Question 1

The script for question 1 is file `lvisSingleline.py`, the arguments to use is:

```
python3 lvisSingleline.py filepath resolution outputpath
```

like:

```
python3 lvisSingleline.py ./2009/ILVIS1B_AQ2009_1020_R1408_053614.h5 30
./data/output.tif
```

Here's how I solve the DEM:

1. Read `.h5` file provided as augument using `lvisGround` class provided before.
2. For each `(lon, lat)` point, calculate ground elevation using command `lvisGround.setElevations(), lvisGround.estimateGround()`
3. For each cell in the DEM to be solved, use `IDW` algorithm to solve the elevation value.
4. Save DEM data into file using `tiffHandle.writeTiff`.

The processes of IDW algorithm are:

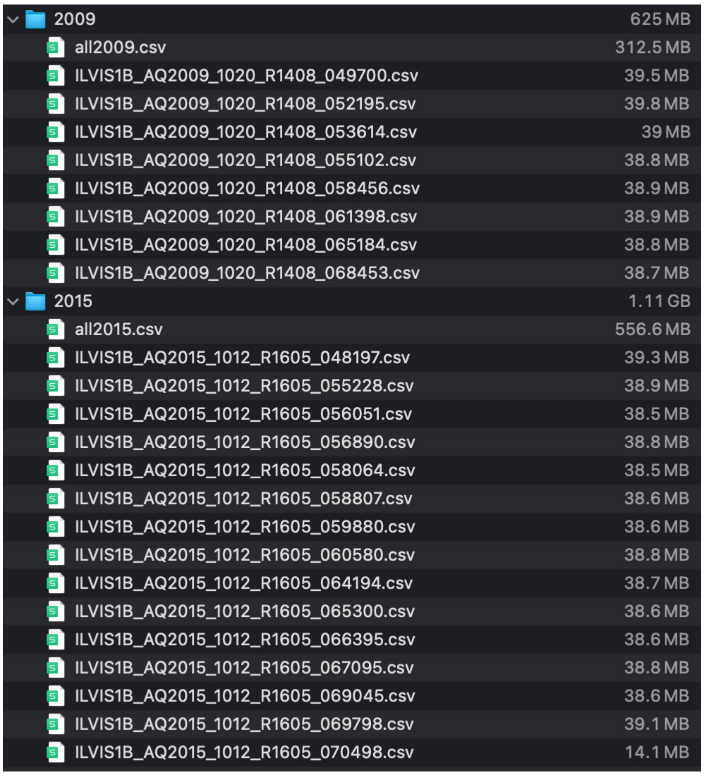
$$u(\mathbf{x}) = \begin{cases} \dfrac{\sum_{i=1}^{N} w_i(\mathbf{x}) u_i}{\sum_{i=1}^{N} w_i(\mathbf{x})} & \text{if } d(\mathbf{x}, \mathbf{x}_i) \neq 0 \text{ for all i} \\ u_i & \text{if } d(\mathbf{x}, \mathbf{x}_i) = 0 \text{ for some i,} \end{cases}$$

However, if all points are considered, the time consumption is too much, so I designed an algorithm to find nearest *15* points and calculate the estimate value. I also make sure that if there isn't too many points in a radical scope, the estimate value will not be considered.

Question 2

Data preprocess

In this question, it is hard to make sure the RAM usage to be kept under 2 GB. In my practice, I preprocess the data and save `lon, lat, z` data into disk. there's about 500MB for 2009 and 1GB for 2015 saved as csv files.



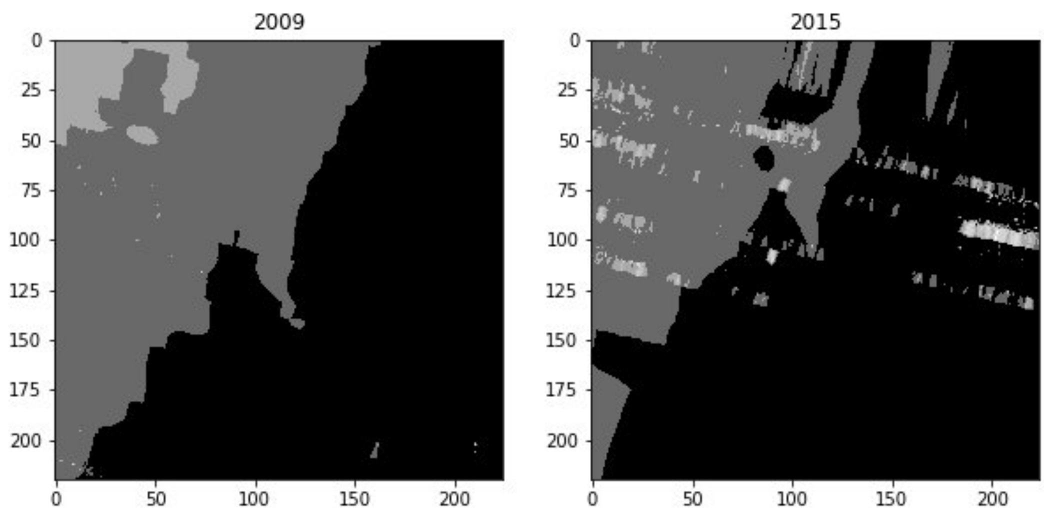
The code to process data is saved as file `lvisHandler.py`, it containes all data that I need to use.

Gap filling algorithm

To solve the gap filling algorithm, I'm still using `IDW` to interpolate gap cell, but increasing the tolerence of the scope, but the `IDW` algorithm have a complexity of $O(n^2)$, thus it seems impossible to do all the work. I eventually selected an rectangle:

$$\text{longtude: } 99^\circ\text{W} - 98^\circ\text{W}$$
$$\text{latitude: } 75^\circ\text{N} - 75.5^\circ\text{N}$$

and the final result is shown below:



It takes about 30mins from reading all `.h5` files, to selecting rectangle, to interpolation and render.

please check file `lvisDEM.py` to reproduce the work above, some files like the selected `.csv` file are provided.

Question 3

I tried some techniques about solving the difference of two region, but the result doesn't look promising. Here are the algorithm that calculate the difference:

- read two dem tiff file.
- calculate difference by using the following formula.

$$\text{diff} = \frac{\text{DEM}_{2015} - \text{DEM}_{2009}}{\text{DEM}_{2009}}$$

- quantify the difference.

The difference that calculated from data is so small that it seems impossible to produce the real data.

Discussion section

I've used several packages in `python`, such as `h5py, numpy, pandas, gdal, pyproj, matplotlib`. By using the techniques and tools in this assignment, we can build a whole process from reading raw `.h5` data to visualizing the DEM model.

The process above produce a simple report for glaciers in IceBridge, about its difference between 2009 and 2015. We can quantify the change by using codes above.

However, the performance of `python` really influence the process. When doing `IDW` interpolation, python can only use one CPU core, it takes a long time to wait for the result and debug.

Given more time and experience, I'll develop a `GUI` interface to make sure the application can be easily used. I may use some library from `gdal` or `arcgis pro` to accelarate the rasterization. Also I may consider more about `RAM` comsumption.

About the `gap-filling`, I think it doesn't perform well when the gap is large. Because of the noise, some result seems wrong. A better algorithm like `kriging interpolation` can be used to improve.