|  |  |  |  |
| --- | --- | --- | --- |
|  | | logo_black.gif | |
|  | 201199221 |  | 1052 |
|  | GEOG5991M |  |  |
|  | Assignment 2 – Final Project |
|  | Nick Gould |

|  |
| --- |
|  |
| 1. |
| 2. |
| 3. |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

**PROJECT DOCUMENTATION**

**INTRODUCTION**

The purpose of the document is to describe the developed code which satisfies the assumed White Star Line Project requirements. This includes the description of the code, the process of the development, the issues raised during the coding, instructions how to run the code and indication of the general resources used. The minimum requirements for the code are that it should generally read some information from files, process the data, display the results and write the analysed information into the file.

White Star Line Project is suggested to create the process of identification of the icebergs from the Lidar and Radar images. The details can be found in the following link:

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

Briefly, the goal is to analyse the radar and lidar images of the sea and identify if the bergs can be pulled by tug or not. In addition to the requirements to assess tow-ability of the icebergs by colours, the interaction with the result image was supported to be able to view the analysed characteristics of the bergs.

**SOFTWARE DESCRIPTION**

The main part of the developed python script intuitively shows the steps which code implements:

* Read image files;
* Identify the ice areas;
* Create iceberg objects and calculates the characteristics of the objects during initialisation;
* Display the source images and identified icebergs with the analysed result text. In addition, the event handling (hovering over the iceberg) is enabled to interact with the output to get the result text on the canvas;
* Save the analysed result text into the text file.

The code consists of three python files which are in control of specific tasks:

* ‘ice\_v2.py’ is the main module which implement the code steps;
* 'icebergstructure.py' includes the Iceberg class definition which encompasses berg characteristics;
* 'imagesupport.py' contains ImageHandle class definition which mainly used to read, process and visualise image datasets.

The python files can be found in the following link in the GitHub repo created for the [Assignment 2](https://github.com/ohajiyev/Assignment2/) purposes:

<https://github.com/ohajiyev/Assignment2/tree/master/python/src/unpackaged/ice/Version2>

The python files commented to acknowledge the variables and methods.

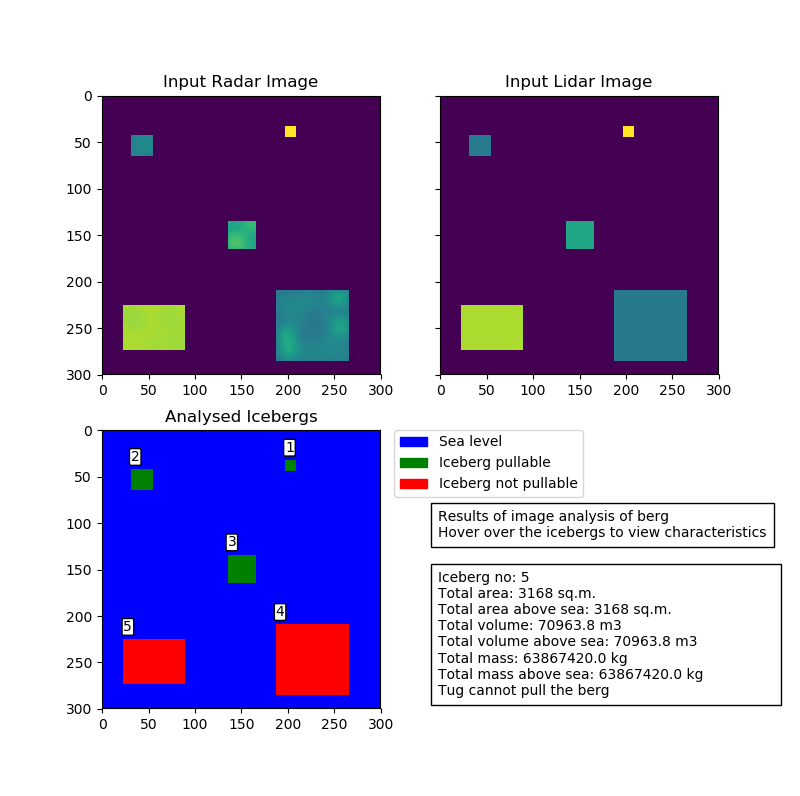
Also, the instructions how to run the code was produced and available in the Appendix 1 of the report and in the following link:

<https://github.com/ohajiyev/Assignment2/blob/master/python/src/unpackaged/ice/Version2/!!!Readme_How_to_run.txt>

Jupyter Notebook file was also created and can be viewed in the below link which also visualise the snapshot of the result (see Figure 1):

<https://github.com/ohajiyev/Assignment2/blob/master/python/src/unpackaged/ice/Version2/ice2_notebook.ipynb>

Figure 1. Snapshot of the result



UML class diagram was created to show the code structure and can be found in the Appendix 2 of the report and in the following link in the GitHub repo:

<https://raw.githubusercontent.com/ohajiyev/Assignment2/master/python/src/unpackaged/ice/Version2/uml_diagram.png>

**THE DEVELOPMENT PROCESS**

The development of the project started by analysing and understanding the requirements. Initial design stage was to implement minimum conditions and attempted to develop the code which handle images with single iceberg. In a such way, it was decided to create two versions of the code. First, to develop the code which read and analyse the images with single iceberg. And after this, the second version of the script which handles the images with multiple bergs was created.

To begin with, the file reading of the images with single berg was implemented and tested with the available images successfully. Initially, the list data type was utilised to read dataset from the file. But further attempt to analyse the image dataset reveals the necessity of using numpy library which simplify manipulating of data array. Next step was visualisation of the input images using matplotlib. As it was planned at the design stage, Iceberg class was created inside the main script file despite of only one iceberg should be on the images. This gave the possibility to expand the script into the multiple icebergs case later. Iceberg class contained attributes and function which calculated the parameters of the berg and results of analysis. Further two methods were created to visualise the result and write it to the file. Thus, the first release version was developed which was minimum viable product and able to read and analyse the input files successfully.

The second version of the code is implemented to deal with the images with multiple bergs. The algorithm for identification of the bergs was designed and implemented. Algorithm goes once through all values of the derived image and assign individual number to each berg. The assumption was introduced that the bergs data should be continuous and without any gaps. Any gaps in input data may wrongly identify the bergs. These are artificial files and in the reality no image can come ideally without any gaps or distortions. For real images the given algorithm hardly can be applied. For the real case, object classification method of machine learning should be utilised. Also, the algorithm cannot differentiate between the bergs with overlaps. The algorithm was tested in the separate code with the couple simple samples created for this purpose. The enhancements were introduced to visualise the results and the interaction with the output was implemented. At a later phase, ImageHandle class was created to collect the various functions/methods to process image datasets.

In general, the code was structured by eliminating repetitive code lines, defining methods/functions and adding the comments to the variables and functions to simplify the understanding and readability of the script.

Anaconda and Spyder was used as IDE for writing the code. Also, GitHub Desktop was practiced for the management of the project and to track the history of the changes in the script.

**ISSUES DURING THE DEVELOPMENT PROCESS**

During the picturing of the analysed images, the issue raised with the implementation of colour scheme. Stake Overflow forum was searched to find the solution. Another question was elevated to enable interaction with the matplotlib canvas. Again, Stake Overflow resources was used to resolve the issue.

**GENERAL RESOURCES**

The main sources used in the development of the code was the University [provided materials](https://www.geog.leeds.ac.uk/courses/computing/study/core-python-odl/). Additionally, [Stack Overflow](https://stackoverflow.com/) website was used to seek solutions for some part of the code which are indicated in the comments of the code. It is important to note that some part of the code may be copied and modified from the different sources which are explicitly shown in the scripts’ comments.

**PROPOSED IMPROVEMENTS**

The code can be improved by adding \_\_init\_\_ method and attributes to reduce the repetition of the referencing of the arguments in the methods of the ImageHandle class. The visualisation part can be enhanced by using real GUI not relying only on matplotlib capability. Machine learning techniques can be implemented to improve identification of the bergs.

**APPENDIXES**

**Appendix 1. HOW TO RUN**

Title:

Assignment 2. Module GEOG5991M

White Star Line Project

Version:

2.0

Github Repo Link:

https://github.com/ohajiyev/Assignment2

Zip file link:

https://github.com/ohajiyev/Assignment2/tree/master/python/src/unpackaged/

ice/Version2/assignment2.zip

===============================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

===============================================================================

INSTRUCTIONS TO RUN:

Download 'assignment2.zip' file and extract.

The extracted folder should contain the following files and folders, as

minimum:

'ice\_v2.py' - main code to run from command prompt

'icebergstructure.py' - Iceberg class definiton

'imagesupport.py' - ImageHandle class definiton

'ice2\_notebook.ipynb' - Jupyter Notebook file

'input' - folder which contains input images 'white2.lidar' and

'white2.radar'

'input/white2.lidar' - lidar image (300x300) of an area of sea with

the multiple bergs. Values (0-255) contain

height data of the objects

'input/white2.radar' - radar image (300x300) of an area of sea with

the multiple bergs. Values (0-255) contain

information to idenitify the bergs (>=100)

'output' - folder which is output folder for the result text file

('result.txt)

Software requirements:

Anaconda3 (64bit):

Python 3.7

Spyder 3.3.2

Jupiter Notebook

Anaconda prompt

The code can be run in Anaconda command line, Spyder and Jupyter notebook.

Anaconda cmd:

1. 'python ice\_v2.py'

Spyder:

1. Open 'ice\_v2.py

2. Ensure that IPython console is activated

3. Run '#%matplotlib qt5' command in IPython console to interact

with interface.

4. Press 'F5' button or 'Run' from the menu to run the code

Jupyter Notebook:

1. Open 'ice2\_notebook.ipynb' in browser

2. Run first line '%matplotlib notebook' to enable interaction

with the output

3. Press 'Run all' from Run menu

===============================================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

===============================================================================

Limitations:

The code only was tested and developed with/for available two sets of the

inputs provided by the University. It is assumed that the bergs data should

be continuous and without any gaps. Any gaps in input data may wrongly

identify the bergs. These are artificial files and in the reality

no image can come ideally without any gaps or distortions. For real images

the algorithms which is used to bergs identification hardly can be applied.

For the real case object classification method of machine learning should

be applied. Also the algorithm cannot differentiate the bergs with

overlaps

Purpose:

To implement the requirements of the Assignment 2 of the Module GEOG5991M.

The version was built to analyse MULTIPLE iceberg images.

White Star Line was selected as the project to satisfy the assignment's

description. The link to the project's problem definition:

https://www.geog.leeds.ac.uk/courses/computing/

study/core-python-odl/assessment2/ice.html

License:

Copyright (c) 2019 Orkhan Hajiyev

Lisence under MIT License

License link:

https://github.com/ohajiyev/Assignment2/blob/master/LICENSE.md

Python version:

3.7 (Python 3.7.1 64-bit | Qt 5.9.6 | PyQt5 5.9.2 | Windows 10)

Coding Tool:

Spyder Version 3.3.2

**Appendix 2. UML DIAGRAM**

