**Description of Program**

The zip file consists of neural network training file and testing file for the neural network.

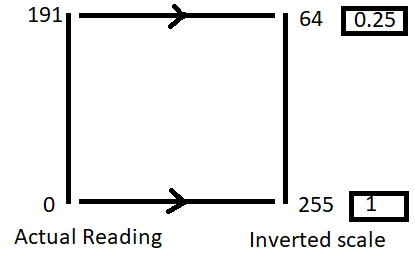
The map\_network.py and map\_network\_resnet.py are used to train the neural network using data in input\_b.csv and output\_b.csv . After training, hdf5 file will be created. Model will be stored in \_classifer.hdf5 file while weight will be stored in \_weight.hdf5 file.

The scan\_tile\*.py files will be used to test neural network using the 3D point cloud data stored in .pose and .3d data. These are the data taken in Leibnitz University.

The scan\_tile\*.py files are loading the pre-trained model stored in hdf5 files. It will then go thru the scan file one at the time. After extracting scan data in .pose and .3d file, adjust orientation using rotation matrix, conducting GPS offset at each scan data.

Preprocessing of data will be made on the x-y grid point which has multiple z values. (Taking the ground value).

Since light or waves can still be reflected from a scan point to the sensor through the space under objects such as tree leaves, these spaces are usually maneuverable. To prevent wrong prediction of space as object at high ground, we are inverting the height values according to figure below before entering neural network. This prevent MaxPooling2D to be more sensitive with the points with higher value.



The scan point will be analyzed to determine the slice number. The scan point will be fitted into 1024x1024 slices. In scan\_tile\*.py, map1 is 2D map consists of all scan points. Map1 will be used to interpolate the grid points in between scan points and sensor position.

Map2 is sliced map obtained from map1. Nominalized map2 will be input into neural network after salt noise reduction using maximum filter. Map2 will be backup as map5 and saved as mapim\*.png. If mapim\*.png has been generated in previous scan, the value stored in mapim\*.png file will be loaded back and concatenate with the new incoming points.

After prediction through neural network, the output will re-shaped back to original 1024x1024 slice size and scaled back with 256 as maximum value. The output (im in scan\_tile\*.py) will be compared against the input (in map5), the obstacle value in map5 will be retained. The output value will be stored in res\*.png (gray value).

Using 245 as thresholding value, pixel with value less than 245 will be marked with blue color. Pixel with value more than 245 will be marked as red color. The value is chosen in order to fit the expected value map. The colored map will be saved as im\*.png (note that value<64 is considered to be unknown value, this is because we impose a clip at actual height of 191. After inversion, it will be reflected as 255-191=64)

After all the scan file pass through the process, the png files are merged. Res\*.png are merged as merged\_gray\_images.png. im\*.png are merged as merged\_images.png .

For edge\_noise reduction, padding and cropping are done using numpy.pad and numpy array function.

**Installation and execution**

1. Install anaconda3 if it is not installed yet
2. Import environment 3Dmapping using project\_environment.yml

Command:

conda env create -f project\_environment.yml

1. Activate the environment

Command:

conda activate 3Dmapping

1. If wants to run training, run map\_network.py (non RESNET) or map\_network\_resnet.py (RESNET)
2. If wants to run testing on scan point files, run scan\_tile\*.py

scan\_tile.py –without RESNET, without edge noise reduction

scan\_tile\_bound.py - without RESNET, with edge noise reduction

scan\_tile\_resnet.py –with RESNET, without edge noise reduction

scan\_tile\_resnet\_bound.py - with RESNET, with edge noise reduction

scan\_tile\_Kmeans.py - testing program for Kmeans clustering , K=7

**Python Environment File**

project\_environment.yml – Consists of anaconda3 environment and all the packages included

**Python Script List**

1. map\_network.py – training of neural network (non-RESNET)
2. map\_network\_resnet.py - training of neural network (RESNET)
3. scan\_tile.py – testing program for pre-trained neural network (without RESNET, without edge noise reduction)
4. scan\_tile\_bound.py testing program for pre-trained neural network (without RESNET, with edge noise reduction)
5. scan\_tile\_resnet.py – testing program for pre-trained neural network (with RESNET, without edge noise reduction)
6. scan\_tile\_resnet\_bound.py - testing program for pre-trained neural network (with RESNET, with edge noise reduction)
7. scan\_tile\_Kmeans.py - testing program for Kmeans clustering , K=7

**Data File List**

1. scan0XX.pose – Consists of GPS and IMU feedback (Euler angles) of vehicle at each scan.
2. scan0XX.3d – Consists of scan point at each scan.
3. input\_b.csv – training data of neural network. (too large in size, go to this [link](https://drive.google.com/file/d/12FK79gaTQEVDxtx6LMv53039HnaOsNAi/view?usp=sharing))
4. output\_b.csv – training data of neural network. (too large in size, go to this [link](https://drive.google.com/file/d/14L1qxBQCsMJgZiPlPVUS7BAofWe2exCK/view?usp=sharing))

**PNG file (Remove all PNG files when running afresh)**

1. merged\_images.png – colored output (red is ground, blue is object)
2. merged\_gray\_images.png – colored output (gray values)

**hdf5 files**

1. map\_network\_classifier.hdf5
2. map\_network\_resnet\_classifier.hdf5
3. map\_network\_weight.hdf5
4. map\_network\_resnet\_weight.hdf5