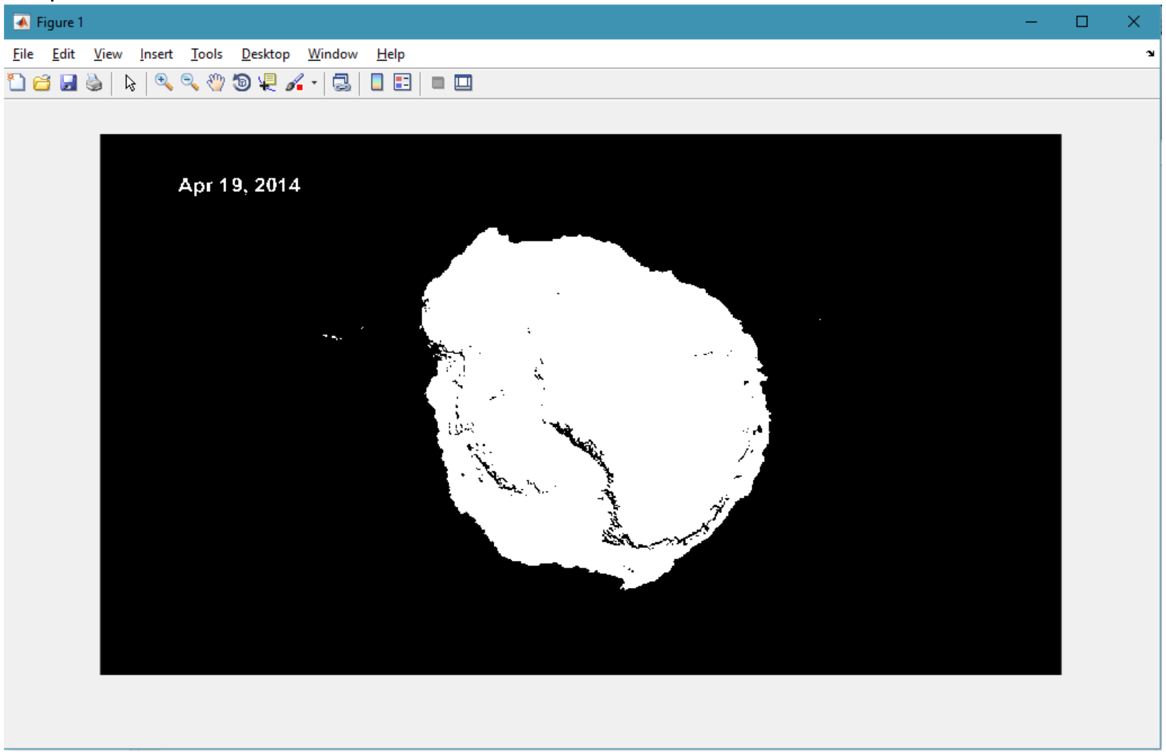


Figure 3 : Showing binary B/W image. The function used here is imbinearize.

Task 4 : Masking out Date Stamp

6. **Explain the method to remove date stamp from the top left of each frame**

To remove datestamp we used two functions called createMask and regionfill. While createMask creates a rectangular mask of user's choice, the regionfill option fills up the area highlighted by createMask.

To begin with we drew a rectangle on a sample image. These rectangle's coordinates were stored in a temporary variable called 'h' in our case.
These co-ordinates were then used to mask and fill all images in the for loop (mentioned below). The for loop goes through each file and applies the mask and regionfill. Once the steps are applied the for loop saves all these images into a new folder called 'Images_WoDate'

Code :

```
 %% Display black and white image from grayscale without datestamp
clearvars
imshow('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files\Images_Grayscale\055.png');
imageread = imread('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files\Images_Grayscale\055.png');
h = imrect;
BW = createMask(h);
J = regionfill(imageread,BW);
imshow(J);

kk = 1;
grayfiles = dir('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files\Images_Grayscale\*.png');
for i=1:340
   readbw = strcat('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files\Images_Grayscale\', grayfiles(i).name);
   convert_bw = imread(readbw);
   without_date = regionfill(convert_bw,BW);
   bw = imbinarize(without_date, 0.5);
   filename2 = [sprintf('%03d',kk) '.png'];
   fullname2 = fullfile('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab
Files','Images_WoDate',filename2);
   imwrite(bw,fullname2);
   kk = kk+1;
end
```

Output:

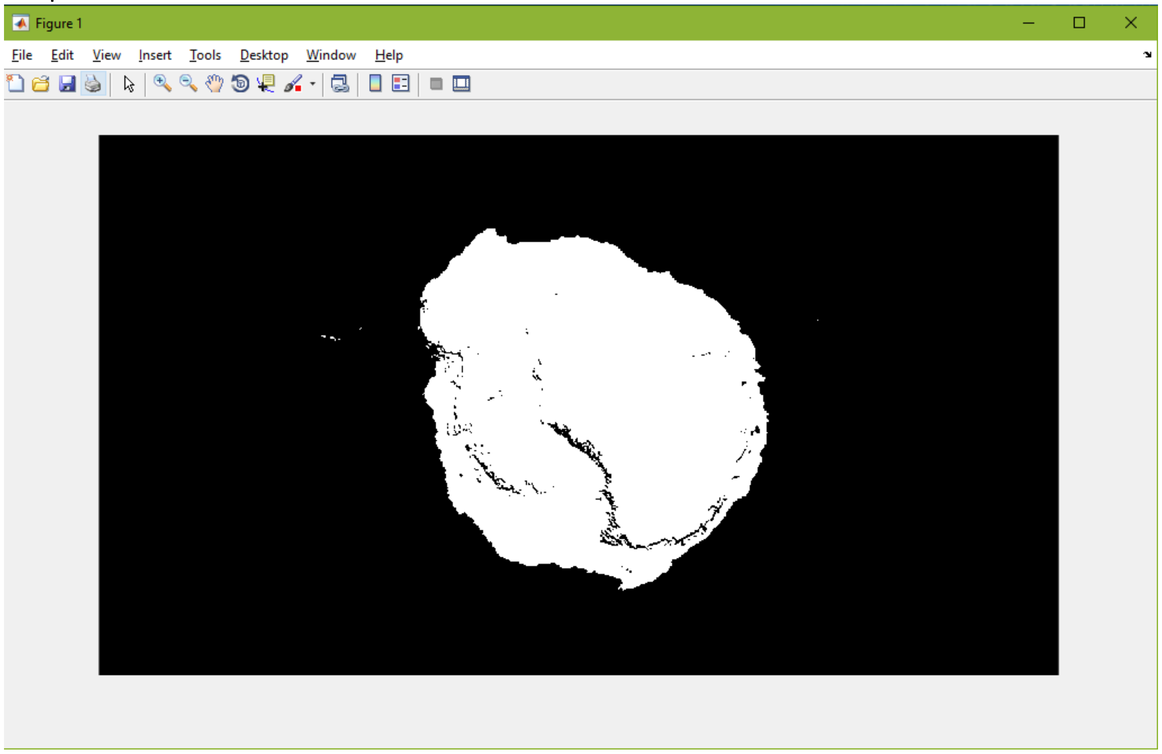


Figure 4 : Showing B/W binary image without date stamp. Please note that this image is from Task 3. The mask was applied to all 340 images under test.

Task 5 : Spatial Filtering

7. **What size N filter did you use**
   N = 5

   How many times did you apply filter for each frame
   5 times

   Code :
```
%% Filtering
%imshow ('C:\Users\dixit\OneDrive\Winter 2016 Courses\221
ELEC\Project\Matlab Files\Images_WoDate\055.png')
image_under_test = imread ('C:\Users\dixit\OneDrive\Winter 2016
Courses\221 ELEC\Project\Matlab Files\Images_WoDate\055.png');
f = ones(5,5) / 25;
filter_image = imfilter(image_under_test,5*f);
imshow(image_under_test), title('Original Image');
figure, imshow(filter_image), title('Filtered Image')
```

   Output:

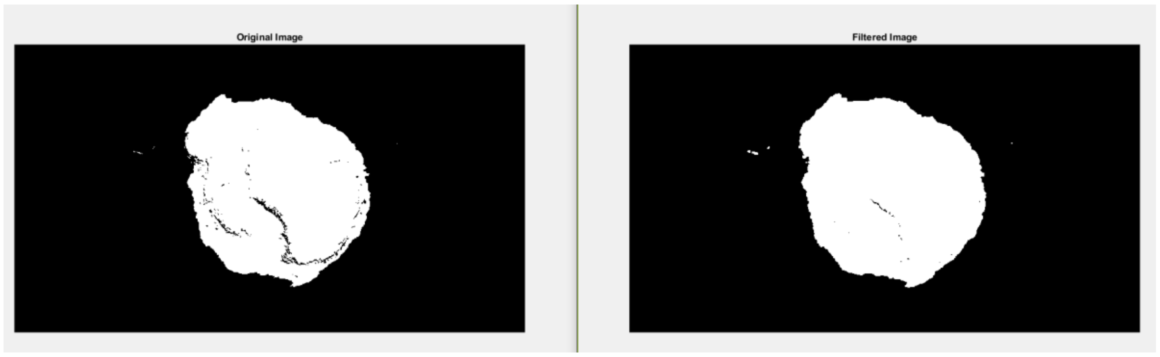


Figure 5 : Showing both original and filtered image from Task 4. The image above has gone through the averaging filter.

8. **Use matlab to find 'fft2' of your filter. Comment on the result.**
   Code:

```
%% FFT
f = ones(5,5) / 25;
Y = fft2(f);
```

Output:

```
Name        Size                Bytes  Class

Y           5x5                   200  double
```

We see the result of our Fourier transform is 1. Since this is a low pass filter we see that our filter retains low frequencies and attenuates high frequency components. Its seen in our image too as the filter tries to smoothen out this high frequency sharpness.

Task 6 : Estimate the change in ice content.

Plot the change in ice

Code :

```
%% Area
clearvars
BW = imread('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files\Images_WoDate_Filtered\001.png');
%imshow (BW)
area1 = bwarea(BW);
whos area1

kk = 1;
Wo_Date_Files_Filtered = dir('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files\Images_WoDate_Filtered\*.png');
for i=1:340
  readbw = strcat('C:\Users\dixit\OneDrive\Winter 2016 Courses\221 ELEC\Project\Matlab Files\Images_WoDate_Filtered\', Wo_Date_Files_Filtered(i).name);
  area_of_filtered = imread(readbw);
```

```
    area(i) = bwarea(area_of_filtered);
    inpercent(i) = (((area(i) - area1)) / (area1)) * 0.01;
    kk = kk+1;
end
figure % opens up new figure window
plot (1:340,area);
title('Change in Ice Content')
xlabel('Number of Data Points')
ylabel('Percent Change of Ice')
```

Output

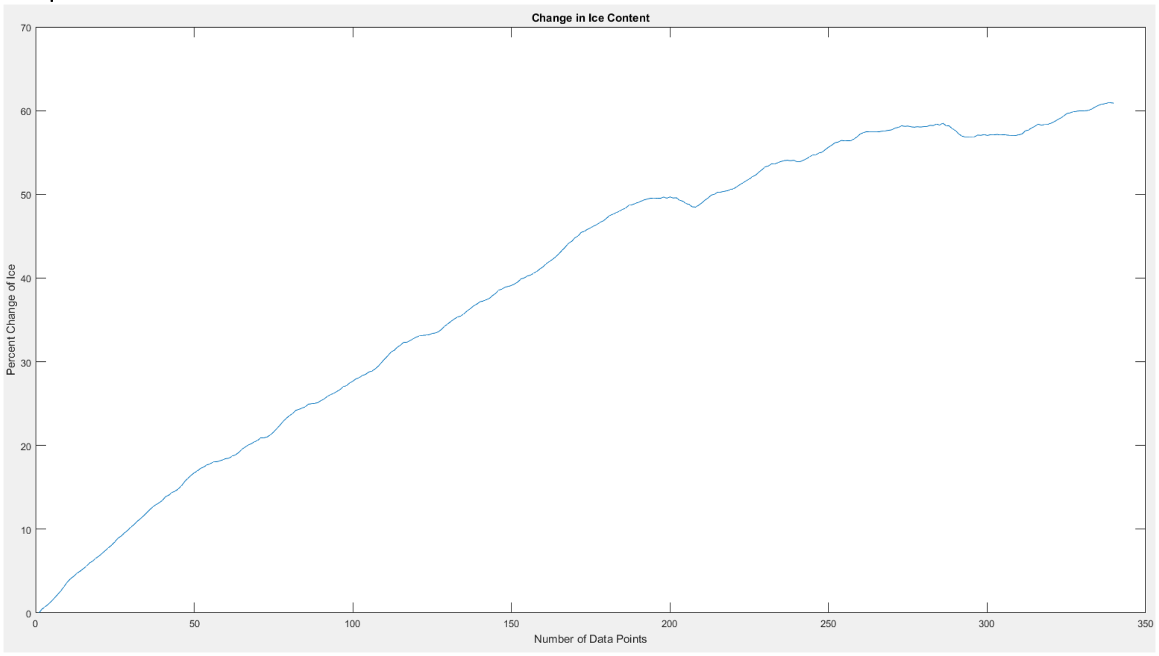


Figure 6 : Data Points (number of days) vs Percent change in ice.

9. **Comments**
We see that as we go from March to September our % change in ice increases. It is to be noted that the Y axis is in % of ice. We see a rise upto 60% from March to September. Since the number of frames extracted was 340 our data points go till 340 instead of 180. Essentially for each second the videoframe function algorithm extracted 2 frames.

Task 7 : Polynomial Regression

Use an appropriate polynomial that models change of ice area over time. Use matlab's 'polyfit' function

Code

```
x = linspace(1,340,10);
y = linspace(0,60,10);
polyf = polyfit (x,y,3);

x1 = linspace(1,340);
```

```
y1 = polyval(polyf,x1);
figure
plot(x,y,'o')
hold on
plot(x1,y1)
hold off
```

Output

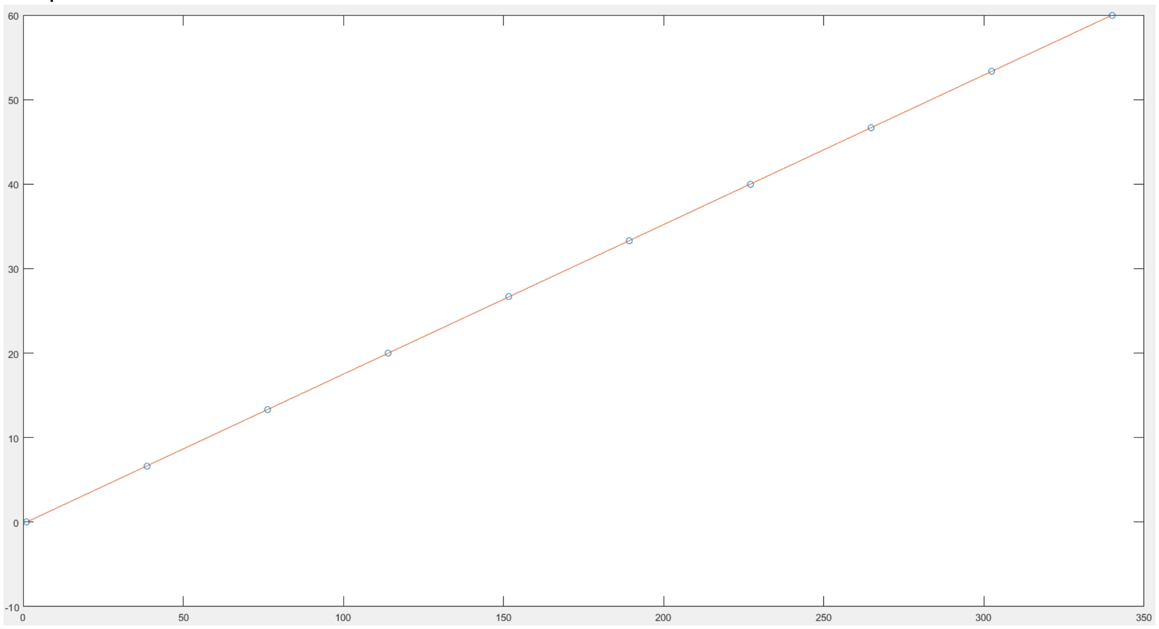
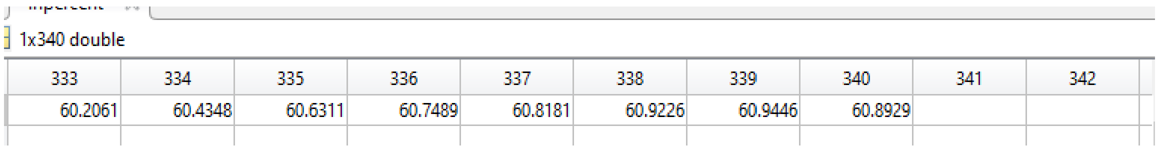


Figure 7 : Comparing Polyfit to the data points obtained in Task 6.

Comments
We see here that we fit the polynomial to our original equation. The red line is our original equation. The dotted line is the polynomial fit to our equation. It is a degree 3 polynomial. We can see close to linear response in the above graph. The calculated polynomial works well with the given system.

10. **Write down the polynomial function you use**
The polynomial function used here is
$5.619^{-21}x^3 \; - 3.18^{-18}x^2 \; + 0.17x - 0.176$

11. **Use polynomial regression mode to predict ice coverage on Sept 30 2014.**
Since our graph is fairly linear we can assume about 60% more ice than normal where normal is the measurement in March. If we look at our values towards the end of measurements.


1x340 double

333	334	335	336	337	338	339	340	341	342
60.2061	60.4348	60.6311	60.7489	60.8181	60.9226	60.9446	60.8929		

We can see that we are floating and fluctuating down to 59-58%. Therefore we can extrapolate that around measurement 351 (i.e Sept 30th 2014) we would see a decrease in ice area by about 1% from 60%. We can expect a value around 59.5%.

Task 8 : Data Collection and Mining

Plot daily ice content for 2014

Code:

```
plot (1:365,website_data);
title('Change in Ice Content')
xlabel('Number of Data Points (days)')
ylabel('Change of Ice (10e6 sqkm)')
```

Here website_data is an array of 1x365 values. Each value represents ice coverage in sqkm.

Output

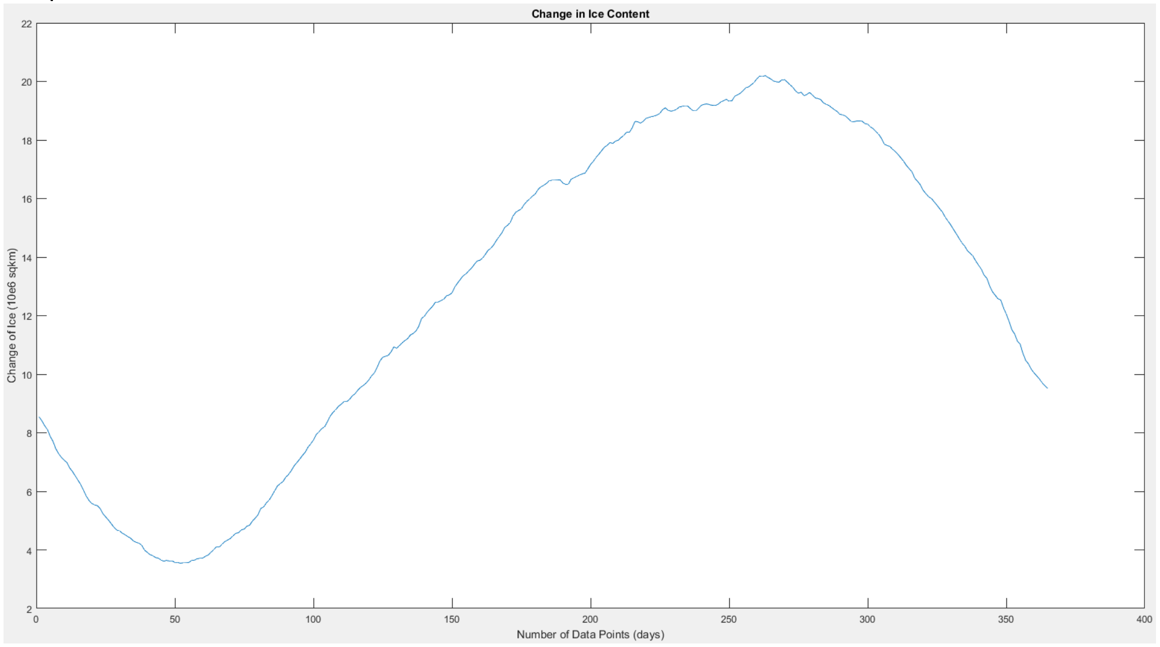


Figure 8 : Figure showing days vs change of ice in sqkm. Our values are only valid till 365 therefore we cutoff at 365 days.

12. **State the minimum and maximum ice extent throughout the year and comment on ice patterns with respect to seasons**

Maximum Ice : 263rd day of the year (September 20th) : 20.2 million sqkm
Minimum Ice : 52nd day of the year (February 21st) : 3.5 million sqkm

I would like to explain Antarctica's weather using a small graph below.
(Ref.
http://www.coolantarctica.com/Antarctica%20fact%20file/antarctica%20environment/vostok_south_pole_mcmurdo.php)

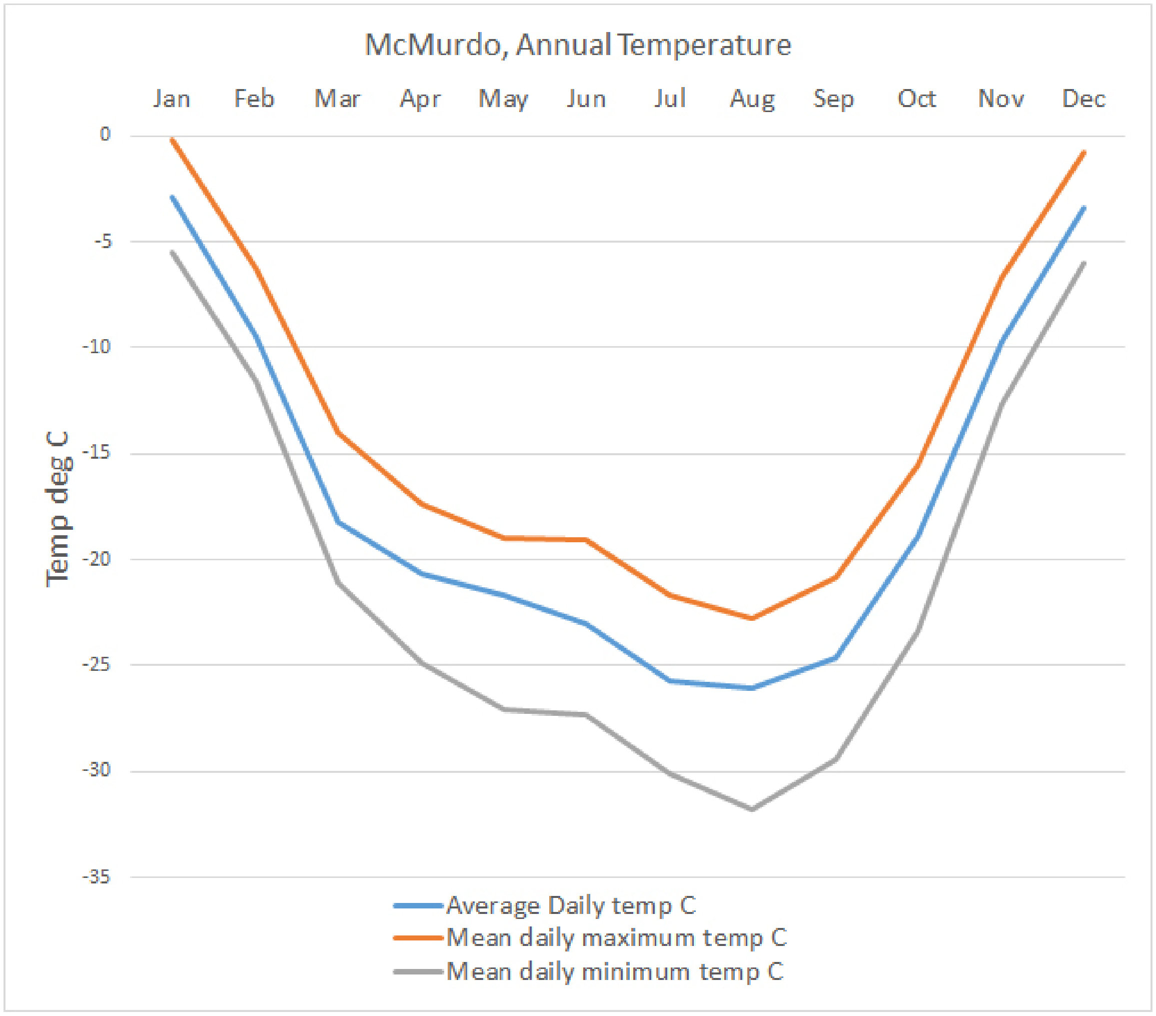


Figure 9 : Explaining Antarctica's weather.

We see that around late August/September we have the coldest temperature in McMurdo (American research base), Antarctica. Our reading and analysis compares well and good with the website and above mentioned graphical measurements. We can see coldest (therefore most ice) temperatures around August/September

Plot percentage change in ice measurements from March 21st to Sep 19th and compare with task 6

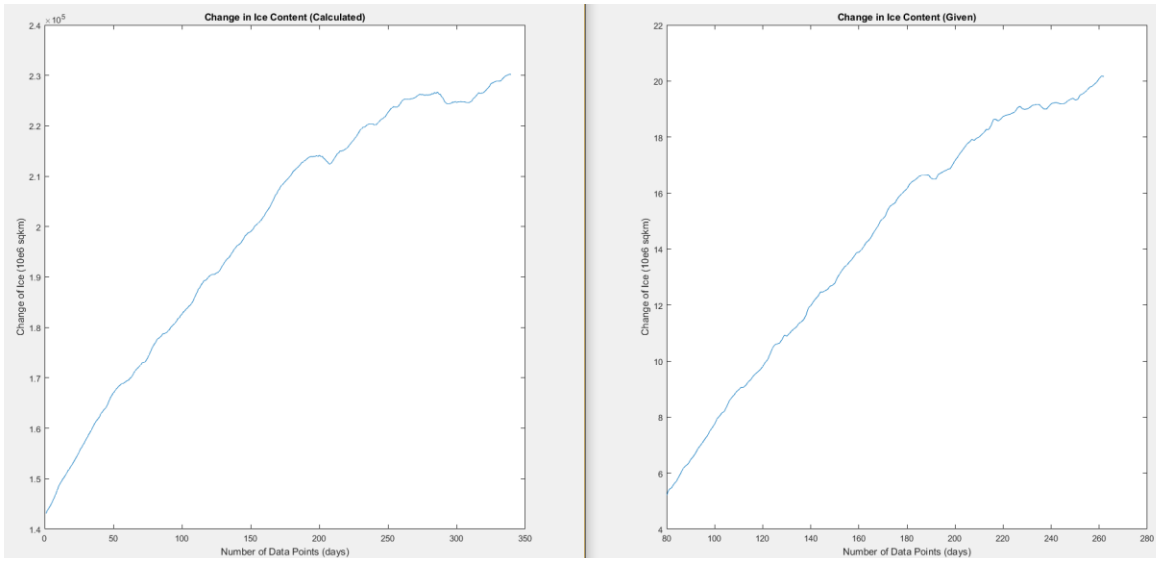


Figure 10 : Showing change in ice content calculated(left) vs given(right).

13. **Comments**
The results are compatible and they mostly match the given results from the website. There seems to be a difference in X axis as the number of frames taken from videoReader were more than 180 (about 340). This happened because videoReader sampled a higher rate. It is to be noted that our data is still correct as 340 frames = 180 days in both cases. Towards our right we see data from day 80 to day 262 (March 21 to Sept 19). Towards our left we see data that we calculated from all our simulations (all 340 frames from March 21 to Sept 19)

We can see that the data is similar to some extent. As we approach Sept 20 we see a peak at the very end of our graph as Sept 20 would mean the most ice in the region.

We see some discrepancy at 0 and the maximum value. This could be because of the implementation of the low-pass filter. Choosing a different value could result in less loss of resolution therefore giving us better white area result calculation. More trial and error or perhaps a better filter design is required in this case.

14. **What other techniques can be utilized to extract more accurate information to estimate ice content?**

Use of better resolution images (1080p) can help us in better implementation of the filter. We would definitely have less edge smoothing therefore resulting in better pixel-to-pixel white area calculation.
Sampling the frames at a higher rate would also help in better ice estimation. Sampling at 60fps or 120fps would result in more information per second thereby helping us in more discrete data points. On the other hand we would possibly need a bigger workstation to handle the increased information and processing power.