**A software application to assess iceberg towing ability**

Student Id: 201277908

MSc in GIS (taught program)

2018-2019

Outline of documents provided for Assessment 2

1. A word file detailing the documents provided for assessment (this file)
   1. Introduction
   2. Intention of the software and the thought process behind its development
   3. Software development process and UML diagram
   4. Running the software and final output
   5. Issues during development and how they were resolved
   6. Further scope of this software development
2. Codes for running

2.1 IcebergTowingApp.py

1. readme.txt
2. License.txt
3. White1.radar.txt
4. White1.lidar.txt

\*\* content 2 to 6 are within the zipped file (GEOG5990M\_[201277908]\_Assessment2)

Link to my Github profile: <https://github.com/nawfee>

Link to my project for this assessment in Github: Repository name: Assessment1\_201277908, <https://github.com/nawfee/Assessment1_201277908>

* 1. Introduction

This project is a python-based code writing for a data analysis-based software application. Python is a high level object-oriented language which provides ability to code for creating models, applications or data analysis. Using python 3.7 version a software application for analysing remote sensing data has been utilized in this project. The work has been done in Windows 10.

For code writing and representation anaconda package (64-bit Graphical Installer) was used here. The IDE (Integrated Development Environment) used was ‘Spyder’ (The scientific python development environment). It allowed the codes to be written in the editor panel and the output to be released in the IPython console. The GUI (Graphical User Interface) of the software developed was also done using Spyder.

* 1. Intention of the software

The software provided in this project is a data analysis-based application which aims to assess remotely sensed data. The main goal of the software is to find out definite pixels from remotely sensed data, to analyse them and come up with a decision.

This software has been specifically developing to determine presence of and towing ability of icebergs in sea. The necessity of such software is in the shipping industry, where iceberg-towing tugs are attached to ship if there are icebergs in the sea. Specifically, the application software will help to find out if there are icebergs present in the sea. If so, their location can be found and then calculate out their volume and mass to give a decision on whether the iceberg can be dragged or not by towing tug. This assessment will help shipping company to decide if the ships should take iceberg-towing tug or not.

The program tries to solve the questions posed on the project idea named, “White Star Line” in the link below:

<https://www.geog.leeds.ac.uk/courses/computing/study/core-python/assessment2/ice.html>

* 1. Software development process

The code for the software has been developed in python version 3.7 in Spyder IDE. The remotely sensed data for assessment of icebergs presence and characteristics were obtained from the University of Leeds, website.

The software development process followed the ‘Incremental development model.’ Here after idea generation, codes were written in separate blocks, managed and tested and if it runs well than new set of works were done in similar manner.

The software development process proceeded by answering the questions given in the link above. The first task set was:

1. Pull in the two data files and display them.

The source files for the software included a satellite radar image, named white1.radar, which can be accessed from the source:

* <https://www.geog.leeds.ac.uk/courses/computing/study/core-python/assessment2/white1.radar>

Another file is an airborne lidar image, named white1.lidar, which can be assessed from the source:

* <https://www.geog.leeds.ac.uk/courses/computing/study/core-python/assessment2/white1.lidar>

Both the files contain comma separated values, that represent pixel values of an area of sea. The pixel values range from 0 to 255. The data spread represents 300 m2 of area of sea using 300 by 300 pixels. Both the data files are saved as text files and was extracted in to the python IDE using csv reader code from the csv module.

Here the radar image provides information on the texture of the sea environment. That is, it tells where the icebergs are present and where not. The lidar data represents the same area but contains pixel values which represents height.

The radar and the lidar data were displayed in the program using matplotlib function. So, matplotlib.pyplot module was imported.

1. Assess which areas of the image are ice using radar data

A condition has been given that pixel values over 100 contains iceberg. So, the first work of the software development was to identify these pixels. Giving the conditions that 100-255 pixel values represent iceberg and the rest water, the row and column index of the starting and ending point of the iceberg was determined.

1. Assess the total mass of ice above sea level

The starting and ending row and column index for the iceberg was passed on to lidar data to assess the height of the iceberg. Before that the lidar data list is converted to array, so numpy module was imported.

Each unit of lidar pixel values represented 10 cm of height, that is 10 units represents 1 meter of height.

A class named Iceberg\_towing was created and initialised.

Next a function was defined (totalvol) to calculate the volume of the iceberg in m3. This involved doing for-loop to assess each pixel values of the iceberg in the lidar image and to do calculation on them so that the height data can be converted to meter and the volume can be calculated from them. The equation for volume calculation is:

Volume = height\*area

Here, height= pixel value \*0.1 (to convert to meter)

Area = length x width of the pixel in m2

The total volume of the iceberg was assessed from the individual volume values, by doing a sum of the values.

The total volume function of the software can be used to determine volume of any icebergs given their height and area values are passed on the functions.

The towing ability is depended on the total mass of the iceberg. So, a function was created to calculate the total mass of the iceberg. Firstly, the mass of the iceberg above water was obtained using the formula:

Mass = density x volume

The density of ice is 900 kg/ m3

The volume is the total volume obtained from the above function.

1. Calculate the total mass of the iceberg

It is considered that only 10% of the iceberg is above water level and that the water level is at 0 meter. In such case the total mass of the iceberg can be calculated by multiplying it by 10.

1. Display the total mass, total volume and whether you can pull the berg on the GUI

A function was defined, name: towingability. In it a condition was set, as such that if the total iceberg mass is equal to greater than 36 million kg, than it can’t be dragged out of the way.

For setting up the Graphical User Interface ‘tkinter’ module was imported from the library. The graphical interface is set in a way that it contains three buttons, code for each was written with each function defined, and they were bind to the functions.

**UML Diagram**

|  |
| --- |
| Radar Data |
| + Pixel values: float |
| +sea: list  ~ matplotlib.pyplot |

|  |
| --- |
| Lidar Data |
| + Pixel values: float |
| + lidar\_sea: list  ~ matplotlib.pyplot  ~ numpy |

|  |
| --- |
| Iceberg\_Towing |
| + \_\_init\_\_  + totalvol()  + totalmass()  + towingability() |

* 1. Running the software

For running the software program, the anaconda 3.7 version is needed to be installed in a windows computer. Within the zipped file provided for this project is the two source files:

White1.radar.txt and White1.lidar, txt, these two along with the source code file for this program named, IcebergTowingApp.py must be copied in the same directory.

In the Spyder IDE, the python file must be loaded. Then running the program, first prints out the row, column start and end index for iceberg. The images of radar and lidar data is displayed in the IPython console as well.

It also opens a new window name ‘Iceberg Towing Application’. Here there are three buttons, which needs to be clicked separately to get the results in IPython consoles.

Total iceberg volume = this will read the value of total iceberg volume in m3.

Total iceberg mass=this will print the value of total iceberg mass in kg

Towing ability= This is the most important button, as it gives decisions on whether the iceberg pulling is possible or not.

* 1. Issues during software development

One of the major issue during this software development was inheriting the pixel values which contain the iceberg. The original radar file has been extracted in 2D list format, so to access the presence of ice the index value of row and column at the start and end of the iceberg was required.

This issue was resolved by defining the condition for not only the iceberg but also for rows and columns above, below and on the left and right of the iceberg.

Another issue was faced after inheriting the row, column index at the start and end of iceberg was to extract the pixel values at those indexes. The list had to be converted to an array, by importing numpy module. Then the index values were passed into the lidar data file, to get the pixel values representing height of the iceberg.

* 1. Further scope of software development

The given program has some limitations. By resolving these issues, the software can be developed further.

The first limitation is not being able to determine the parameters for the three functions defined. If these can be defined, then passing in any values of height and area of the iceberg, can help to assess their towing ability.

The second limitation is that this software locates only one iceberg. If there is more than one iceberg in the sea and their towing ability must be determined separately than the code needs to be developed further. The row and column starting and ending index for each iceberg must be assessed and each iceberg must be given an ID, then their total mass and towing ability can be determined.