Delft University of Technology

Bachelor End Project
AESB3400

Validation of ICESat-2 Data in the Netherlands

Stijn Lodder (4574621)

Supervisors:
Hans van der Marel
Roderik Lindenbergh

July 7, 2019

Abstract

On September 15 2018 the new ICESat mission ICESat-2 was launched. This satellite with the new ATLAS photon counting altimeter will measure height all over the world. In the Netherlands there is the AHN, which is an airborne measured height map of the entire Netherlands. The ICESat-2 data can be checked against the AHN to get the quality of ICESat-2 and from the planned ground tracks of ICESat-2 the coverage can be determined. Used is the available ICESat-2 data which spans from 15-10-2018 to 9-1-2019. Bad data is filtered and to the histogram of the difference between ICESat-2 and AHN a distribution is fitted from which the bias and precision are determined to quantify the quality. Overall the quality of the ICESat-2 in the Netherlands is good enough to complete the scientific goal of ICESat-2 with a bias of -4.2 cm and precision of 43.5 cm. ICESat-2 does seem to have a tendency to be too low, which creates a larger histogram tail on the negative side. The cause of this is not discovered among the different groupings in which the data is split. However there are some groups of data, like a specific land cover, pass, beam strength, cloud confidence and photon count, which are better or worse in quality.

Contents

1 Introduction **4**
1.1 Motivation . . . 4
1.2 Objective . . . 4

2 ICESat-2 and Validation Data **5**
2.1 Mission Objectives . . . 5
2.2 Orbit . . . 5
2.3 ATLAS . . . 6
2.4 Mabel . . . 6
2.4.1 Laser Setup . . . 7
2.4.2 Lidar Sensor . . . 9
2.5 Data Products . . . 11
2.6 ATL08 and quality flags . . . 12
2.7 AHN reference height . . . 13

3 Methodology **14**
3.1 Planned path download and extraction . . . 14
3.2 Overview processing . . . 14
3.3 Time Specific Orbit processing . . . 16
3.4 Bulk downloading specific ICESat-2 data . . . 18
3.5 ICESat-2 & AHN2 Data extraction . . . 21
3.5.1 Position & index of ICESat-2 data retrieval . . . 21
3.5.2 AHN2 retrieval . . . 21
3.5.3 ICESat-2 Heights retrieval . . . 21
3.5.4 ICESat-2 flag & other variable retrieval . . . 22
3.6 Validation Metrics . . . 23

4 Planned ICESat-2 over the Netherlands **24**
4.1 Repeats . . . 24
4.2 Temporal frequency and Time specific orbit . . . 27

5 ICESat-2 validation results in the Netherlands **30**
5.1 Datapoints . . . 30
5.2 Quality of all data points . . . 31
5.3 Quality per Land Cover . . . 37
5.4 Quality per Pass . . . 40
5.5 Quality per Strong/Weak beam . . . 42
5.6 Quality per Cloud confidence . . . 44
5.7 Quality per Photon Count . . . 46

5.8 Validation 48

6 Conclusion **49**

7 Recommendations **50**

Appendices **53**

A Scripts **53**

A.1 timedRGTextractor 53
A.2 wgetATLAScommand 55
A.3 wget Command 56
A.4 ATL08import 59
A.5 Datastructer 59
A.6 Precision 62
A.7 Processor 63

1 Introduction

1.1 Motivation

On the 15th of September 2018 the ICESat-2 satellite was launched with the aim to measure vegetation height and more importantly the shrinking ice sheets and their effect on the rising sea level and continue the research from the first ICESat (named ICESat-1 in this report). On the third of October 2018 the first measurments the height of an ice sheet in Antarctica were made. Go forward to of May 2019 and the first data is made publicly available. The quality of this brand new data needs to be checked. For the check the Netherlands is used, because there is good height reference data (AHN) and the situation is well know.

1.2 Objective

The objective of this report is to research the usefullness and validaty of ICESat-2 in the Netherlands with as main question:

Is ICESat-2 data quality good and valid in the Netherlands?
Where relevant subquestions are:

- How good is ICESat-2 Coverage over the Netherlands?
- What is the quality of ICESat-2 heights over the Netherlands?
- How much do different factors affect quality?

2 ICESat-2 and Validation Data

This chapter discusses the general and technical information of the ICESat-2 satellite mission and the ATLAS instrument it measures with.

2.1 Mission Objectives

The ICESat-2 is on a 3 year mission with 4 scientific objectives:

1. Quantify contribution of melting icesheets to sea level change
2. Quantify ice sheet mass changes regionally
3. Esimate and monitor sea ice thickness and it's change
4. Quantify Earths carbon inventory by measuring vegetation height worldwide

These objectives are currently important while there seems to be a warming earth, which can lead to many negative outcomes for humanity. The fifth IPCC (2014) assessment report states in high confidence that the Greenland and Antarctic ice sheets and glaciers worldwide are shrinking. And also the Antarctic sea ice seems to have regional differences in extent. ICESat-2 is the tool to measure and monitor these changes.

2.2 Orbit

ICESat-2 was launched on September 15 2018 with the Delta II rocket from Vanderberg Air Force Base in California USA (Space Launch Report, 2018). The orbit of ICESat-2 has eight different paths called repeats. One repeat takes 91 days. The altitude is at ±500 km and has a 92° inclination, which makes the satellite able to measure up to 88° North/South (ESA, 2019a). Like ICESat-1, which was at ±600 km altitude and had an inclination of 94° (ESA, 2019b), the high inclination is chosen to get more north/south pole coverage, because this is the main area of interest for the ICESat missions . However the orbits of ICESat-1 and ICESat-2 are not the same and thus the ground tracks will not be the same. There are, however, many crossings of the ground tracks especially in the areas of interest to the ICESat-2 mission (north/south poles), where the ground track density is high (Markus et al., 2017).

While measuring, the time of the ICESat-2 can be split in three parts. Time above ice, above land and above ocean. In these different timeslots different measuring is executed (Markus et al., 2017).

1. Above ice in the polar regions ICESat-2 the objective is to retrieve good temporal data for the changing ice mass in time. So, ICESat-2 will point it's 6 lasers at the same angle/position along the ground tracks (nadir) every repeat.

2. Above land the objective of ICESat-2 is to measure vegetation height and make a dense grid within 2 years of operating. To accomplish this ICESat-2 must measure in multiple off-nadir angles. This way the data will cover more area.

3. Above Ocean the ICESat-2 will do calibration procedures and collect data to make an ocean elevation map.

2.3 ATLAS

2.4 Mabel

The Multiple Altimeter Beam Experimental Lidar (Mabel) is an airborne lidar made to simulate ICESat-2 data. It was developed in 1 year and already did test flights starting from 2010. With these tests, the possiblilties for ICESat-2 were tested and eventually photon counting was deemed possible and ATLAS was made (McGill et al., 2013). While Mabel is made to simulate ATLAS on ICESat-2 it is not the same instrument. To test options with Mabel, Mabel needed to be more flexible (Forfinski-Sarkozi & Parrish, 2016). In the table 1 the main differences between Mabel and ATLAS can be seen.

	Mabel	**ATLAS**
Footprint	2 m	15 m
FOV	4.2 m	41 m
Frequency	5-20 kHz	10 kHz
Laser Setup	16*532 nm and 8*1053 nm	6*532 nm in 3 pairs
Swath	2 km	6km
Filter Width	150/400 pm	30 pm
Aperture	0.013 m^2	0.79 m^2
Altitude	20 km	500 km

Table 1: Differences Mabel and ATLAS (Forfinski-Sarkozi & Parrish, 2016)

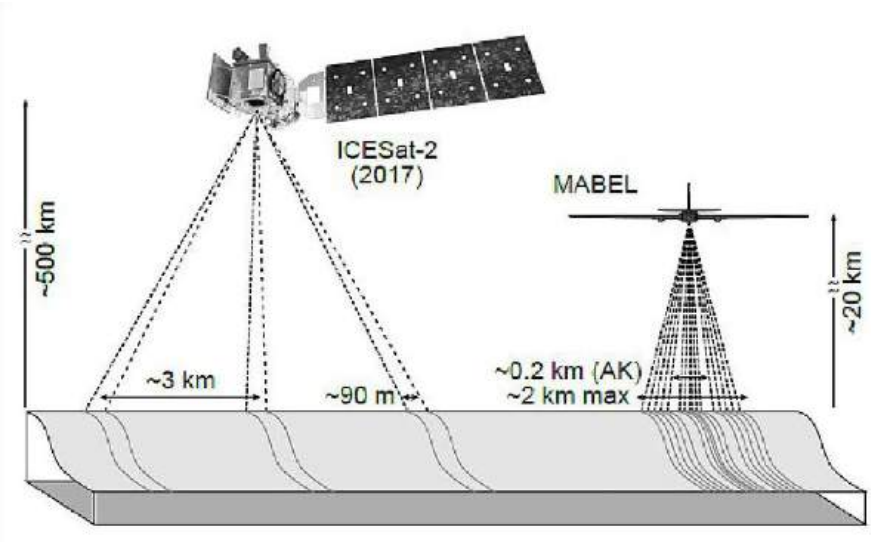


Figure 1: Comparison ICESat-2 & Mabel mission (ESA, 2019a)

With the Mabel mission the ICESat-2 mission was deemed possible. And the specifics of the ICESat-2 and ATLAS like laser setup were determined. Not much data is released of Mabel. Mabel did make prediction of the quality of ICESat-2, however these could not be found.

2.4.1 Laser Setup

The laser instrument emits 6 laser along its tracks. These 6 lasers are set in 3 pairs and each pair has a weak and strong laser. A view of the tracks can be seen in figure 2.

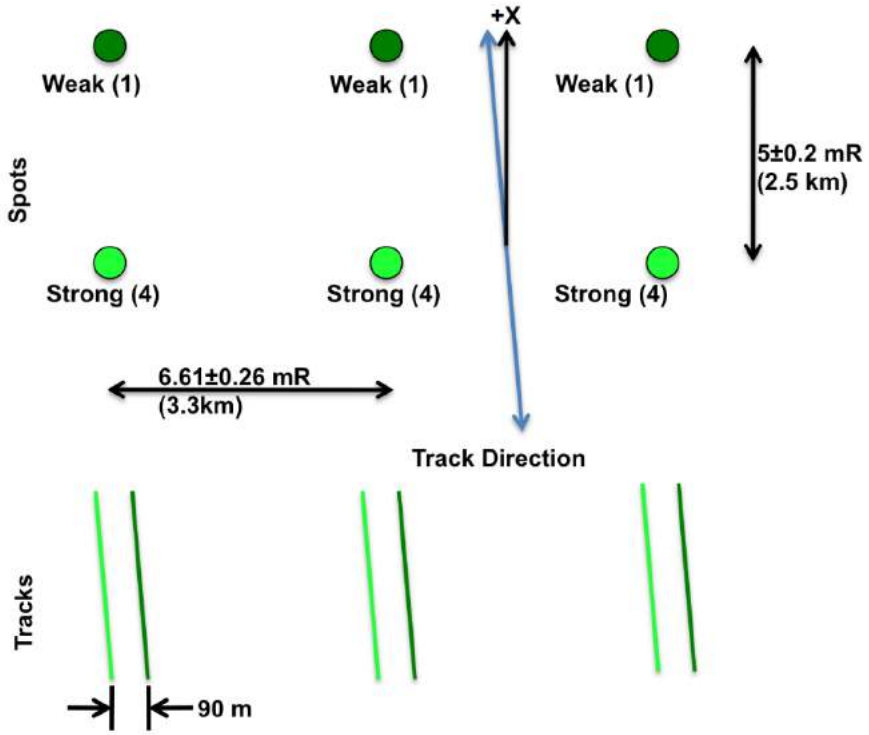


Figure 2: Laser Setup (Markus et al., 2017)

The ICESat-2 measures with pairs (weak & strong) of lasers on both sides of the ground track with 3.3 km between the pairs. This is a development of ICESat-2, which came from a limitation of ICESat-1. On ICESat-1 there were only two relatively close lasers and a slope cannot be estimated with just one line of measurements. So, ICESat-1 needed multiple years in which mulitple lines of measurements are made over the inclined surface after which the incline can be calculated. ICESat-2 solves this problem. Instead of one pair over lasers there are 3 pairs of lasers and between these pairs the incline can be calclulated (Markus et al., 2017).

The spacing within the pairs is set to 90 m. The reason for this spacing is not incline, but roughness of surfaces. The ICESat-2 mission is mainly aimed at measuring the ice sheets so the spacing is set to be optimal on ice sheets/glaciers. The exact spacing is determined from data from a Airborne Topographic Mapper (ATM) mission, which measured the Russel Glacier in Greenland with different pair spacings between 250 m and 25 m to determine the error in roughness as seen in figure 3. The roughness of an ice sheet is mostly under 5 m, where according to the measurements a spacing under 100 m is good for ice sheets and thus for ICESat-2 (Markus et al., 2017). For ICESat-2 spacing can be changed by adjusting the yaw. At the start of the ICESat-2 mission it is chosen to set the spacing to 90

m.

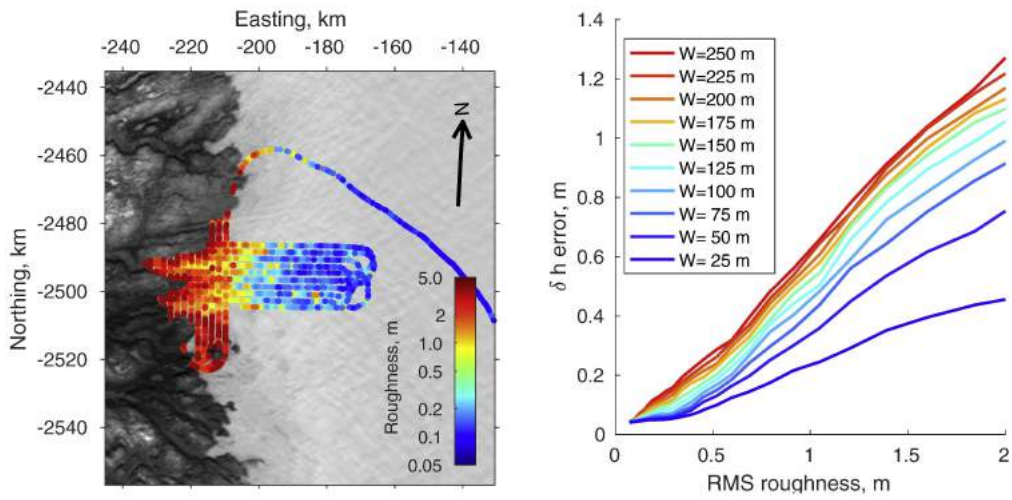


Figure 3: ATM measured Roughness and Precision (Markus et al., 2017) with W as spacing within the pairs.

2.4.2 Lidar Sensor

Lidar works by sending light from an airborne or spaceborne vehicle. That light reflects on a surface and some light is reflected back to the satellite and at the satellite it is measured. This measuring of lidar light is most commonly used with full waveform sensors, which measure photons and their energy in a time interval. Then a profile is obtained from which vegetation height and terrain height can be calculated. ATLAS on the ICESat-2 uses photon counting, which counts individual photons and gives each photon a travel time and location, then from the frequency distribution of the travel times of the photons the vegetation and terrain height can be calculated. The distribution from the histogram is comparable to the full waveform energy distribution (A. L. Neuenschwander & Magruder, 2016). The advantage of using Photon Counting is that a less powerful laser is required. However, a disadvantage is a limited field of view, because otherwise solar light will make for too much noise relative to the signal photons (McGill et al., 2013).

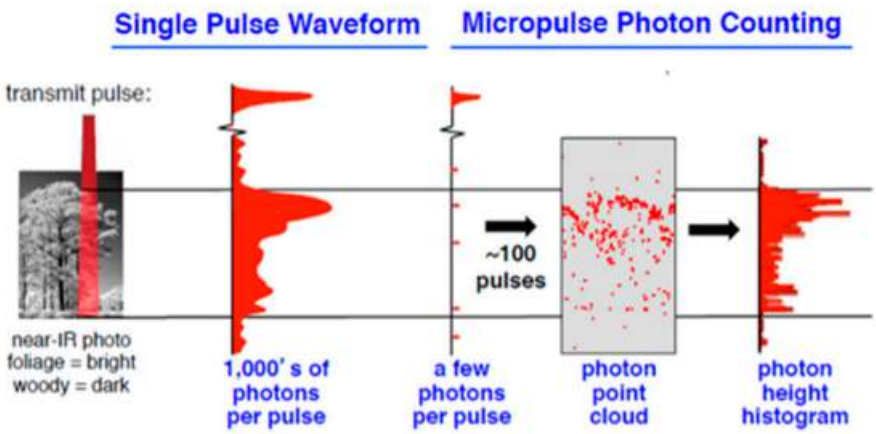


Figure 4: Full Waveform and Photon Counting(A. L. Neuenschwander & Magruder, 2016)

The effectivity and reliablity of the lidar depends on the amount of light returning to the satellite, this is dependant on multiple factors. First off, reflectively on the surface. This can be very different for various land covers. Ice and Snow has a high reflectivity, which is good for ICESat-2. Forest reflects much less than ice and snow, but it also depends on which type of forest. To make an expectation of how effective ATLAS will be above different forests, test data is retrieved with which an estimation of reflectance can be made. Other factors affecting lidar data are reflectors which are not the target surface. The main unwanted reflector is clouds, which is an unavoidable hindrance. Clouds block almost all light as can be seen in figure 5 from Mabel. Another hindrance that can be seen in figure 5 is aerosols which blocks some light, however not all (Markus et al., 2017).

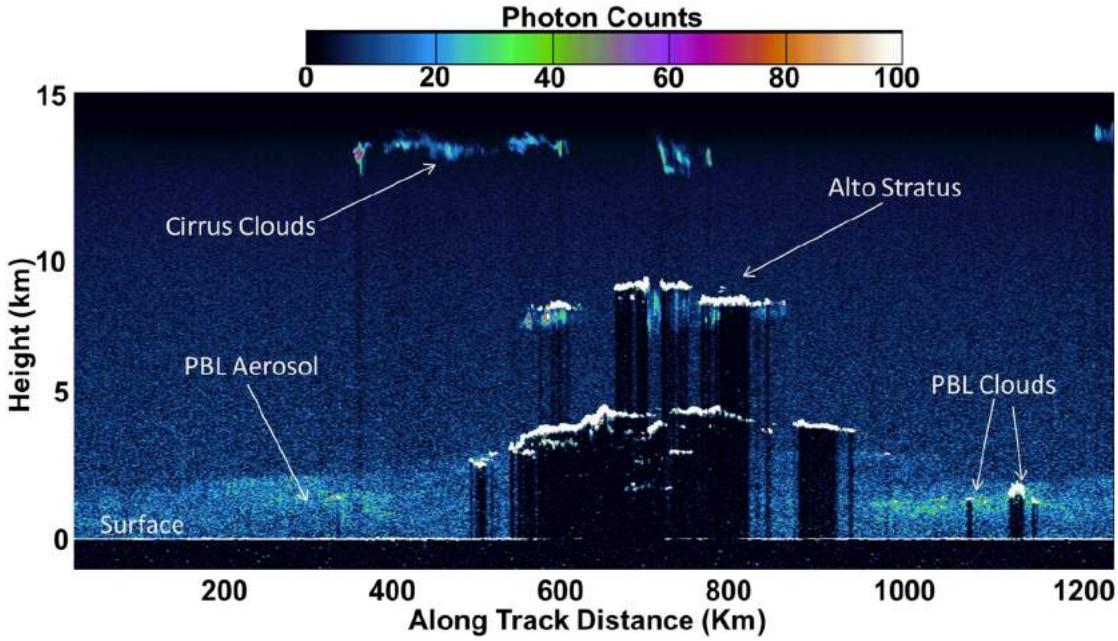


Figure 5: Reflection hinderance (Markus et al., 2017)

2.5 Data Products

ICESat-2 produces 21 different data products named ATL00 to ATL21 (missing ATL05). ATL00 to ATL02 are low level data products with not much processing. ATL03 is the most basic processed data product, the position and elevation of every photon. More interesting products are mulitple ice based data products (ATL06, ATL07, ATL10, ATL11, ATL14, ATL15, ATL20 and ATL21), however in the Netherlands there is no use for this data. In the Netherlands there is more use for the water heights (ATL8, ATL12, ATL13 and ATL19) and vegetation/terrain heights (ATL08, ATL18). In these data products geophysical models are used to correct. For atmospherical corrections NASAGMAO GEOS-5 and MOG2D, for ocean tide corrections GOT4.8, for corrections from changing earth rotation IERS2010 is used (Markus et al., 2017). All product are referenced in WGS84, because NASA, works with this reference. For the northing and easting this is no problem, however the elevation is also referenced to the WGS84 ellipsoid. ICESat-2 does provide the necessary height reference to convert elevations to EGM2008(Markus et al., 2017). It may be better for Dutch applications to convert to the Dutch reference height (NAP).

The choices of data product for analysis of ICESat-2 in this report is ATL03 and ATL08. ATL03 is a more raw dataset in which all geolocated photons with a height

are given, ATL08 is a more processed dataset. ATL08 works in segments of 100 meter in which it takes all photons and filters and derives variables as explained in chapter 2.6. After looking and experimenting with the data it is chosen to use ATL08, because ATL03 has a high noise ratio and in the available time for this study it is not possible to create results using ATL03. ATL08 already has a filtering program and gives variables, which can be used with less processing. Because ATL08 processes 100 meter segments, it uses the middle point of the 100 meter segment as the defined location. This point will be called the data point.

2.6 ATL08 and quality flags

ATL08 is used, which gives terrain elevation and canopy height and does not focus on ice, like most data products. This data product is a further step from ATL03. In ATL03 the raw data is processed to geolocate photons with little noise filtering. ATL08 goes further in noise filtering and classifies the photons as terrain/canopy/top of canopy. The ATL08 algorithm works with 10 km segments in the following steps (A. Neuenschwander & Pitts, 2019).

1. The DRAGANN filter finds an initial approximate terrain level
2. If, according to Landsat Vegetation Continuous Fields (VCF), there is canopy, there is some extra processing necessary to get the initial terrain elevation and in a similar way the initial top of canopy is determined.
3. Classify Photons
   (a) Photons within the point spread function (determined from uncertainty and local topography) from the initial terrain level are classified as ground photon.
   (b) Photons 1 meter under to 4 meter above the initial canopy height are classified as top of canopy
   (c) Photons in between ground and top of canopy are classified as canopy.
   (d) Photons outside any classification range are set as noise.
4. After classifying the condition is applied that if in 500 signal photons there is less than 5% canopy, the canopy photons are set as noise.
5. Then the product is divided in 100 meter segments for all six beams separately. If there are less then 50 photons in 100 meter, the part is invalid.

In the data products there are also flags on the data, which can be useful when looking for data. Some usefull ones are (A. Neuenschwander & Pitts, 2019):

1. Land cover flag (segment_landcover) is the land cover reference from the MODIS mosaic.
2. Cloud confidence flag (cloud_flag_asr) is the confidence in the cloud flag.
3. Photon count (n_seg_ph) is the amount of photons in the segment

More about the data product ATL08 with all flags and specifics can be found in an article by A. Neuenschwander & Pitts (2019).

2.7 AHN reference height

For the reference height AHN is used. AHN is the national elevation map of the Netherlands. The AHN is made from airborne laseraltimetry (AHN, 2018). There are 3 iterations of AHN1, 2 and 3. The first was made in 1994-2004. 3 Years after AHN1 was finished, the second iteration AHN was made in 2007-2012, where the quality was improved relative to AHN1. 2 Years after AHN2 was finished, the third iteration AHN started in 2014 and it is not yet finished (2019) (AHN, 2018). The AHN has a DTM and DSM product. It is chosen to use the DTM, because in the ICESat-2 there is always terrain height, so this is the best for comparison, ignoring vegetation, buildings etc. The second iteration of AHN is used, because the area measured is not limited. AHN3 is not yet finished and in the eastern part of the Netherlands there is no data. No reference means no comparison possible. There is already not too much ICESat-2 data, so data can not be lost due to not having a reference height. AHN1 does have the same coverage as AHN2. However AHN1 has a worse quality, so AHN2 is better for reference.
AHN uses NAP as a height reference. NAP is the Dutch national, of which 0 is approximately mean sea level (Rijkswaterstaat, 2018).

Now that, the ICESat-2 its data and the reference data are explained. The methods using these datasets to get to the validation of the ICESat-2 data are explained in the following chapter.

3 Methodology

This chapter discusses the usage of the data and the methods behind them to get to the validation of the ICESat-2 data.

3.1 Planned path download and extraction

The coverage of ICESat-2 is seen with the ground tracks. From NASA (2018), the ground tracks are downloaded. The ground tracks through the Netherlands are needed so the ground tracks in the eastern hemisphere are downloaded and visualized using QGIS 3.4. The only processing needed is a clip to the extend of the research area. In case of the Netherlands this is done to make the data clearer and more easy to handle. The data over the Netherlands is extracted using a Dutch border polygon layer downloaded from PDOK (2019).

3.2 Overview processing

Other data used for the coverage and quality assessment does need to be processed. In the flowsheet in figure 6 the data processing is shown. The programs used are Matlab, QGIS, Google Earth and GNU wget.

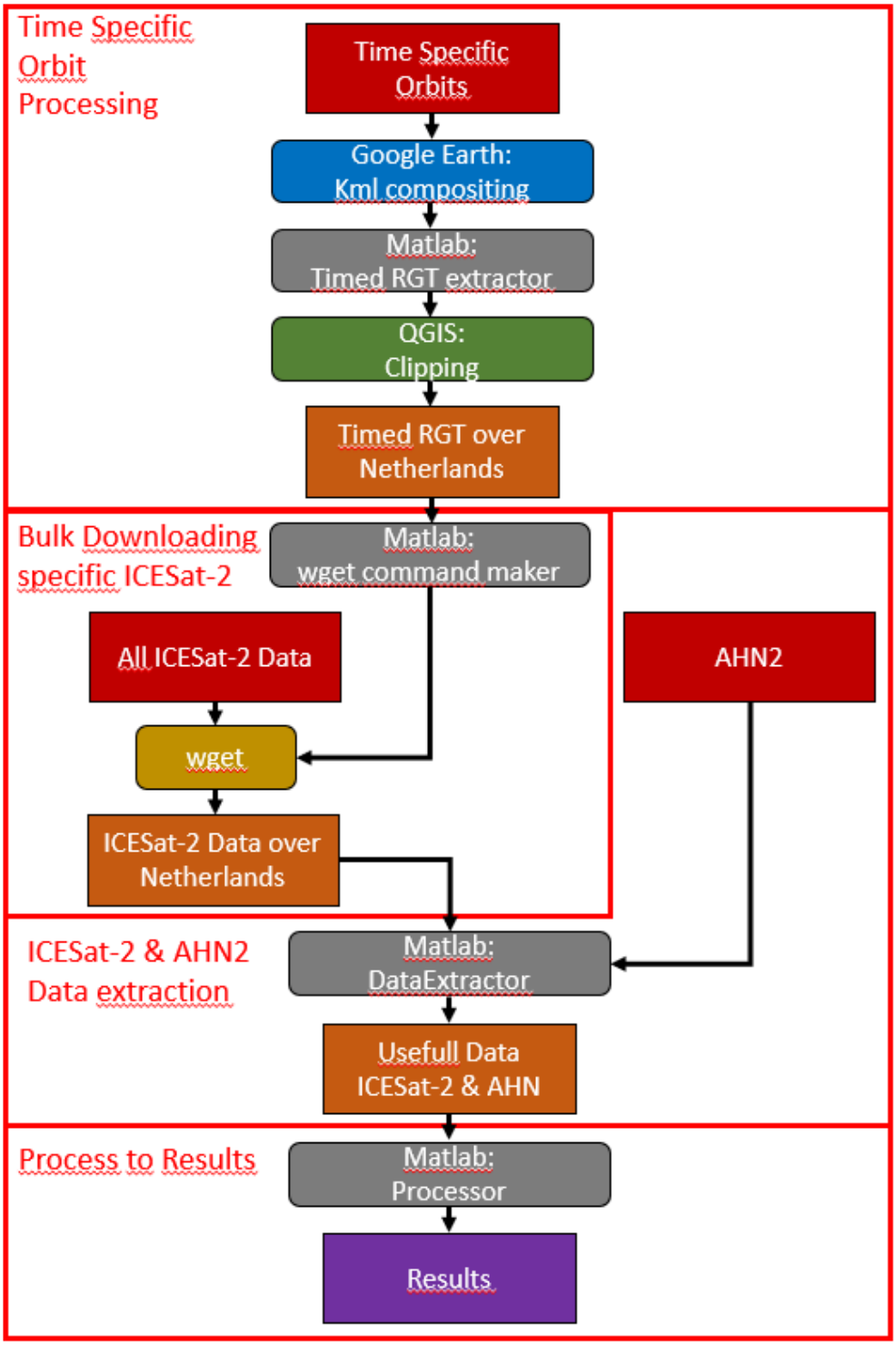


Figure 6: Data Processing Flowsheet

The processing is split in 4 parts:

1. Time Specific Orbit processing
2. Bulk downloading specific ICESat-2 data
3. ICESat-2 & AHN2 Data extraction
4. Process to results

In this sequence the entire workflow will be discussed.

3.3 Time Specific Orbit processing

In the time specific orbit data, the flight path of ICESat-2 is provided time specific, so it can be seen at which time the ICESat-2 passes over an area. This cycle data is given by NASA (2018) in form of 1387 kml files. Each kml file contains time-place data of a period of 1 hour and 34 minutes. The place coordinate is the reference ground point on earth of the ICESat-2. 1387 kml files of 1 hour 34 minutes means a cycle of 90.54 days. The data in the kml is given in 2 forms:

1. a line, which represents the reference ground track of ICESat-2 in the kml time interval (1 hour 34 min) made with 5656 or 5657 points.
2. points, which represent the reference ground position of ICESat-2 in the kml interval. The resolution of the points are a point every minute. The kml time interval is 1 hour and 34 minutes, so there are 94 points in every kml file.

An example of 1 kml with line and points is shown in figure 7.

Figure 7: 1 of the 1387 kml files of cycle 02 with RGT line and minute specific points

The data in the time specific orbit data is for the whole repeat it is running. However, not every 1 hour 34 minutes the repeat goes through the Netherlands, so a lot of the kml data is useless for this research. Also the point data in the kml is useless for position, because the reference ground tracks (position ICESat-2 projected on earth also named RGT) lines are much better and accurate for seeing where ICESat-2 passes. To shrink the data to just the desired data of RGT over the Netherlands, a Matlab script is made. This script imports all the kml files, extracts the RGT lines, checks if they cross the target area (the Netherlands) and if it crosses, saves a new kml file. This code can also be used for smaller target areas, the only thing needed is a center point of the target and the margin of how close a RGT can be. However small targets have a lot less chance of getting a result, especially because there are only 4 cycles available of which the first 3 cycles follow the same repeat. The lines only contain a day as date and not a more

specific time, which is needed to search the data for downloading. Minute specific orbit data is in the points. These points go over the lines, so the closest points on every line to the target area are selected. The line (RGT) is cut at the points in a way that the only part of the line that stays. This part crosses over the target area and goes between the minute specific point before and after is crosses to the target area. These points have minute specific data, so the beginning and end of the line are at the time specific point. This time data is added to the line, so when clicking on the line in QGIS the description says the time interval of the line. This worked for cycle 2.
Working with time specific orbit data from cycles 1, 3 and 4 was harder. They were not working with the Matlab function *kml2struct* from Slegers (2012) and also in Visual Studio Code the contents of the 1387 kml files could not be seen. The only place where the kml files could be imported/exported was Google Earth. So to create a work around and still process the data in Matlab, the 1387 kml files were all imported into Google Earth and exported as 1 kml files with all lines and points with all data. Then the Matlab script had to be adjusted to work with the composited kml file. Looking at the new data another problem occurs, in the cycle 2, ICESat-2 always passes the entire Netherlands within the 1 minute interval. This is not the case for the other cycles, so it was chosen to extract 2 minute intervals. The final script *timedRGTextractot* is shown in Appendix A.1. This script works with every cycle as long as it is composited to 1 kml file.

3.4 Bulk downloading specific ICESat-2 data

The ICESat-2 data can be downloaded from the NSIDC https site via EarthData. The data is seperated in data product, then by day, and then by time of changing orbital segment. To download multiple files GNU wget should be used according to NSIDC. Wget is a command prompt application, which downloads files from https among other types of websites. Following the exact instructions from NSIDC on wget did not work and changes to the command had to be made. Also the goal is to download all the data above the Netherlands, which means being able to distinguish between datafiles before downloading. There is no possiblity of downloading and then checking the location of the data, that would be too much data to download. The structure of a wget command is:

directory/ wget [options] [url]

The url is the webadress at which the data is located. The options are extra functions, which are needed to adjust how and what is downloaded and how the site must be handled. An example of downloading 1 HDF5 file from the EarthData is:

```
wget --http-user=[EarthData username] ...
     --http-password=[EarthData password] --load-cookies ...
     mycookies.txt --save-cookies mycookies.txt ...
     --keep-session-cookies -r --reject "index.html*" -np -e ...
     robots=off https://n5eil01u.ecs.nsidc.org/ATLAS/ ...
     /ATL08.001/2018.10.19/ATL08_20181019034611_03130111_001_01.h5}
```

First some site handling options are used namely logging in with username and password, saving cookies and turning off robot exculsion, also the index.html files, which are automatically created when downloading, are immediately deleted. After this the url+filename are written out so wget knows where to look and which file to download. However, for downloading all data of the Netherlands (from about 30 RGTs as retrieved from the TimedRGTextractor) the exact urls and filenames are needed of the data in the Netherlands. Luckily NSDIC have a logic to the naming of their files. An example url+filename is chosen en shown here:

https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/
2018.10.19/ATL08_20181019000119_03110106_001_01.h5

This url can be split in many different parts, which all have a meaning.

url/filename Part	**Meaning**	**Obtained**
https://n5eil01u.ecs.nsidc.org	Host site	always the same
/ATLAS	Instrument from satellite	always the same
/ATL08	Data product	Chosen
.001	Number of revision	check NSIDC
/2018.10.19	Date of measurement	From RGT
ATL08_	Data product	Chosen
20181019	Date as yyymmdd	From RGT
0001199_	Time as hhmmss	unknown
0311	Number of RGT as ####	From RGT
01	Number of Cycle as ##	from RGT
06_	Orbital segment numer as ##	from RGT
001_	Number version ###	check NSIDC
01	Number revision ##	check NSIDC
.h5	File extension	always the same

To make the total wget command to download all data over the Netherlands, a script in Matlab is made which creates a string, which can be pasted into the command prompt and then downloads the data. This Matlab script asks some parameters and uses the clipped timed RGT to determine the url and filenames in the following way.

The red url's first 2 parts which are always the same for downloading ICESat-2 data, the host site address and the ICESat-2 instrument name. The data product is asked of the script user and has to be a 2 digit number i.e. 04. This can be any available data product. The number of the revision is asked of the script user and it has to be a 3 digit number i.e. 001, it can be found on the NSDIC/EarthData websites. Currently (6-6-2019) revision 001 is the latest and only revision. The Date of Measurement is retrieved from reading the shapefile with the clipped timed RGTs over the Netherlands from QGIS.

The blue filename has 8 parts which are not constant. For the Data product the same is used as in the url. The date is also retrieved as in the url. The RGT number is retrieved from the RGT shapefile from QGIS. The Cycle number is calculated from the date. The orbital segment is always 02 or 06 in the Netherlands. The orbital segment is between 27° N and 59,5° N. The differentiation between 02 and 06 is the travel direction of the satellite moving north (ascending) is 02 and moving south (descending) is 06 in the segment. The direction is calculated from the latitude increasing of decreasing and it is determined if the orbital segment number is 02 or 06. This does limit the script to data between 27° N and 59,5° N, which is no problem for research in the Netherlands. This limitation can be solved by using the location to determine orbital segment lattitde in combination with the direction to determine the orbital segment number. The version and revision number are asked of the script user and can be found on the NSDIC/EarthData websites. This leaves one part namely the time, here is a complication. The time is not seperated by minute like the time specific RGT points, the time is the moment the ICESat-2 enters a new orbital segment. This time is not known. And the time intervals of entering a new orbital segment are not consistent so the time cannot be calculated. This means the exact filename cannot be retrieved and a workaround is needed.

After studying wget options a search function is found which will be the workaround. Instead of the url+filename at the end of the wget command, only the url is at the end the filename is set in front with the -A (accept) option. this option makes it that wget downloads all files in the url, which have a particular string at the end of the filename. So, just the parts of the filename after the time part are specified and a file is downloaded if it ends exactly the same. As described above this entire last part can be exactly determined. The Matlab script makes the wget command for every RGT over the Netherlands and joins all the separate commands together with ampersands, so when the joined command is pasted in the command prompt all data over the Netherlands will be searched and downloaded from the NSDIC/EarthData site. The Matlab script is shown in Appendix A.2. And the generated wget command is shown in Appendix A.3.

3.5 ICESat-2 & AHN2 Data extraction

3.5.1 Position & index of ICESat-2 data retrieval

Now that the HDF5 datafiles are obtained the needed data needs to be extracted. First the datafiles are screened to get the directories and to see, on which passess actual data was retrieved and the 'Data' struct variable is created to store all needed data to make it easy for further processing.

The first data retrieved is the latitude and longitude of the data points. For data retrieval of the HDF5 file a self made Matlab function is used called *ATL08import* seen in Appendix A.4. This function needs the HDF5 filename, beam name and the name of the variable in the HDF5 file and creates a simple vector with the requested data. The latitude and longitude is first of all used to get the index of the points which lay inside the Netherlands. The indices of these points are used to make the rest of the processing faster, because there are datapoints of the entire orbital segment in the HDF5 file and working with all datapoints is slower then just using the points in the Netherlands. To get the limits of the Netherlands a shapefile is used from PDOK (2019) with the Dutch border. The latitude and longitude of all the datapoints in the Netherlands is stored in the Data struct.

3.5.2 AHN2 retrieval

With the longitude/lattitude data the reference height (AHN2) can be downloaded. For this the AHN functions of Hans van der Marel are used. These functions download the right AHN data from the input coordinates. First the latitude and longitude are converted from WGS84 to RD, the coordinate system of the Netherlands, then for every datapoint, the AHN tile on which it is positioned is determined. This tile is then downloaded if it is not already downloaded and the AHN height of the datapoint is retrieved and stored in the Data struct. If an error occurs when searching the AHN tile or downloading the tile, the AHN height is set to NaN and the datapoint is not used for height difference.

3.5.3 ICESat-2 Heights retrieval

Then the heights are retrieved from the HDF5 file. However, ATL08 gives mulitple height variables for the 100 meter segment namely:

1. median terrain height of segment (h_te_med)
2. mean terrain height of segment (h_te_mean)
3. max terrain height of segment (h_te_max)
4. min terrain height of segment (h_te_min)
5. best fit terrain height at middle (data) point (h_te_besfit)

6. interpolated terrain height at middle (data) point (h_te_interp)

The AHN height is a height at the datapoint (point in the middle of the segment), so to get a accurate estimation of accuracy and precision a height at this point should be used. This leaves interpolated and best fit height, as the others do not give the height of the middle of the segment, but a statistic of the height over the segment. Best fit is a linear or a third or fourth order polynomial fit of the height, while interpolated is more directly the height at the midpoint from close classified photons. The interpolated height is used as the terrain height of the data point, as the interpolated height is more directly from the ICESat-2 data, it will give a better view of the quality of ICESat-2. This ICESat-2 terrain height is downloaded for all data point in the Netherlands using the indices from the longitude/latitude. This height at the data point will be compared to the AHN height however, the AHN height is in NAP, the Dutch national height reference, and the ICESat-2 height is in WGS84 ellipsoidal height. This needs to be the same to make a good evaluation of accuracy and precision. So, the ICESat-2 heights are converted to NAP heights, because NAP has a more comprehensible value in the Netherlands than the WGS84 ellipsoidal height. This conversion is done using the *etrs2nap* Matlab function made by Hans van der Marel. After the height reference is made the same, the ICESat-2 height is stored in the data struct.

3.5.4 ICESat-2 flag & other variable retrieval

Lastly, some variables are imported from the HDF5 file, which may give more information how different factors affect the accuracy. These variables are:

1. Amount of photons (n_seg_ph). The number of photons in the segement from which the height is calculated.
2. Cloud flag (cloud_flag_asr). A number from 0 to 5 where 0 is high confidence clear skies and 5 is high confidence clouded skies.
3. Landcover (segment_landcover). MODIS land cover classification with 16 classes.

With these variables it can be seen if different factors affect the quality of the ICESat-2. Also the quality difference between the different passes and between the different beam strengths. When the data extraction script is finished the Data struct is saved this struct has all necessary data. The Data struct is used in further processing to get results. This is done so when processing results not the entire downloading en retrieving data from the HDF5 and AHN has to be done every time.

3.6 Validation Metrics

In this research, the validation is focused on the quality of the data, meaning bias and precision of the measured heights. If the quality is good enough to accomplish the scientific goal and there are no big errors, which make the data unreliable, the ICESat-2 data is valid.

The quality is retrieved from the AHN height being subtracted from ICESat-2 creating the height error and stored in the data struct for further result processing. With all the data in the Data struct the results will be processed. The height differences are put in vectors. With all vectors, calculations are done and height error histograms are made to visualize and get results, by fitting distributions to the histograms and from these distributions retrieving the bias and precision. Bias is defined as the error (distance from zero) of the average of the fitted distribution. This can be negative (ICESat-2 is too low) or positive (ICESat-2 is too high). Precision is in a normal distribution the standard deviation (sigma), where there is 68% chance that a measurement lays within the range of 1 sigma from the mean. A normal distribution is a bad fit for the height error histogram, so a t-distribution is used. A t-distribution is like a normal distribution with more prominent tails. This leads to one extra parameter ν. Because of the bigger tails in a t-distribution the sigma does not have the exact same characteristics. This means sigma cannot be used in the same way to get the precision. Instead the precision is calculated from the cdf function of the fitted t-distribution (using the Matlab function *precision* Appendix A.6). Then to negate errors of reference data and calculations/conversions, the law of total variance is used.

These methods are all used to get the answers to the research questions of. The ICESat-2 coverage results are discussed in chapter 4 and the quality validation will be discussed in chapter 5.

4 Planned ICESat-2 over the Netherlands

This chapter discusses using available data of the planning of ICESat-2 to get an indication of the coverage of ICESat-2 over the Netherlands.

4.1 Repeats

The coverage of ICESat-2 over the Netherlands can be seen from the ground tracks. There are 2 types of ground tracks. The reference ground track (RGT), which is the satellite path projected on earth's surface, and the beam ground track, which is the path of the beam from ATLAS on earth's surface. The beam pattern seen in figure 2 can also be seen in the downloaded beam pattern. To make this clear a zoomed-in view of the ground tracks is displayed in figure 8, where the same spacing can be seen. Through figure 8 there are 2 overpasses of ICESat-2, both leaving 6 beam ground tracks (green) and a refererence ground track (purple) under the actual ICESat-2. From the arrows on the ground tracks the direction of ICESat-2 can be seen.

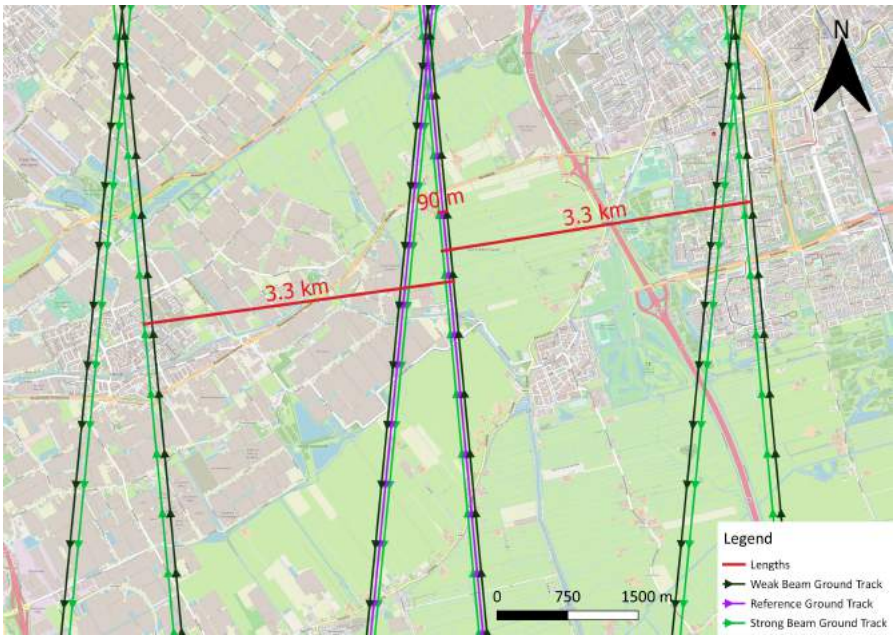
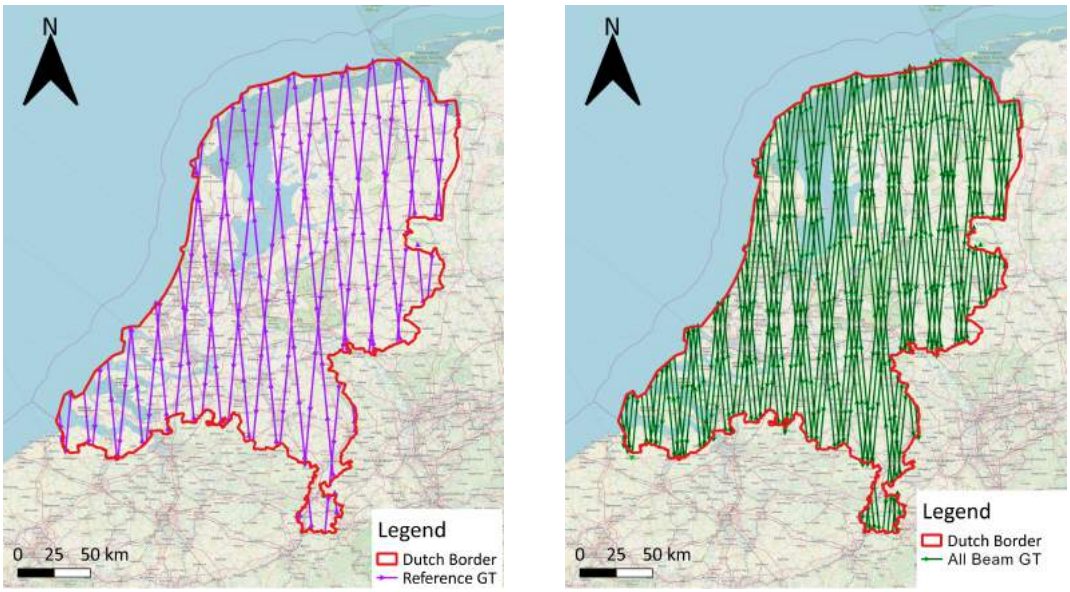


Figure 8: Beam Spacings Visualized with Repeat 1 west of Delft

ICESat-2 passes over the Netherlands many times within the same repeat. A repeat is a collection of paths which in its entirety can be completed in 91 days and cover the enitre globe. These paths in repeat 1 over the Netherlands are visualized in QGIS. The Dutch border can also be seen in figure 9. The reference ground

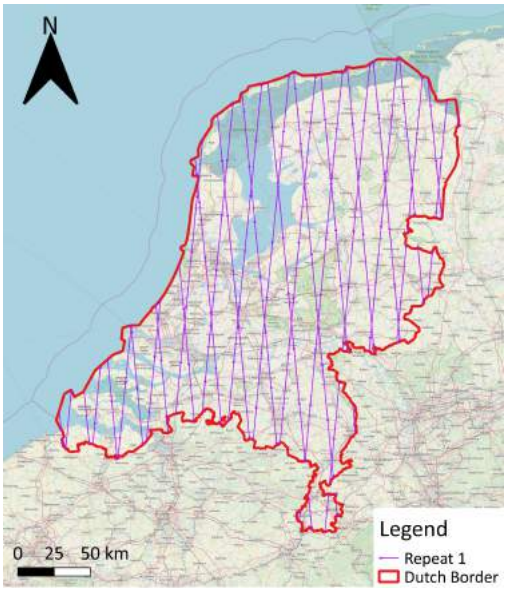
tracks are the actual passes from ICESat-2 over the Netherlands, the RGT. The reference ground tracks create a constant diamond pattern and because ICESat-2 measures with multiple beams, the beam ground tracks seen in figure 9b are multiple constant diamond patterns with a small offset. The diamond pattern means that not every area has a measurement close to it. East and West of places where ground tracks cross there are no close measurements. However one of the objectives of ICESat-2 is to create a dense map of worldwide height of vegetation, these areas also need closer measurements, so repeats where other beam ground tracks are used must be done.



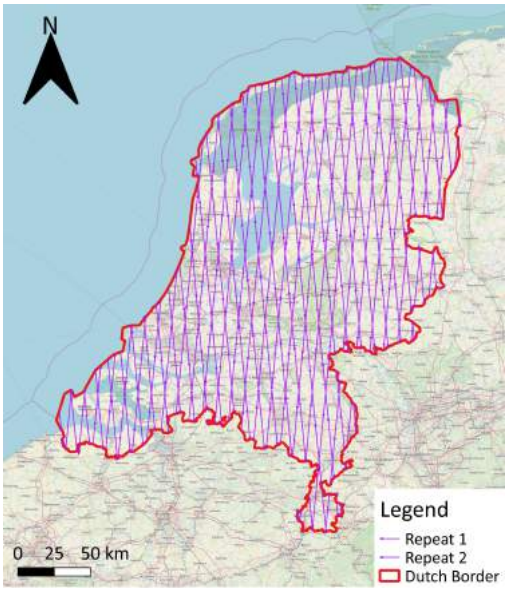
(a) Reference Ground Tracks (b) Weak and Strong Beam Ground Tracks

Figure 9: ICESat-2 Ground Tracks Repeat 1 over the Netherlands

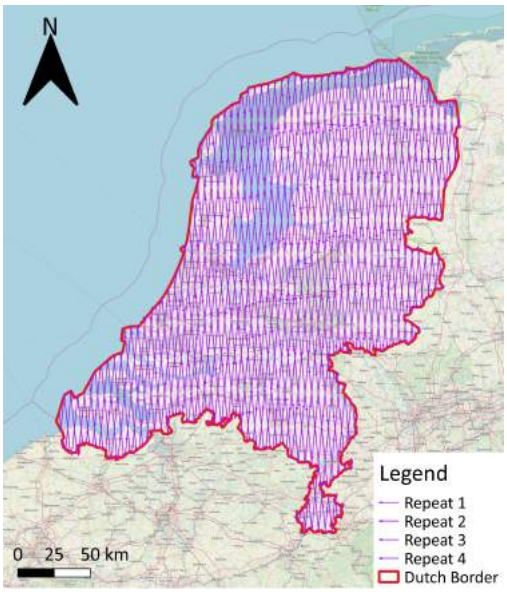
The following repeats have different ground tracks to make for a good dense network of measurements. In QGIS the ground tracks with 1,2,4 and 8 repeats from NASA (2018) are displayed, because with every power of two repeats (1,2,4 and 8) the diamond pattern is constant. The Data from NASA (2018) limits us to 8. With 8 repeats seen in figure 10d the network seems quite dense. However, at this scale it is hard to see and these purple ground tracks are just the reference tracks and not all the beams. To get a better view of the 8 repeat network, this network is displayed in figure 11 at a larger scale and with the weak and strong beam ground tracks.



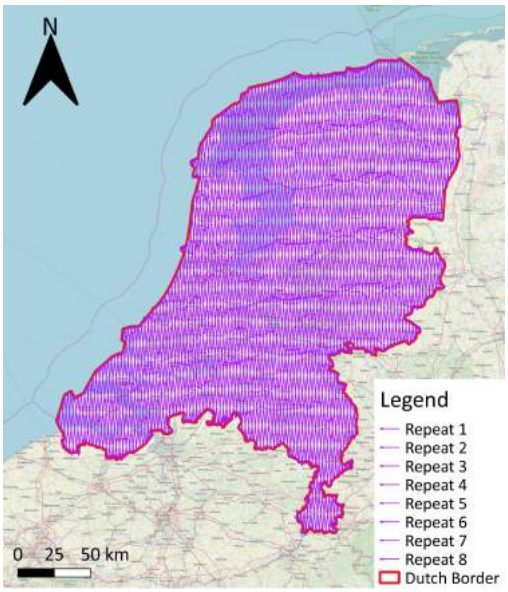
(a) Repeat 1 GT



(b) Repeat 1 & 2 GT



(c) Repeat 1, 2, 3 & 4 GT



(d) Repeat 1, 2, 3, 4, 5, 6, 7 & 8 GT

Figure 10: ICESat-2 Ground Tracks Repeats over the Netherlands

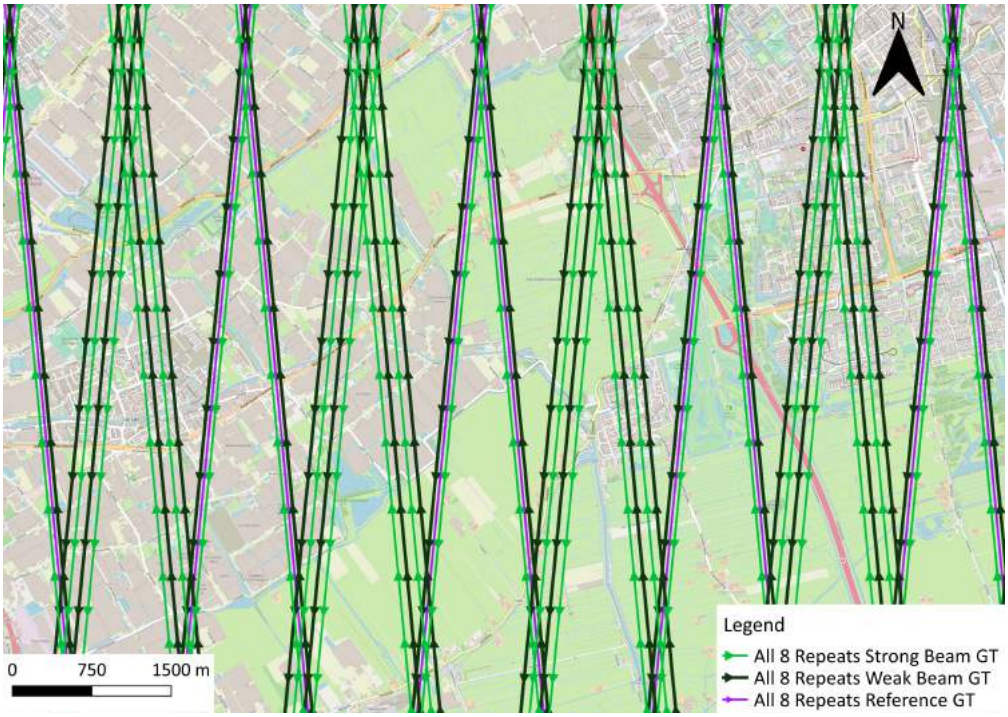


Figure 11: ICESat-2 Beam Ground Tracks all Repeasts west of Delft

All 8 repeats together give a good coverage over the Netherlands as seen in figure 10 and 11. However, getting all these repeats done takes time. ICESat-2 takes 91 days to do a repeat 1 time and it will take multiple passes over every repeat to actually get data over the whole repeat. This is why the ICESat-2 mission is set to take at least 3 years. Then the coverage with all repeats should be good.

4.2 Temporal frequency and Time specific orbit

The repeats are just the different paths ICESat-2 can measure. It says nothing about the temporal frequency of measurements in a point. This depends on the revisiting time, which is the time between measurements of the same point. To get an indication of the revisiting time, the flight plan of ICESat-2 is needed where it says which repeats are flown at different times. Doing one repeat costs 91 days. The measuring cycles are then also divided in 91 day intervals. NASA (2018) provides (on 27-5-2019) the first four cycles which have been run and have to be run. Cycles 1,2 and 3 do repeat 1 and Cycle 4 will do repeat 2. No more of the planning is revealed, so it is not possible to get the revisiting time and the overall temporal data collection plan.

The *timedRGTextractor* script, Appendix A.1. To get the RGTs over the Netherlands as center point 52.156215° N, 5.389725° E in Amersfoort is chosen with margins 3° Latitude and 1.5° Longitude. The script found 30 RGTs which pass over the target area in the cycles.

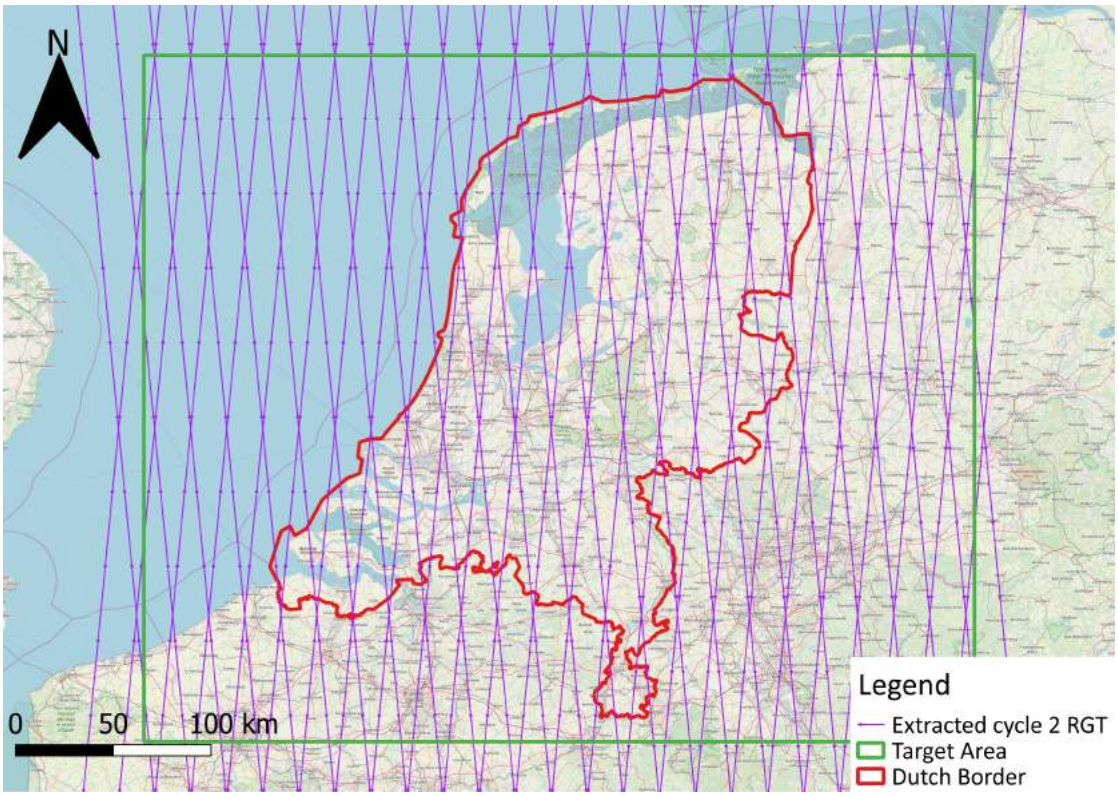


Figure 12: Cycle 2 RGTs over the Netherlands

By using the RGT lines with date and time information the right data in the ICESat-2 library can be found and downloaded. The photons of the ICESat-2 are geolocated, so from the latitude and longitude, the actual measured ground track can be seen. There is a difference between the ground track from photon data and from planned repeat. This offset is shown in figure 13.

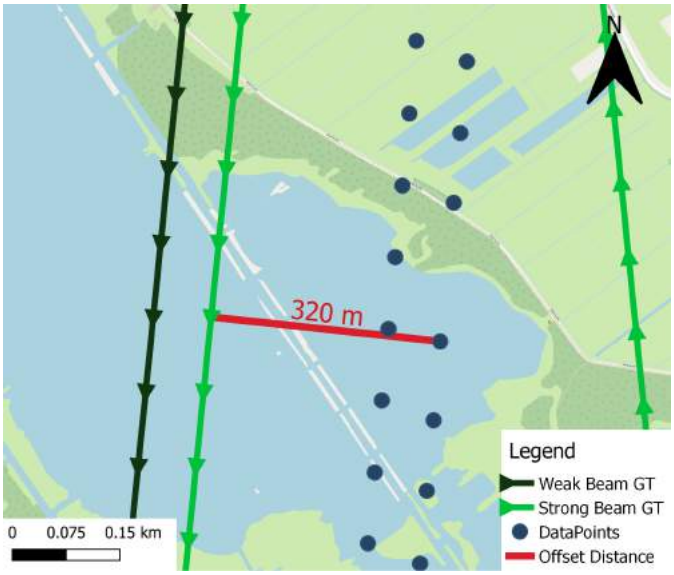


Figure 13: Offset between planned repeat and actual data

According to NASA (2018), the predicted ground tracks have a max error of 100 meter and a few minutes. It is unclear if they mean the time specific orbit tracks or the repeat tracks. The specific orbit tracks of the beams do not exist so the actual data can only be compared to the planned repeats. There seems to be a constant offset of 320 meter. This is much bigger than the 100 meter, so the predicted error is either to the time specific orbit, much larger than expected, or the planned repeat paths were adjusted without updating the publicly available planning.

With the planning of ICESat-2 measurements the coverage is discussed and used in the downloading of the right ICESat-2 data. This data is processed and resuls in the validation of ICESat-2 in the Netherlands, which is discussed in chapter 5.

5 ICESat-2 validation results in the Netherlands

This chapter discusses the validation of quality of ICESat-2 in the Netherlands by looking at all ICESat-2 heights referenced to AHN heights. The ICESat-2 data is also split in parts to see what parameters influence quality.

5.1 Datapoints

According to the time specific time orbit ICESat-2 passes the Netherlands 30 times. These 30 passes are looked for by wget, however wget only found 16 passes. For the passes which are not found the data is not deemed good enough. Every one of the 16 passes measures with 6 beams this means there should be data for 96 beams. However, there are only 69 beams with data. The missing data is here also deemed not good enough. From the 69 beams there are 34397 positions where ATL08 data is retrieved.

Planned Passes	30	100 % of passes
Available Passes	16	53.3 % of passes
Available Beams	96	100 % of beams
Beams with Data	69	71.9 % of beams
All Data Points	34397	100 % of Raw Data

Table 2: Summary of amount of Data

In figure 14, the latitude and the longitude of all data points are plotted on a map grid with the Dutch border from PDOK (2019). Every data position is a red dot. A pass can be recognised by the 3 parallel red lines. The 3 red lines are actually 6 red lines, 1 for every beam. However, because the weak beam and strong beam are close to each other they are indistinguishable at this scale.

Only 1 pass seems to have 3 continuous red lines over the whole of the Netherlands. All other passes have either gaps or there are just a few data points present. Again missing data is due to data being rejected. If there are not enough signal photons in a segment, due to clouds for example, the data is not usable, so it is not made available.

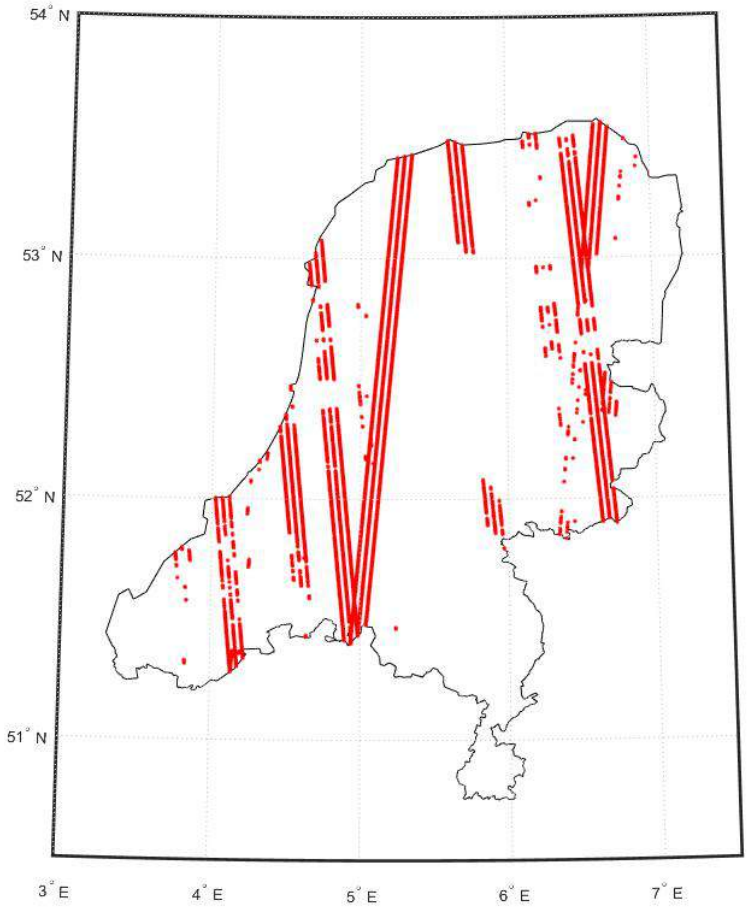


Figure 14: Map with all available data points (15-10-2018 to 9-1-2019) in the Netherlands

5.2 Quality of all data points

The accuracy of ICESat-2 is determined by the height error. The height error is determined by the difference between the ICESat-2 heights and reference AHN2 heights. This data is already calculated and available in the data struct, this process is described in chapter 3.5.1. In total at 28345 datapoints a AHN2 height is found, which is 82.4 % of all points. From all the data points the height difference is plotted in a histogram, seen in figure 15, to visualize the accuracy of ICESat-2. First over this histogram a normal distribution is fitted. However, as can be seen in figure 15, the normal distribution is a bad fit mainly due to the tails, which are

too big. This is solved by using a t-distribution which is like a normal distribution with larger tails. The difference between the normal and t distribution is in the extra ν parameter in the t distribution, which can be seen along with the σ and μ parameters at the top left of figure 15. Where μ is the position parameter (average) and ν and σ are the shape parameters.

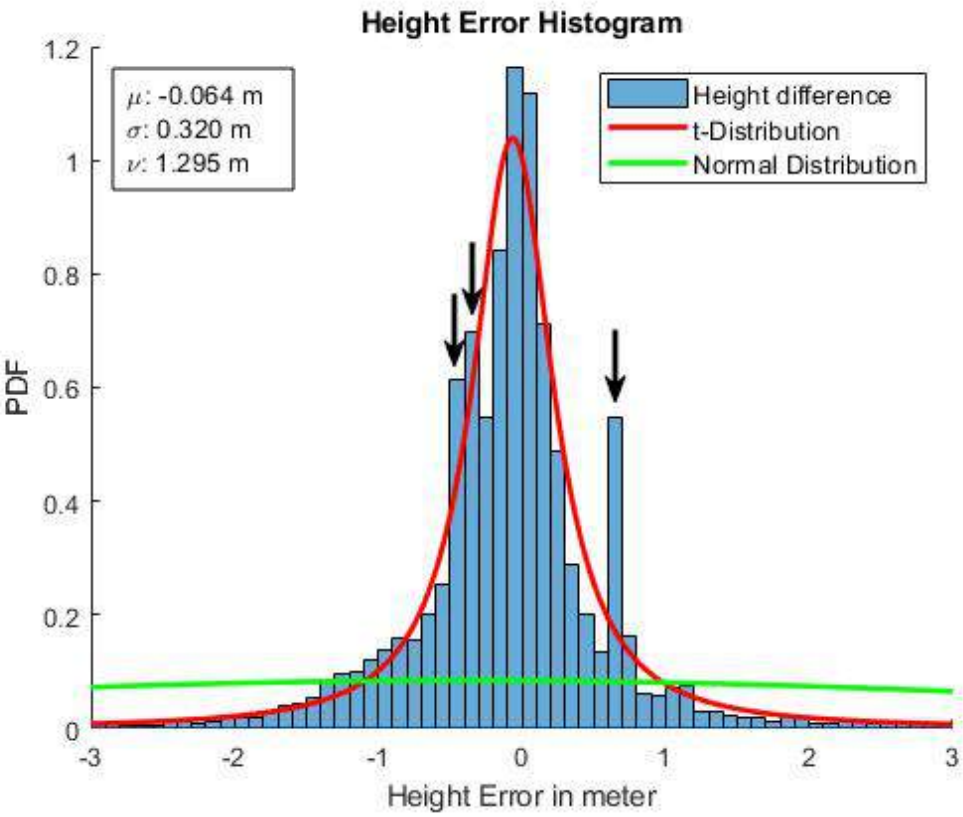


Figure 15: Height Error Histogram all points with AHN available

There are 3 bars, which are big outliers in this histogram. These outliers are highlighted by arrows in figure 15. The outliers skew the fitted distribution. To determine if this is a fault of ICESat-2, the cause for these outliers must be found, because if it is not the fault of ICESat-2 it should be ignored as it has nothing to do with the quality of ICESat-2. In figure 16, the histogram is converted to a stacked bar histogram per pass over the Netherlands to see if the outliers come from a specific pass.

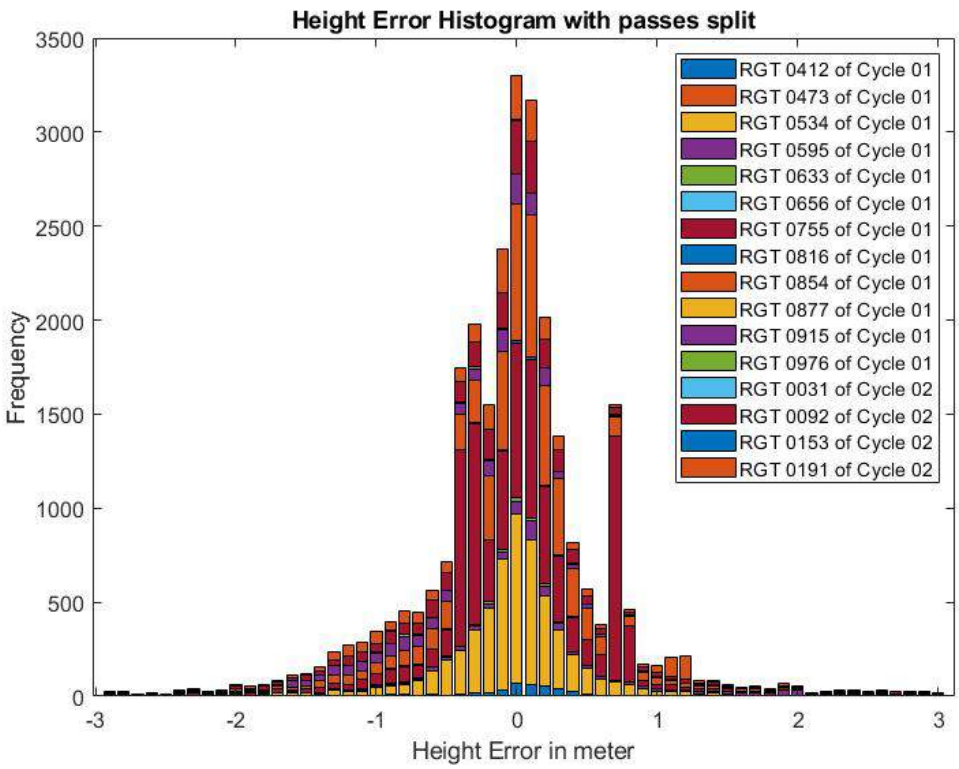


Figure 16: Stacked Height Error Histogram

It can be seen that the seventh pass RGT 0755 of cycle 01 is responsible for all outliers. The seventh pass is the pass with most data points of all passes. A lot of the data on this pass is over the IJsselmeer and Waddensea. This could be the problem, because the AHN data does not work over water. The script should give and AHN height of NaN if there is no AHN data, so it will be ignored. However, it could be that at this went wrong. To check if the water is the cause, the land cover data from the ICESat-2 is used by making a histogram of the height error of only data points classed as water. This histogram is shown in figure 17

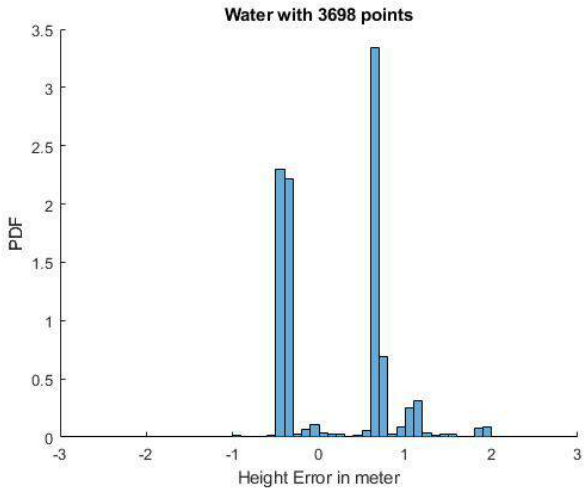


Figure 17: Water Land Cover Height Error Histogram

From figure 17, it can be clearly stated that the datapoints over water are the cause for the outliers. The 3 high bars are at the exact place of the outliers (probably the height of the IJsselmeer and Waddensea). This means the fault is not of ICESat-2, but from the AHN data. ICESat-2 can measure over water so the ICESat-2 heights over water could be accurate, but without good reference data it cannot be quality checked. This reference data is hard to get right due to varying water levels in time. In all further results data points over water are ignored. This means less points will be used.

All Data Points	34397	100 % of Raw Data
AHN available	28345	82.4 % of Raw Data
AHN available & Non-Water Data	24647	71.7 % of Raw Data and 100 % of Points

Table 3: Usable Data summary

Again the height error histogram is made. Now without the water points.

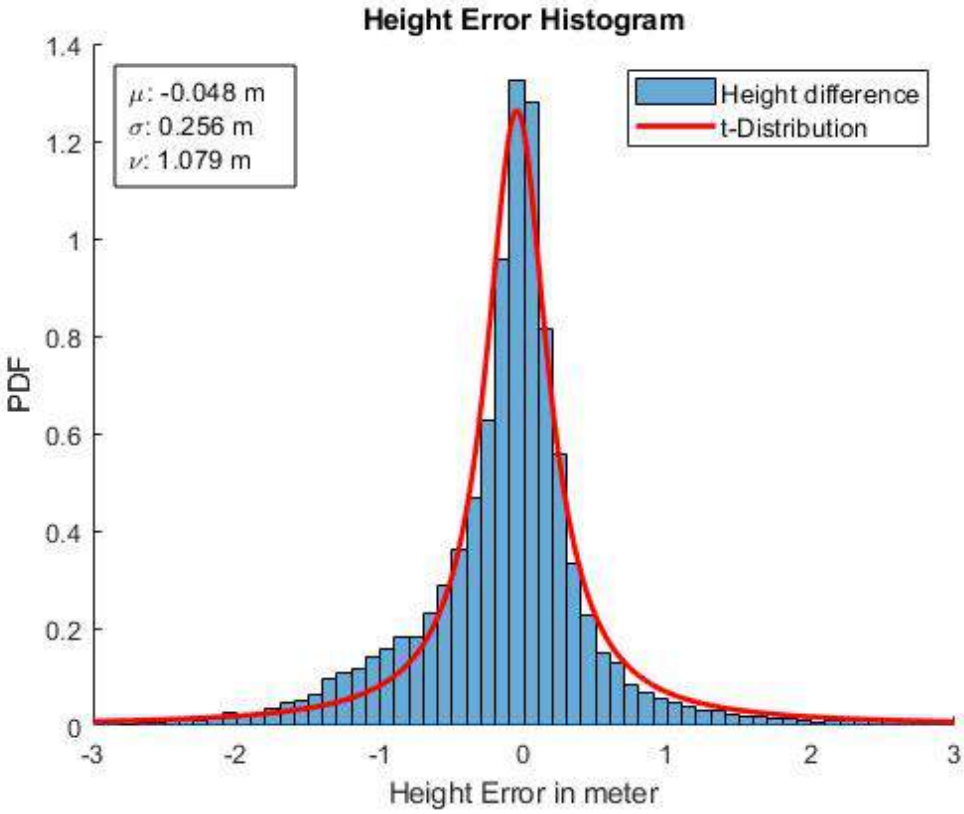


Figure 18: Height error Histogram with all usable data

Now there are no big irregularities in the histogram. The tail at the negative side of the histogram does seem to be constantly bigger. This means ICESat-2 is more likely to give a too low height. With the height error histograms splitted in different ways, maybe the cause for this will be found.
The bias of the height error is the offset of the average from the zero. For the bias the mu off the t-distribution can be taken. So, the bias of all data points is -4.8 cm. This is calculated using 24647 points and the standard deviation of the bias is the bias divided by the square root of the number of measurements, so the standard deviation of the bias is 0.031 cm.

The precision is equal to the range from the mean where the there is a 68% chance that a measurement will be in that range. The precision of all data points is 44.2 cm with a standard deviation (likewise calculated as standard deviation of bias) of 0.28 cm.

The precision can not yet be directly related to ICESat-2, because there are more

factors which contribute to the height error. The AHN data and conversion of data are not perfect and this will also lead to a worse bias and precision. AHN has a preciseness of 7 cm (AHN, 2018) and the conversion calculations has a approximate preciseness of 6 cm. With the variances the total precision can be split so the precision of ICESat-2 is determined. This leaves an ICESat-2 precision of 43.5 cm. This is not much change, because the precision of AHN and the conversions are much better.

The aim of ICESat-2 over land is to get an index of the amount of vegetation over the globe for this they want to make an 3-m precise vegetation height map (Birkett, 2011). The vegetation height is the difference between terrain height and top of canopy height. If both heights have a precision of 2.12 m, this is achieved. From all the data points the ICESat-2 precision is 43.5 cm with a bias of -4.8 cm. This seems to be good enough for the land goal of ICESat-2. On ice and water surfaces ICESat-2 should be significantly more precise.

To give a better view of the actual data used and its quality ,a cross section along a beam ground track is made with AHN2 height and ICESat-2 height to show the height difference. And also the data points are plotted on a map, like figure 14, with the height difference visualized in the color of the points.

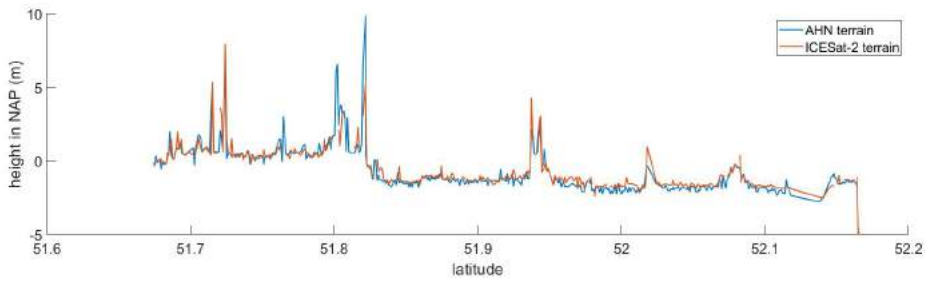


Figure 19: Cross-Section along Beam Ground Track

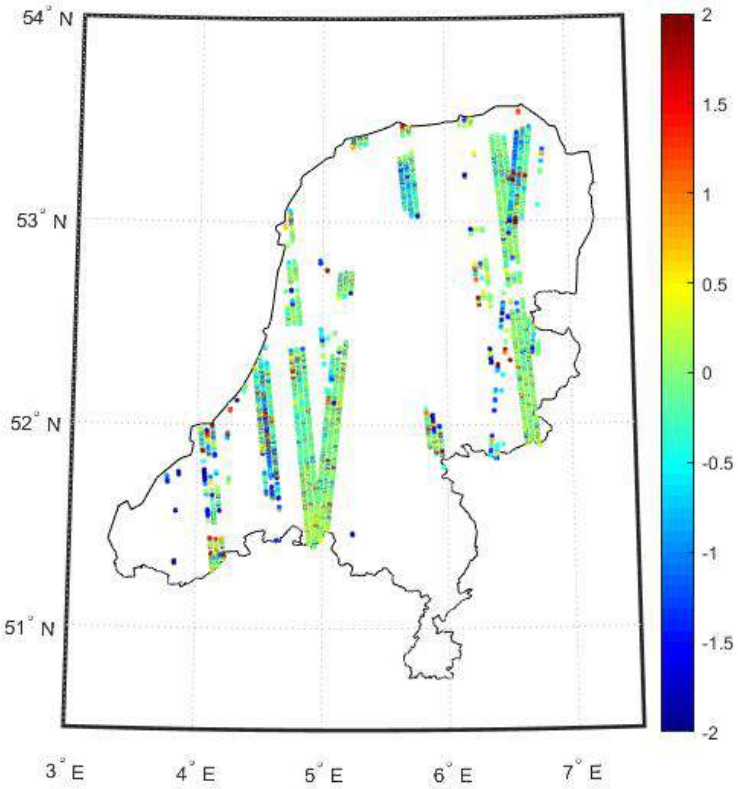


Figure 20: Data points with height difference in the Netherlands

In both figure 19 and 20 it is clear ICESat-2 does give values similar to AHN. In figure 19, the ICESat-2 line follows the AHN line well. However, this is a small amount of points from, which further conclusions can not be made. In figure 20, most point are close to 0 height difference, but there are some areas worse than others and more times ICESat-2 is too low than too high. Like seen in the histogram.

To see what affects the quality of ICESat-2, the data is split in groups using different passes, beams and variables given by ICESat-2 to see where quality might be particularly bad.

5.3 Quality per Land Cover

The ICESat-2 ATL08 data gives a land cover classification for every data point. These are used to see which land covers give good measurements and which land

covers make for a bad influence on the ICESat-2 measurements. ICESat-2 uses the MODIS land cover classification (MCD12Q1). This land cover product uses 17 classes. Every class has a value to recognise it. This value is given in the Data point. Not every land cover is present in the measured data in the Netherlands and not every land cover which is present has the same amount of data points. In table 4 the amount of data per land cover is shown.

Value	**Class label**	**Number of Points**	**Percentage of all Points**
0	Water (removed)	0	-
1	Evergreen Needleleaf Forest	449	1.82 %
2	Evergreen Broadleaf forest	0	-
3	Deciduous Needleleaf forest	39	0.16 %
4	Deciduous Broadleaf forest	1	0.00 %
5	Mixed forest	1731	7.02 %
6	Closed shrublands	6	0.02 %
7	Open shrublands	4	0.02 %
8	Woody savannas	189	0.77 %
9	Savannas	22	0.09 %
10	Grasslands	4821	19.56 %
11	Permanent wetlands	220	0.89 %
12	Croplands	5303	21.52 %
13	Urban and built-up	4290	17.41 %
14	Cropland/Natural vegetation mosaic	7517	30.50 %
15	Snow and ice	0	-
16	Barren or sparsely vegetated	55	0.22 %

Table 4: Data per Land Cover

The more data points a land cover has, the better the quality of ICESat-2 over that land cover can be determined. Land covers with less than 100 points are not good enough for an interpretation so these will be ignored. Of the land covers with more than a 100 points the height error histogram is shown in figure 21. In these histograms irregularities can be seen and the bias/precision can be determined.

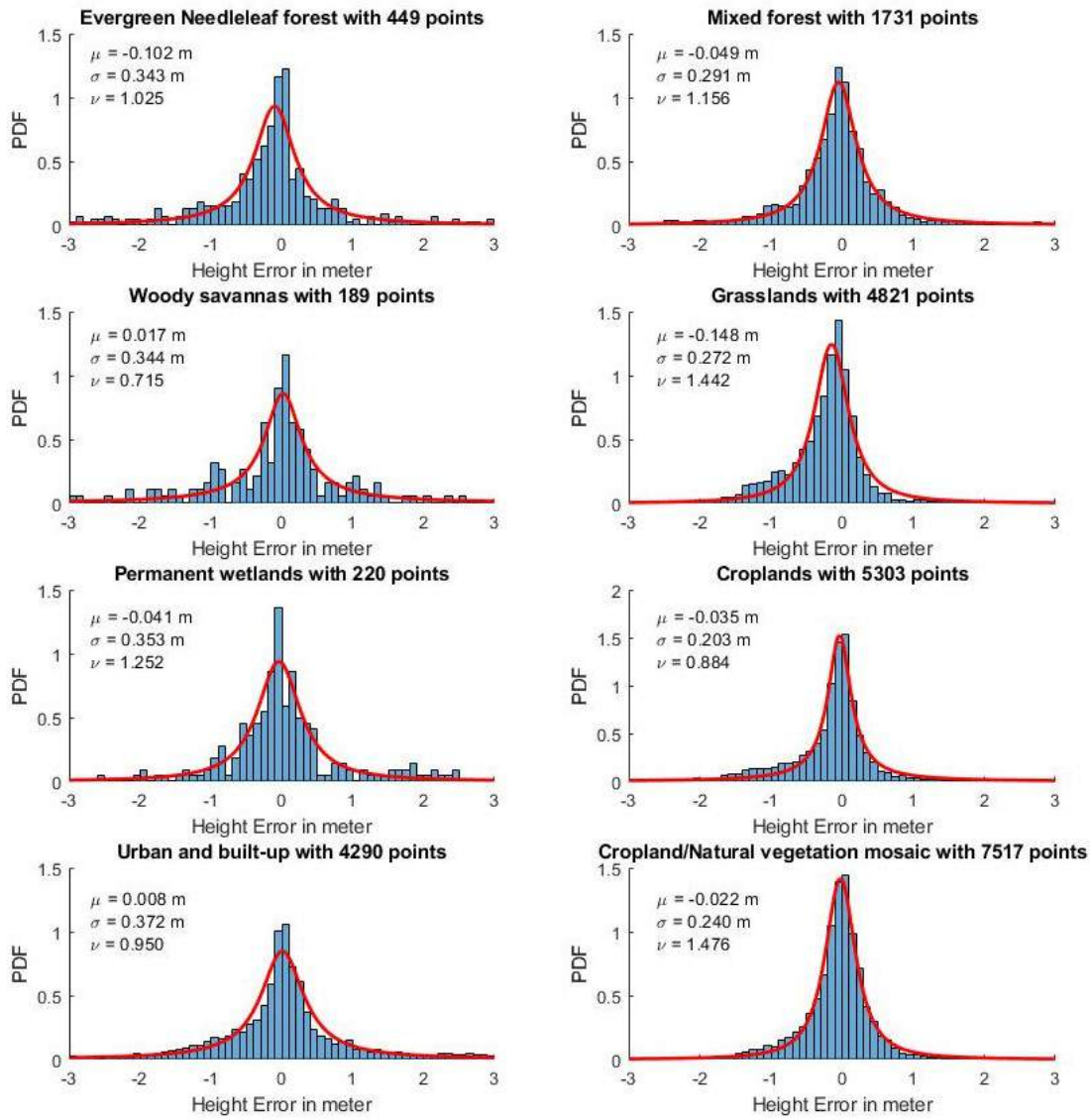


Figure 21: Height error pdf of all Land Cover with more than 50 datapoints

There seems to be no big irregularities. Only in the 3 histograms with less than 500 data points, the histogram has some small irregularities. However, this is likely due to the limited data available. In all histograms the negative tail is bigger than the positive tail as also seen in the complete data point height error histogram. However, this occurs in all land types so a specific land type is not the cause. The biases and precisions from the histograms are different. To get more clarity on the biases and precisions, they are displayed in table 5.

Land Cover	Bias (cm)	Bias std (cm)	Precision (cm)	Precision std (cm)
Evergreen Needleleaf	-10.2	0.48	60.8	2.87
Mixed forest	-4.9	0.12	48.1	1.15
Woody savannas	1.7	0.13	83.7	6.09
Grasslands	-14.8	0.21	40.2	0.58
Permanent wetlands	-4.1	0.28	55.9	3.77
Croplands	-3.5	0.04	40.5	0.56
Urban and built-up	0.8	0.01	69.8	1.07
Cropland/Natural vegetation	-2.2	0.03	34.7	0.40

Table 5: Quality per Land Cover

There are some differences per land cover. Evergreen Needleaf and Grasslands have a higher than average bias. This seems to be caused by the larger negative tail, which slightly shifts the average. This larger negative tail is present in almost all histograms, not as pronounced in every histogram. It does not seem to be caused by a specific landcover, because it is not particular in 1 land cover as the error in the water land cover was. The bias of Woody savannas and Urban is smaller than on average. They seem to have more pronounced tails on both sides, which gives a bad precision, but also a good bias.

5.4 Quality per Pass

The ICESat-2 data is given per pass over an orbital segment. The height error histogram can be made per pass to see if there are irregularties in specific passes and the bias/precision can be determined.

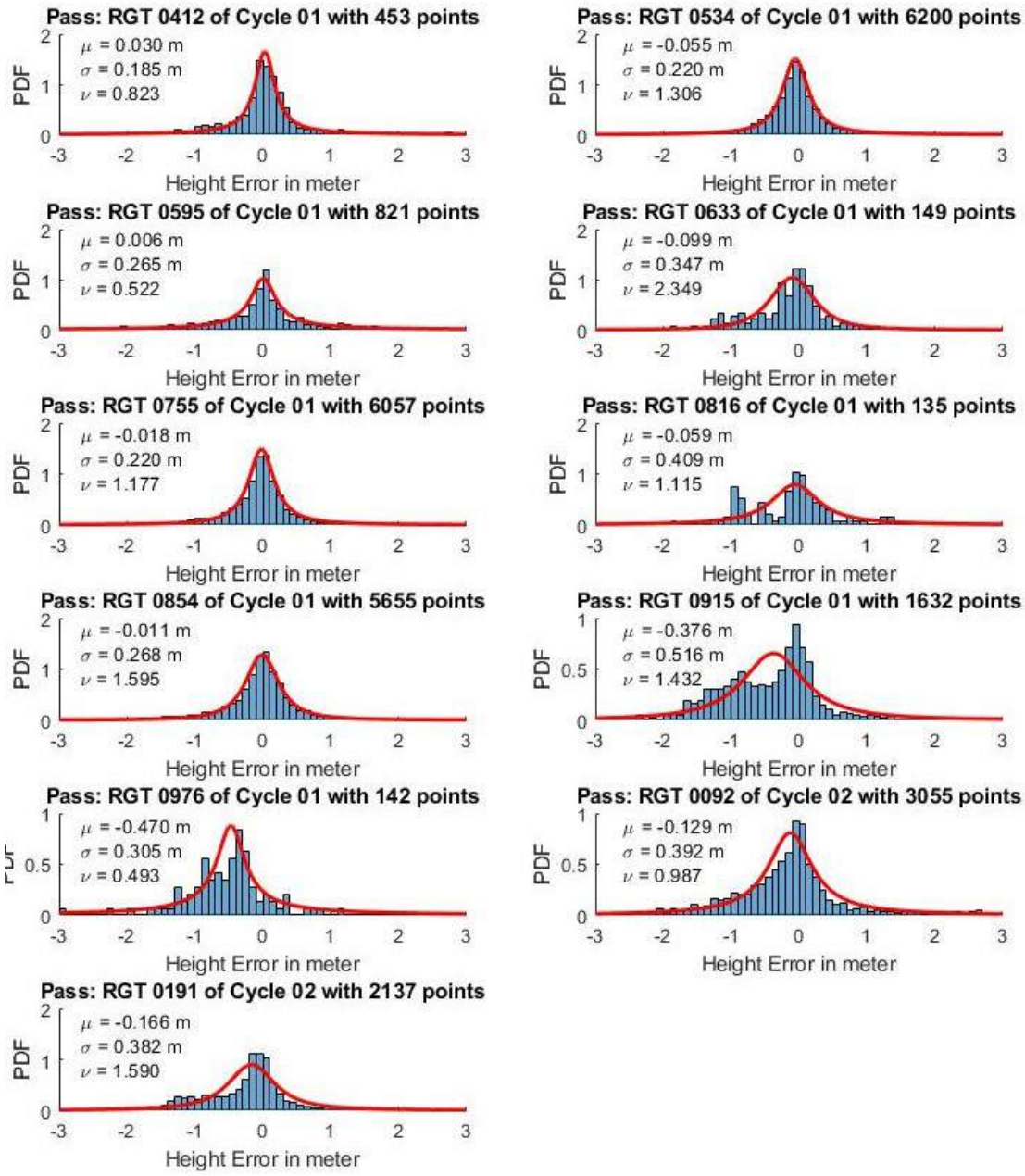


Figure 22: Height error pdf of all passes with data

Among all the histograms there are irregularities. Passes of RGT 0633 cycle 01, RGT 0816 cycle 01 and RGT0976 of cycle 1 have irregularities especially RGT 0816 cycle 01 at -1 meter. However, these three passes have not more than 150 data points which makes it harder to really get a sound conclusion.

Most histograms also have the large negative tail irregularity as seen in the land type histogram and the histogram of all data. Especially RGT 0915 cycle 01, which even seems to have an extra peak around -0.8 meter. The other histograms with larger negative tails probably also have a small peak at -0.8 meter, but they are much pronounced as peak, more as a general rise in the negative tail. Pass RGT 0915 cycle 01, RGT 0976 cycle 01 and RGT 0191 cycle 02 are inspected further, because they have the biggest larger negative tail. The positions of the the measurements are all over the Netherlands. There is no particular place where this error occurs. The grasslands land type is a little more frequent, but not too different form the other passes. There seems to be no other differentiating factor that stands out at, which can cause the peak at -1 height error. The effects of the larger tails skew the t-distribution fit and also greatly affect the bias and precision as seen the table below.

Pass	Bias (cm)	Precision (cm)	Precision std (cm)
RGT 0412 Cycle 01	3.0	39.0	0.09
RGT 0534 Cycle 01	-5.5	34.5	0.01
RGT 0595 Cycle 01	0.6	98.6	0.12
RGT 0633 Cycle 01	-9.9	43.1	0.29
RGT 0755 Cycle 01	-1.8	36.2	0.01
RGT 0816 Cycle 01	-5.9	69.1	51.15
RGT 0854 Cycle 01	-1.1	37.9	0.51
RGT 0915 Cycle 01	-37.6	76.4	0.01
RGT 0976 Cycle 01	-47.0	124.0	0.87
RGT 0092 Cycle 02	-12.9	72.1	0.02
RGT 0191 Cycle 02	-16.6	54.4	0.03

In the passes where the negative tail is greatly increased and the passes with a small amount of data points, the bias and precision are bad. The passes, which do not fall in either of those groups, have a bias and precision better or similar to the bias and precision of all data points. in those passes the quality is satisfactory. The bias standard deviation is not calculated because there are enough points in these passes to make the standard deviation very small (< 0.01 cm).

5.5 Quality per Strong/Weak beam

The ICESat-2 uses 2 beam strenghts. The strong beam uses four times more energy than the weak beam. This should result in more data points from the strong beam and more data points should make a more accurate height. The amount of data points are:

Strong Data Points	15933	60.1 % of Points
Weak Data Points	10558	39.9 % of Points

The strong beam has about 1.5 times the data points than the weak beam. It should be noted that that does not mean it has 1.5 times more data. ATL08 works with segments and if there are enough signal points in a segment it can deduct heights. 1 data point is 1 segment with data so a data point is not directly data of photons, however it is related. And there is a clear difference, the strong beam has more data points. To check this, the height error histogram is made of separately strong beam and weak beam data points as seen in figure 23.

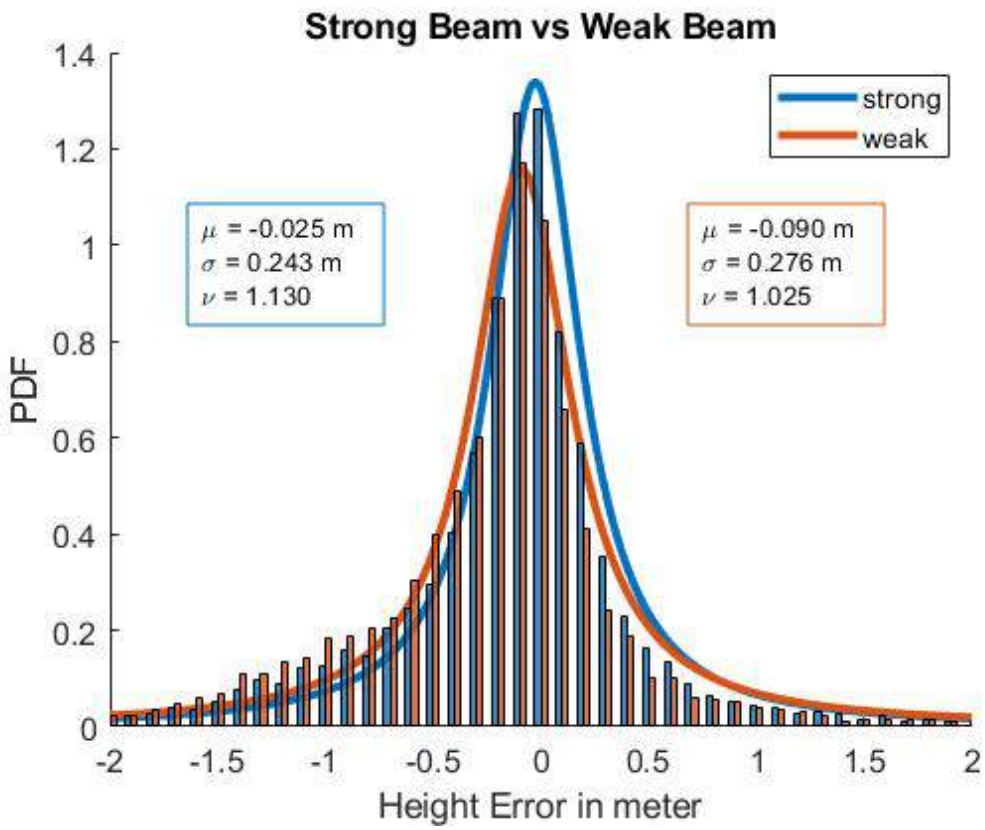


Figure 23

The weak beam has a little more bias than the strong and the tails differ slightly. On the negative side the tail of the weak beam is slightly higher while on the positive side the tail of the strong beam is slightly higher. The precision of the stong beam is 40.5 cm and the precision of the weak beam is 49.0 cm. The weak beam precision is a little worse than the precision using all the data points and

the strong beam precision is a little better than the precision using all the data point. Both beams give good data, the strong beam is a little better as expected.

5.6 Quality per Cloud confidence

The ICESat-2 data provides a cloud flag. This is not a 0/1 clouds or no clouds flag, it is a flag with 6 tiers. These tiers go from 0 - high confidence in clear skies to 5 - high confidence in clouded skies. All labels are displayed below.

Tier	**Label**	**Data Points**	**% of Points**
0	High Confidence in clear skies	17261	65.2 % of Points
1	Medium Confidence in clear skies	3145	11.9 % of Points
2	Low Confidence in clear skies	2215	8.4 % of Points
3	Low Confidence in clouded skies	1685	6.4 % of Points
4	Medium Confidence in clouded skies	1043	3.9 % of Points
5	High Confidence in clouded skies	1142	4.3 % of Points

There is a problem with the cloud analysis in ATL08. If there are not enough signal points, the data point is rejected. A cloud massively decreases the amount of signal points and thus there is a high chance of the data being rejected and that results in not too many clouded data to interpret from. All data points are classified in one of these tiers. The height error histograms are made of the data points devided in the tiers and from this the bias and precision is calculated.

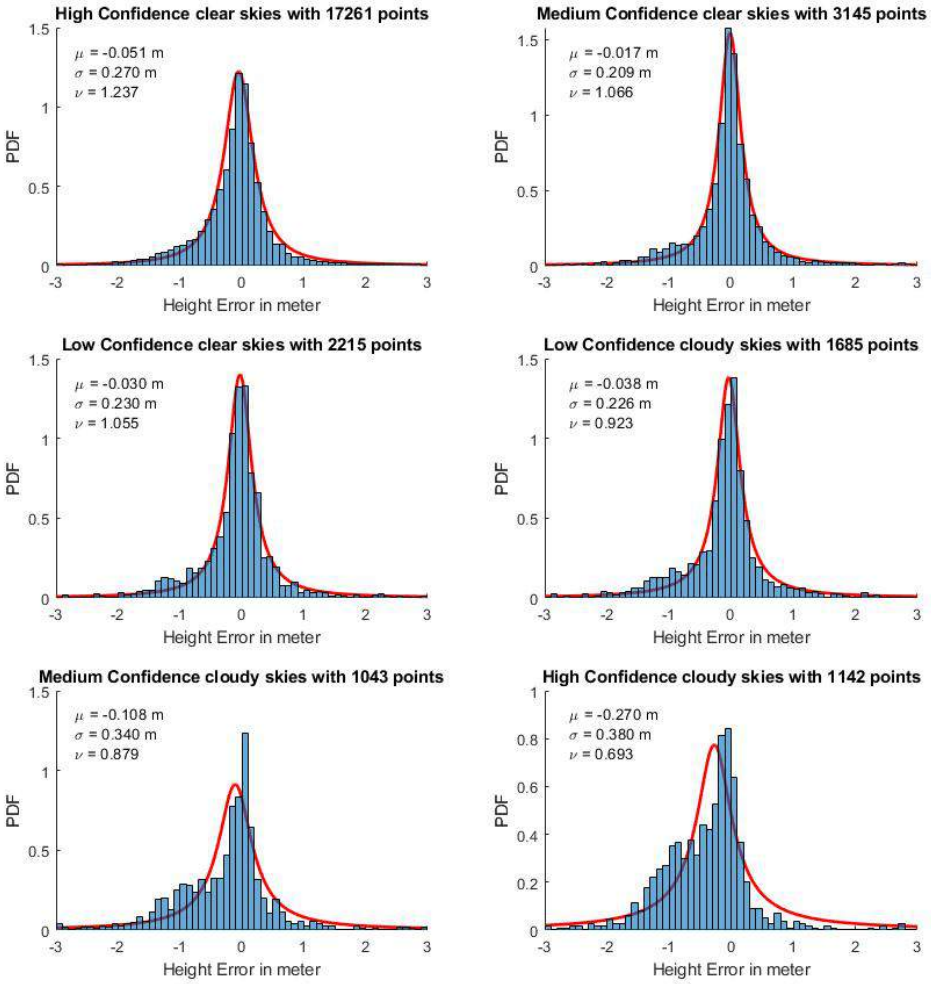


Figure 24: Heigh Error Histograms of different Cloud flag tiers

The first four tiers have similar histograms with a relatively small bias and narrow peak, which would indicate good precision. Again, the negative tail does slightly increase in the first four tiers and in the last 2 tiers the negative tail grows very much and so also has much effect on the fitted distribution. This results in the bias/precision listings in table 7.

Tier	Bias in cm	Bias std in cm	Precision in cm	Precision std in cm
0	-5.1	0.04	42.9	0.32
1	-1.7	0.03	36.3	0.65
2	-3.0	0.06	40.0	0.85
3	-3.8	0.09	43.2	1.05
4	-10.8	0.33	68.2	2.11
5	-27.0	0.80	96.0	2.84

Table 6: Quality per Cloud flag tier

The quality of the first four tiers is good. In both bias and precision they are better than the overall bias precision of all points. The fourth and fifth tier decrease very much in quality, logically this is because of clouds. The tiers represent confidence in the presence of clouds. From the quality it could be assumed that in tier 4 and 5 there are clouds present, which affect the measurements negatively. In tier 0 to 3 there are no clouds or not enough clouds present to affect the measurements negatively. It may be better to neglect the tier 4 and 5 data to improve the overall data quality.

5.7 Quality per Photon Count

ATL08 uses segments of 100 meter, within which the photons are filtered and many different variables are determined and saved in a data point. The more photons in a segment, the better the variables will be determined. That is, if the increase in photons are not only noise. ATL08 gives the number of photons in a segment. This is signal photons and noise. However, with more photons overall the signal photons should also be more. The quality of the heights should increase with more points. To test this multiple height error histograms are made. The data points are divided in four groups data points with 50-100, 100-200, 200-300, and more than 300. These ranges are chosen so every group has enough data points to make good histograms. Also the data point is rejected if there are less than 50 photons so there is no data for a photon count below 50. From the histograms the bias and precision is calculated

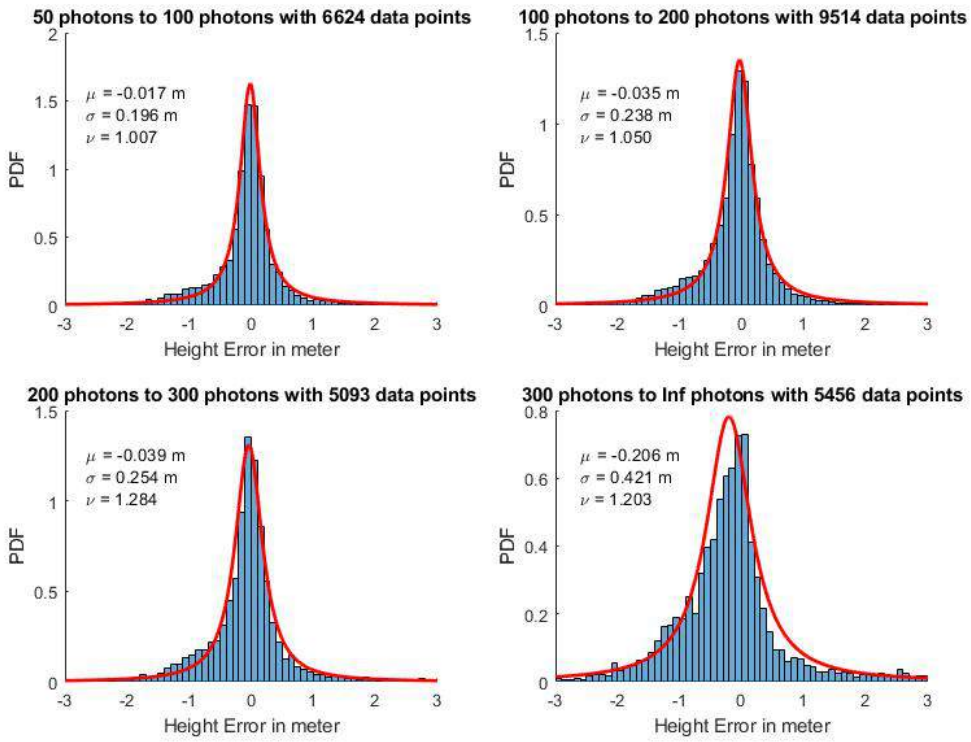


Figure 25: Height Error Histograms of with different photon count

The histograms show yet again a higher negative tail. In the first 3 ranges this is a small increase, however in the range 'More than 300' photons the entire left side of the peak seem to dip slower. This seems to be more than the slightly increased negative tail as seen before. This will make the bias and precision much worse.

Photon Count	**Bias in cm**	**Bias std in cm**	**Precision in cm**	**Precision std in cm**
50 - 100	-1.7	0.02	35.3	0.43
100 - 200	-3.5	0.04	41.5	0.43
200 - 300	-3.9	0.05	40.1	0.56
More than 300	-20.6	0.28	68.4	0.92

Table 7: Quality per Cloud flag tier

The biases and precisions do show something different than expected. A higher photon count results in a worse quality. Between 50-100 photons the quality is good the bias is less than half of the bias when taking all points and the precision

is better too. The 100-200 and 200-300 are about as good as taking all points, but worse than just the data points with 50-100 photons. The data points with more than 300 photon is very bad. The bias goes to 5 times the bias of taking all points and the precision gets almost 2 times worse. This is not good and it would seem that a lots of photons mainly means lots of noise and not more good data for a good interpretation.

5.8 Validation

The precision is not very good and this makes it bad for looking at a small scale at the data. AHN data is much better at a large scale. However, the ICESat-2 is not made for small scale research, it is made to measure on a global scale and for that purpose the precision is enough. The bias is also not perfect, but as with the precision it is good for a global scale measurements. Also in working with the data no big errors/faults are found, the only anomaly is the slightly larger negative tail, however this is not enough to not make the ICESat-2 data over the Netherlands invalid.

6 Conclusion

From all the ground tracks coverage is can be seen. With all 8 repeat paths a relatively dense diamond pattern grid of measurements is formed over the Netherlands resulting in a good coverage. However, the planning of when repeats are done is not known and repeats will have to be done multiple times. As the first three cycles all do repeat one. This means the temporal frequency of measurements and the time after which an dense grid is formed is not known.

The quality of the ICESat-2 heights is quantified with bias and precision. The bias and precision are determined by making the histogram of the difference between ICESat-2 height and AHN2 height and fitting a t-distribution to it. From the distribution the average is the bias and the range from the average where 68% of points are is the precision. After filtering the bad data a bias of -4.8 cm and precision of 43.5 cm is calculated. This is enough accuracy for the scientific goal ICESat-2 has for a 3 meter precise vegetation height. From the histogram there can be seen a asymmetry in which the tail at the negative side is larger for unknown reason.

Looking at the quality of different groupings of data. Most of the land cover have a similar or better bias/precision then the total data. With some land covers having a worse than average bias or precision mostly due to the tails being larger. In data per pass there are some big differences. The larger negative tails are much more pronounced in some pass histograms than others especially RGT 0915 of cycle 01. However there seems no particular aspect of this pass, which would make it more prone to lower heights. The difference between data of the beam strengths. Is mainly the amount of data the strong beam has 1.5 times more data points than the weak beam, which is related to the amount of photons captured. The bias and precision of the weak beam is slightly worse than the strong beam, which is expected with less data. The from tier 0 to 3 in cloud confidence the data quality is still good and slightly better to the average of all points. The data from tier 4 and 5 worsen very much. Both bias and precision in tier 4 and 5 are so much worse it may be better to ignore these points. The more photons captured in a data point does not mean a better quality of data. The data worsens the more photons captured. This is against expectations, and it seems that more photons means more noise photons and not more signal photons.

With a quality good enough for large scale research (as the ICESat-2 mission is) and no big errors in the data the ICESat-2 data in the Netherlands is valid.

7 Recommendations

For improvement of this research or suggestions for further research, there are some recommendations. In some time redo a validation/quality check of ICESat-2, because:

- In this research the very first released data is used. No revisions or improved version were yet made, so it is likely the quality of ICESat-2 will get better.
- With some time more data is available to make better results. In some histograms there are not really enough points to make a very strong conclusions.
- More repeats will be done and so more variability in data positions will be available. In this research cycle 1 and part of cycle 2 are used, these cycle both use repeat 1 having also another repeat in the data would be good.

The vegetation height quality also heavily influences the land carbon inventory calculation, which ICESat-2 wants to do. The quality of the vegetation height should then also be checked.
Also it would be better to use ATL03 instead of ATL08 to determine the quality of ICESat-2. In this research there was not enough time to filter the noise from ATL03 and not enough computing power to work with ATL03. However, it is less processed data form ICESat-2 and would give a better view of the quality of ICESat-2.

References

AHN. (2018). *De details van het Actueel Hoogtebestand Nederland.* Retrieved 7-7-2019, from http://ahn.maps.arcgis.com/apps/Cascade/index.html?appid=75245be5e0384d47856d2b912fc1b7ed

Birkett, C. (2011). *The ICESat-2 Mission: Laser altimetry of ice, clouds and land elevation.* Retrieved 7-7-2019, from https://www.aviso.altimetry.fr/fileadmin/documents/OSTST/2011/oral/03_Friday/Plenary/New%20Frontiers/07%20Birkett.ICESat2.pdf

ESA. (2019a). *ICESat-2 (Ice, Cloud and land Elevation Satellite-2).* Retrieved 7-7-2019, from https://directory.eoportal.org/web/eoportal/satellite-missions/i/icesat-2

ESA. (2019b). *ICESat (Ice, Cloud and land Elevation Satellite).* Retrieved 7-7-2019, from https://directory.eoportal.org/web/eoportal/satellite-missions/i/icesat

Forfinski-Sarkozi, N. A., & Parrish, C. E. (2016). Analysis of MABEL Bathymetry in Keweenaw Bay and Implications for ICESat-2 ATLAS. *Remote Sensing*, *8*(9). Retrieved from http://www.mdpi.com/2072-4292/8/9/772 doi: 10.3390/rs8090772

IPCC. (2014). *Climate change 2014: synthesis report. Contribution of Working Groups I, II and III to the fifth assessment report of the Intergovernmental Panel on Climate Change.* IPCC, Geneve, Switzerland.

Markus, T., Neumann, T., Martino, A., Abdalati, W., Brunt, K., Csatho, B., ... others (2017). The Ice, Cloud, and land Elevation Satellite-2 (ICESat-2): science requirements, concept, and implementation. *Remote sensing of environment*, *190*, 260–273.

McGill, M., Markus, T., Scott, V. S., & Neumann, T. (2013). The multiple altimeter beam experimental Lidar (MABEL): An airborne simulator for the ICESat-2 mission. *Journal of Atmospheric and Oceanic Technology*, *30*(2), 345–352.

NASA. (2018). *Icesat-2: Technical specs.* Retrieved 17-5-2019, from https://icesat-2.gsfc.nasa.gov/science/specs

Neuenschwander, A., & Pitts, K. (2019). The ATL08 land and vegetation product for the ICESat-2 Mission. *Remote sensing of environment*, *221*, 247–259.

Neuenschwander, A. L., & Magruder, L. A. (2016). The Potential Impact of Vertical Sampling Uncertainty on ICESat-2/ATLAS Terrain and Canopy Height Retrievals for Multiple Ecosystems. *Remote Sensing*, *8*(12). Retrieved from http://www.mdpi.com/2072-4292/8/12/1039 doi: 10.3390/rs8121039

PDOK. (2019). *Datasets.* Retrieved 23-5-2019, from https://www.pdok.nl/

Rijkswaterstaat. (2018). *Normaal Amsterdams Peil (NAP).* Retrieved 7-7-2019, from https://www.rijkswaterstaat.nl/zakelijk/open-data/normaal-amsterdams-peil/index.aspx

Slegers, J. (2012). *kml2struct.* Retrieved 7-7-2019, from https://nl.mathworks.com/matlabcentral/fileexchange/35642-kml2struct

Space Launch Report. (2018). *Delta II Data Sheet.* Retrieved 16-5-2019, from https://www.spacelaunchreport.com/delta2.html#log

A Scripts

A.1 timedRGTextractor

```
%% timedRGTextractor
clear all;clc;
% Script for extracting time specific RGT data over a target area
% folder for saving .kml and .shp
folder = 'I:/BEP/Glacier/cycle4'
% Locate and read composited time specific RGT .kml
filename = "I:\BEP\RGT\cycle4\doc.kml";
bigkml = kml2struct(filename)
%% Separate composite kml into struct per RGT
%setup variables
count=1;
border = 1;
kml = struct('RGT',[]);
for i = 2:length(bigkml)
    if strcmp(bigkml(i).Geometry,'Line') == true  %check if new RGT starts
        count = count+1;
        border(count)=i; %get border index value
        % Seperate kml
        kml(count-1).RGT = bigkml(border(count-1):(border(count)-2));
    else
    end
end

%% Determine Target center and margin
target = [8.053078, 46.488703] %[5.389776 52.156250]  %[lon lat] in WGS84 ...
    center point of the target area
margin = [0.5 0.5];            %[lon lat] in WGS84 margin of target area from ...
    center point
%get targetbox coord.
lon = [target(1)-margin(1) target(1)+margin(1)];
lat = [target(2)-margin(2) target(2)+margin(2)];

%% Setup struct to for .shp of RGT over target
fieldGeo = 'Geometry';
fieldName = 'Name';
fieldDesc = 'Description';
fieldLat = 'Lat';
fieldLon = 'Lon';
S = struct(fieldGeo, [], fieldName, [], fieldDesc, [], fieldLat, [], fieldLon, []);

%% Get RGT over Target with description
nfiles = length(kml); %amount of RGTs
nRGT=0;               %setup counter
```

```
for i = 1:nfiles
    line(i) = kml(i).RGT(1);  %get RGT line
    % search the line for coordinates in interested area
    LonIn = find(lon(1)<line(i).Lon & (line(i).Lon)<lon(2));
    LatIn = find(lat(1)<line(i).Lat & (line(i).Lat)<lat(2));
    Common = intersect(LonIn,LatIn);
    Check = isempty(Common);
    % save RGT if it crosses interested area
    if Check == 1
    else
        nRGT=nRGT+1;        %Number of RGT found
        fprintf('RGTs found: %d\n', nRGT) %process indicator
        for j = 2:length(kml(i).RGT)            %get distance to point for time
            PointLoc = [kml(i).RGT(j).Lon kml(i).RGT(j).Lat];
            dist(j) = sqrt(((target(1)-PointLoc(1))^2) + ...
                ((target(2)-PointLoc(2))^2));
        end
        minDist = mink(dist,4);                   %get closest point
        PointInd = find(ismember(dist,minDist)); %get index closest point
        %get time of point
        Desc1 = kml(i).RGT(PointInd(2)).Description;
        Desc2 = kml(i).RGT(PointInd(4)).Description;
        %get point coord.
        Point1 = [kml(i).RGT(PointInd(2)).Lon kml(i).RGT(PointInd(2)).Lat];
        Point2 = [kml(i).RGT(PointInd(4)).Lon kml(i).RGT(PointInd(4)).Lat];
        %find segment in between point
        SegmInd1 = find(line(i).Lon == Point1(1) & line(i).Lat == Point1(2));
        SegmInd2 = find(line(i).Lon == Point2(1) & line(i).Lat == Point2(2));
        Segm = [SegmInd1 SegmInd2];
        % extract segment out of total RGT line
        if SegmInd1≤SegmInd2
            RGTLat = line(i).Lat(min(Segm):1:max(Segm));
            RGTLon = line(i).Lon(min(Segm):1:max(Segm));
        elseif SegmInd1≥SegmInd2
            RGTLat = line(i).Lat(max(Segm):-1:min(Segm));
            RGTLon = line(i).Lon(max(Segm):-1:min(Segm));
        end
        % fill in struct for shaperead
        S(nRGT).Geometry = 'Line';
        S(nRGT).Name = num2str(i);
        S(nRGT).Description = [Desc1 Desc2];
        S(nRGT).Lat = RGTLat;
        S(nRGT).Lon = RGTLon;
        filename = sprintf(join([folder 'TargetRGT/%s_%d']),'RGT_',i);    ...
            %create new kml filename
        kmlwriteline(filename, RGTLat, RGTLon, 'Description', [Desc1 Desc2]);   ...
            %save as kml
    end
    if mod(i,10) == 0         %progress indicator
```

```
        fprintf('%.2f %% \n',(i*100/nfiles))
    elseif i == nfiles
        fprintf('%.2f %% \n',(i*100/nfiles))
    end
end

%% Saving Shapefile
shapefile = join([folder 'TargetRGT/CycleRGTs']);
shapewrite(S,shapefile)
```

A.2 wgetATLAScommand

```
%% wgetATLAScommand
clear all; close all; clc;
% Script for creating the wget command to retrieve the HDF5 files from the
% cut specific orbit data from QGIS
% This script only works for data between 27 degrees N and 59,5 degrees N

%% Needed variables
EarthDataUser = input('EarthData username: ', 's');
EarthDataPassword = input('EarthData Password: ', 's');
DataProduct = input('Data product number in 2 digits: ', 's');
Version = input('Version number in 3 digits: ', 's');
Revision = input('Revision number in 2 digits: ', 's');

%% read shapefile from QGIS with time specific RGT data
filename = 'I:\BEP\QGIS SHP\Cycle1_NL\cycle1_NL.shp';
cycle = shaperead(filename);
%% start loop through the RGT
for i = 1:length(cycle)
    % Get description with the date, time and RGT number
    desc = cycle(i).Descriptio;
    % Locate and extract date
    dateloc = find(desc=='-');
    date = desc(dateloc-2:dateloc(2)+4);
    % Locate and extract time
    timeloc = find(desc==':');
    time = desc(timeloc-2:timeloc(2)+2);
    % Combine date and time and convert to datetime variable
    dtstring{i,1} = join([date ' ' time]);
    dt(i) = datetime(dtstring{i,1});
    dtstring{i,2} = desc(1:8);
    % Setup earthdata url with date for wget
    ATLfoldername = join(['https://n5eil01u.ecs.nsidc.org/ATLAS/ATL' ...
        DataProduct '.' Version '/' datestr(datenum(dt(i)), 'yyyy.mm.dd')]);

```

```
    % Make RGT number a 4 digit string
    rgtlen = length(cycle(i).Name);
    if rgtlen == 1
        rgtstr = join(['0' '0' '0' cycle(i).Name]);
    elseif rgtlen == 2
        rgtstr = join(['0' '0' cycle(i).Name]);
    elseif rgtlen == 3
        rgtstr = join(['0' cycle(i).Name]);
    else
    end

    % Determine cycle number from date
    if dt(i) ≥ datetime(2018,12,29)
        nrcycle = '02';
    else
        nrcycle = '01';
    end
    % Determine orbital segment number
    % 06 for ascending, 02 for descending
    direc = cycle(i).Y(1)-cycle(i).Y(2);
    if direc ≤ 0
        direcstr = '02';
    else
        direcstr = '06';
    end
    % Setup HDF5 filename search criteria for wget
    ATLfilename = join(['*_' rgtstr nrcycle direcstr '_' Version '_' Revision ...
        '.h5']);
    % Setup wget command for 1 RGT
    OneCommand{i} = join(['wget --http-user=' EarthDataUser ' --http-password=' ...
        EarthDataPassword ' --load-cookies mycookies.txt --save-cookies ...
        mycookies.txt --keep-session-cookies -r --accept "' ATLfilename '" -np ...
        -e robots=off ' ATLfoldername '/']);
end
%% Combine the wget commands to 1 command
TotCommand = join(OneCommand, ' & ');
% Print total wget command
fprintf(TotCommand{1})
```

A.3 wget Command

```
wget --http-user=username --http-password=password --load-cookies mycookies.txt ...
    --save-cookies mycookies.txt --keep-session-cookies -r --accept ...
    "*_00310202_001_01.h5" -np -e robots=off ...
    https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.30/ & wget ...
    --http-user=username --http-password=password --load-cookies mycookies.txt ...
```

```
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_00920202_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2019.01.03/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_01530202_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2019.01.07/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_01910206_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2019.01.09/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_02520106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.10.15/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_03130106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.10.19/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_03510102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.10.21/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_03740106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.10.23/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_04120102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.10.25/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_04730102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.10.29/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_05340102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.02/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_05950102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.06/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_06330106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.09/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
```

```
"*_06560102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.10/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_06940106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.13/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_07550106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.17/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_08160106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.21/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_08540102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.23/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_08770106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.25/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09150102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.11.27/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09760102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.01/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09760102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.05/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09760106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.08/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09760102_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.09/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09760106_001_01.h5" -np -e robots=off ...
https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.12/ & wget ...
--http-user=username --http-password=password --load-cookies mycookies.txt ...
--save-cookies mycookies.txt --keep-session-cookies -r --accept ...
"*_09760106_001_01.h5" -np -e robots=off ...
```

```
    https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.16/ & wget ...
    --http-user=username --http-password=password --load-cookies mycookies.txt ...
    --save-cookies mycookies.txt --keep-session-cookies -r --accept ...
    "*_09760106_001_01.h5" -np -e robots=off ...
    https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.20/ & wget ...
    --http-user=username --http-password=password --load-cookies mycookies.txt ...
    --save-cookies mycookies.txt --keep-session-cookies -r --accept ...
    "*_09760102_001_01.h5" -np -e robots=off ...
    https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.22/ & wget ...
    --http-user=username --http-password=password --load-cookies mycookies.txt ...
    --save-cookies mycookies.txt --keep-session-cookies -r --accept ...
    "*_09760106_001_01.h5" -np -e robots=off ...
    https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.24/ & wget ...
    --http-user=username --http-password=password --load-cookies mycookies.txt ...
    --save-cookies mycookies.txt --keep-session-cookies -r --accept ...
    "*_09760102_001_01.h5" -np -e robots=off ...
    https://n5eil01u.ecs.nsidc.org/ATLAS/ATL08.001/2018.12.26/>>
```

A.4 ATL08import

```
function [values] = ATL08import(filename, beam, name)
% This function reads and ATLAS HDF5 file and returns datagroup in a matlab as ...
    variable

% Input
%   filename is the name of HDF5 file
%   beam is the chosen beam [1l, 1r, 2l, 2r, 3l, 3r] from which the data is ...
    retrieved
%   name is the title of the data group/flag

% Output
%   values = data from chosen beam/group

% Setup directory of group in HDF5 struct with data
group = join(['/gt' beam '/land_segments/' name]);
values = h5read(filename, group);
% Substitute infinte values for NaN
values(find(values≥10^10))=NaN;
end
```

A.5 Datastructer

```
%% Data structer
```

```
clear all; close all; clc;
tic
%% setup files and directory
basefolder = 'I:\BEP\n5eil01u.ecs.nsidc.org\ATLAS\ATL08.001';
% get list of files and folders in any subfolder
filelist = dir(fullfile(basefolder, '**\*.*'));
% remove folders
filelist = filelist(¬[filelist.isdir]);
% remove empty files
for i = 1:length(filelist)
    html(i) = filelist(i).bytes == 0;
end
filelist(html) = [];

%% setup variables for storing data
beam = {'1l', '1r', '2l', '2r', '3l', '3r'};
border = shaperead('I:\BEP\QGIS SHP\Borders\DutchBorderWGS84.shp');
Data = struct('file',[],'beam', []);
Diff_all = [];

%% begin loop over file
for i = 1:length(filelist)
    filename = join([filelist(i).folder '\' filelist(i).name]);
    Data(i).file = filelist(i).name;
    Data(i).beam = struct('track',[], 'lat', [], 'lon', [], 'Δ_dist', [], ...
        'cs_dist', [], 'terrainMean', [], 'DiffMean', [], 'terrainInterp', [], ...
        'DiffInterp', [], 'terrainBestfit', [], 'DiffBestfit', [], ...
        'AHN_terrain', [], 'npho', [], 'night', [], 'cloud', [], 'landcover', ...
        [], 'slope', [], 'uncertainty', [], 'roughness', []);
    %% Loop over beams
    for j = 1:6
        fprintf('File %d/%d beam %d/6 \n', i, length(filelist), j);
        Data(i).beam(j).track = beam(j);

        lat = ATL08import(filename, char(beam(j)), 'latitude');
        lon = ATL08import(filename, char(beam(j)), 'longitude');
        [in, on] = inpolygon(lon,lat,border.X,border.Y);
        lat_in = lat(in);
        lon_in = lon(in);
        Data(i).beam(j).lat = lat_in;
        Data(i).beam(j).lon = lon_in;

        npho = ATL08import(filename, char(beam(j)), 'n_seg_ph');
        npho_in = npho(in);
        Data(i).beam(j).npho = npho_in;
        night = ATL08import(filename, char(beam(j)), 'night_flag');
        night_in = night(in);
        Data(i).beam(j).night = night_in;
        cloud = ATL08import(filename, char(beam(j)), 'cloud_flag_asr');
```

```
        cloud_in = cloud(in);
        Data(i).beam(j).cloud = cloud_in;
        landcover = ATL08import(filename, char(beam(j)), 'segment_landcover');
        landcover_in = landcover(in);
        Data(i).beam(j).landcover = landcover_in;
        solarel = ATL08import(filename, char(beam(j)), 'solar_elevation');
        solarel_in = solarel(in);
        Data(i).beam(j).solarel = solarel_in;
        solaraz = ATL08import(filename, char(beam(j)), 'solar_azimuth');
        solaraz_in = solaraz(in);
        Data(i).beam(j).solaraz = solaraz_in;
        slope = ATL08import(filename, char(beam(j)), 'terrain/terrain_slope');
        slope_in = slope(in);
        Data(i).beam(j).slope = slope_in;
        uncer = ATL08import(filename, char(beam(j)), 'terrain/h_te_uncertainty');
        uncer_in = uncer(in);
        Data(i).beam(j).uncertainty = uncer_in;
        rough = ATL08import(filename, char(beam(j)), 'terrain/h_te_std');
        rough_in = rough(in);
        Data(i).beam(j).roughness = rough_in;

        for k = 2:length(lat_in)
            Data(i).beam(j).Δ_dist(k) = vdist(lat_in(k-1), lon_in(k-1), ...
                lat_in(k), lon_in(k));
        end
        Data(i).beam(j).cs_dist = cumsum(Data(i).beam(j).Δ_dist);
        for k = 1:length(lat_in)
            latlon = [double(lat_in(k)) double(lon_in(k))];
            rd=etrs2rd(latlon);
            ahnunit=ahnfind(rd);
            if isempty(ahnunit) == 1
                AHNterrain = NaN
            else
                try
                    ahnfile=ahndownload(ahnunit,'ahn2_5m');
                    ahn=ahnread(ahnfile,rd,1);
                catch
                    ahn.a = NaN;
                    continue
                end
                if size(ahn.a) == [1 1]
                    AHNterrain=ahn.a;
                else
                    AHNterrain=ahn.a(2,2);
                end
            end
            Data(i).beam(j).AHN_terrain(k) = AHNterrain;
        end

```

```
        terrainMean = ATL08import(filename,char(beam(j)),'terrain/h_te_mean');
        terrainInterp = ATL08import(filename,char(beam(j)),'terrain/h_te_interp');
        terrainBestfit = ...
            ATL08import(filename,char(beam(j)),'terrain/h_te_best_fit');
        terrainMean_in = terrainMean(in);
        terrainInterp_in = terrainInterp(in);
        terrainBestfit_in = terrainBestfit(in);

        NAPterrainMean = etrs2nap(lat_in, lon_in, terrainMean_in);
        NAPterrainInterp = etrs2nap(lat_in, lon_in, terrainInterp_in);
        NAPterrainBestfit = etrs2nap(lat_in, lon_in, terrainBestfit_in);
        Data(i).beam(j).terrainMean = NAPterrainMean';
        Data(i).beam(j).terrainInterp = NAPterrainInterp';
        Data(i).beam(j).terrainBestfit = NAPterrainBestfit';

        Data(i).beam(j).DiffMean = ...
            Data(i).beam(j).terrainMean(1:length(Data(i).beam(j).AHN_terrain)) - ...
            Data(i).beam(j).AHN_terrain;
        Data(i).beam(j).DiffInterp = ...
            Data(i).beam(j).terrainInterp(1:length(Data(i).beam(j).AHN_terrain)) ...
            - Data(i).beam(j).AHN_terrain;
        Data(i).beam(j).DiffBestfit = ...
            Data(i).beam(j).terrainBestfit(1:length(Data(i).beam(j).AHN_terrain)) ...
            - Data(i).beam(j).AHN_terrain;
        clear strdist
    end
end
toc
```

A.6 Precision

```
function prec = precision(percentage, pd, XGrid)
% This function calcutes the range from the average between which the
% integral of the pdf is equal to the given percentage from a fitted
% disttribution

% Input
%   percentage is the integral of pdf in range
%   pd is the fitted distribution
%   XGrid is the vector with the x-values

% Output
%   prec is the precision

ref = (1-(percentage/100))/2; % percentage to ratio
YPlot = cdf(pd, XGrid); % create cdf
```

```
Dif = abs(YPlot - ref); % difference to ratio
[val, ind] = min(Dif); % find index closest cdf point to ratio
prec = abs(pd.mu-XGrid(ind)); % get x value convert to precision
end
```

A.7 Processor

```
%% Processor
close all; clear all; clc;
% This script processes the data struct made by datastructer of ICESat-2
% ATL08 data
% Load Data struct with imported data
temp = load('I:\BEP\DataExp2.mat');
Data = temp.Data;
% Turn on/off processes
cross =         0;
mappoint =      0;
ExecSHP =       0;
ExecAll =       0;
ExecBeam =      0;
ExecPass =      1;
ExecCloud =     0;
ExecPho =       0;
ExecNight =     0;
ExecLandcover = 0;

%% Set constants/variables
limits = [-3, 3]; % limits for plots
xbin = limits(1):0.1:limits(2); % bins for histograms
perc = 68; % percentage for precision
fig = 1; % variable to make extra figure windows
%% composite all data to single vectors
% setup empty vectors
HeightDifferenceMean = [];
HeightDifferenceInterp = [];
HeightDifferenceBestfit = [];
Lat_in = [];
Lon_in = [];
clouds = [];
npho = [];
night = [];
landcover = [];
slope = [];
uncertainty = [];
roughness = [];
emptyData = 0;
```

```
for i = 1:length(Data)
    for j = 1:6
        % fill vectors
        HeightMean = NaN(1,length(Data(i).beam(j).lat));
        HeightMean(1:length(Data(i).beam(j).DiffMean)) = Data(i).beam(j).DiffMean;
        HeightDifferenceMean = [HeightDifferenceMean HeightMean];

        HeightInterp = NaN(1,length(Data(i).beam(j).lat));
        HeightInterp(1:length(Data(i).beam(j).DiffInterp)) = ...
            Data(i).beam(j).DiffInterp;
        HeightDifferenceInterp = [HeightDifferenceInterp HeightInterp];

        HeightBestfit = NaN(1,length(Data(i).beam(j).lat));
        HeightBestfit(1:length(Data(i).beam(j).DiffBestfit)) = ...
            Data(i).beam(j).DiffBestfit;
        HeightDifferenceBestfit = [HeightDifferenceBestfit HeightBestfit];

        Lat_in = [Lat_in;Data(i).beam(j).lat];
        Lon_in = [Lon_in;Data(i).beam(j).lon];
        npho = [npho;Data(i).beam(j).npho];
        night = [night;Data(i).beam(j).night];
        clouds = [clouds;Data(i).beam(j).cloud];
        landcover = [landcover;Data(i).beam(j).landcover];
        slope = [slope;Data(i).beam(j).slope];
        uncertainty = [uncertainty;Data(i).beam(j).uncertainty];
        roughness = [roughness;Data(i).beam(j).roughness];
        % count empty data points
        if isempty(Data(i).beam(j).lat)==1
            emptyData = emptyData+1;
        else
        end
        % get index water land type data points
        notwater = find(landcover≠0);
        water = find(landcover==0);

    end
end
% filter out water land type data points
HeightDifferenceInterp = HeightDifferenceInterp(notwater);
Lat_in = Lat_in(notwater);
Lon_in = Lon_in(notwater);
npho = npho(notwater);
night = night(notwater);
clouds = clouds(notwater);
landcover = landcover(notwater);
slope = slope(notwater);
uncertainty = uncertainty(notwater);
roughness = roughness(notwater);
%% make shape file for QGIS
```

```
if ExecSHP == 1
    DataPoints = struct('Geometry', 'MultiPoint', 'Name', 'DataPoint', 'Lat', ...
        double(Lat_in), 'Lon', double(Lon_in), 'dH', HeightDifferenceInterp);
    shapefile = 'I:\BEP\QGIS SHP\DataPoints.shp';
    shapewrite(DataPoints,shapefile)
else
end

%% cross-section example
if cross == 1
    % cross section index
    CSind = ...
        round(length(Data(3).beam(4).lat)*0.2):round(length(Data(3).beam(4).lat)*0.6);
    hold on
    % cross section lines
    plot(Data(3).beam(4).lat(CSind),Data(3).beam(4).terrainInterp(CSind), ...
        'LineWidth', 1.5)
    plot(Data(3).beam(4).lat(CSind),Data(3).beam(4).AHN_terrain(CSind), ...
        'LineWidth', 1.5)
    % plot specifics
    legend('AHN terrain', 'ICESat-2 terrain')
    xlabel('latitude')
    ylabel('height in NAP (m)')
    set(gca, 'FontSize', 18)
else
end

%% Height Difference
if ExecAll == 1
    figure(fig) % new figure window
    fig = fig+1;
    hold on
    HistoHeightInterp = ...
        double(HeightDifferenceInterp(¬isnan(HeightDifferenceInterp))); % remove ...
        NaN & convert to double
    [Nall,edges] = histcounts(HistoHeightInterp, 'Normalization','pdf'); %get ...
        bar heights
    edges = edges(2:end) - (edges(2)-edges(1))/2; %get middle of bars
    % fit t-distribution
    XGrid = limits(1):0.01:limits(2);
    pd1 = fitdist(HistoHeightInterp', 'tlocationscale');
    YPlot = pdf(pd1,XGrid);
    plot(XGrid, YPlot, '-r', 'LineWidth', 2); % plot fit
    histogram(HistoHeightInterp, xbin,'Normalization','pdf'); % plot histogram
    % option to fit normal distribution
    normalfit = 0;
    if normalfit == 1
        YPlot2 = pdf(pd2,XGrid);
        pd2 = fitdist(HistoHeightInterp', 'normal');
```

```
        plot(XGrid, YPlot2, '-g', 'LineWidth', 2);
        leg = {'Height difference', 't-Distribution', 'Normal Distribution'};
    else
        leg = {'Height difference', 't-Distribution'};
    end
    allprec = precision(perc, pd1, XGrid); % calculate precision
    % plot specifics
    xlim(limits);
    tbox1 = [sprintf('\\mu: %1.3f m\n', pd1.mu) sprintf('\\sigma: %1.3f m\n', ...
        pd1.sigma) sprintf('\\nu: %1.3f m', pd1.nu)];
    annotation('textbox', [0.15, 0.7, 0.2, 0.2], 'String', tbox1, ...
        'FitBoxToText', 'on')
    legend(leg, 'FontSize', 10)
    title('Height Error Histogram')
    ylabel('PDF')
    xlabel('Height Error in meter')
else
end


%% point location
if mappoint ==1
    figure(fig) % new figure window
    fig = fig +1;
    border = shaperead('I:\BEP\QGIS SHP\Borders\DutchBorderWGS84.shp'); % ...
        import shp
    border = cell2struct(struct2cell(border), {'Geometry', 'BoundingBox', ...
        'Lon', 'Lat', 'gml_id', 'Code', 'Landsnaam'}); % convert to struct
    lonlim = [3 7.5]; % lon limits of map
    latlim = [50.5 54]; % lat limits of map
    worldmap(latlim, lonlim) % setup map
    hold on
    geoshow(border, 'Facecolor', 'white'); % plot polygon border
    scatterm(Lat_in, Lon_in,[], HeightDifferenceInterp, '.')
    % Set color legend
    colormap jet
    caxis([-2 2])
    colorbar
    %geoshow(Lat_in,Lon_in, 'DisplayType', 'multipoint','Marker', '.') % plot ...
        data points
    hold off
else
end


%% n_photons
if ExecPho == 1
    figure(fig) % new figure window
    fig = fig+1;
```

```
    hold on
    % start counts
    count = 0;
    count_3 = 0;
    count_4 = 0;
    ninter = 4; % number of intervals
    phostruct = struct('from', 50,'index', [] ,'height', []); % setup struct
    for i = 1:ninter
        count = count+1; % count
        if i == ninter % if at last interval
            phostruct(i).index = find(npho ≥ phostruct(i).from); % get index
            phostruct(i+1).from = Inf;
            phostruct(i).height = HeightDifferenceInterp(phostruct(i).index); % ...
                get heights
        else
            phostruct(i+1).from = i*100; % interval start value
            phostruct(i).index = find(npho ≥ phostruct(i).from & npho ≤ ...
                phostruct(i+1).from); % get index
        end
        phostruct(i).height = HeightDifferenceInterp(phostruct(i).index); % get ...
            heights
        subplot(round(ninter/2), 2, i) % begin subplot
        % fit t-distribution
        pdpho = fitdist(double(phostruct(i).height'), 'tlocationscale');
        mus(i) = pdpho.mu;
        sigmas(i) = pdpho.sigma;
        XGrid = limits(1):0.01:limits(2);
        YPlot = pdf(pdpho,XGrid);
        hold on
        histogram(phostruct(i).height, xbin, 'Normalization', 'pdf') % plot ...
            histogram
        plot(XGrid, YPlot, '-r', 'LineWidth', 2); % plot fit
        % plot text
        titlestr = {sprintf('%d photons to %d photons with %d data points', ...
            phostruct(i).from, phostruct(i+1).from, length(phostruct(i).height))};
        title(titlestr)
        infostr = {num2str(pdpho.mu,'\\mu = %1.3f m'), num2str(pdpho.sigma, ...
            '\\sigma = %1.3f m'), num2str(pdpho.nu, '\\nu = %1.3f')};
        if mod(count,2) == 1
            count_3 = count_3+1;
            annotation('TextBox', [0.14 1-(-0.225+count_3/2.2) 0.5 0.5], ...
                'String', infostr, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                'EdgeColor', 'none')
        else
            count_4 = count_4+1;
            annotation('TextBox', [0.585 1-(-0.225+count_3/2.2) 0.5 0.5], ...
                'String', infostr, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                'EdgeColor', 'none')
        end
```

```
        phoprec(i) = precision(perc, pdpho, XGrid); % calculate precision
        % plot specifics
        xlim(limits)
        xlabel('Height Error in meter')
        ylabel('PDF')
    end
    hold off
else
end

%% night/day
if ExecNight == 1
    figure(fig) % new figure window
    fig = fig+1;
    hold on
    % get data points
    nightDif = HeightDifferenceInterp(find(night==1));
    dayDif = HeightDifferenceInterp(find(night==0));
    % get bin heights
    hcx = histcounts(nightDif,[xbin Inf], 'Normalization','pdf');
    hcy = histcounts(dayDif,[xbin Inf], 'Normalization','pdf');
    bar(xbin,[hcy;hcx]', 'BarWidth', 1.8); % plot bar
    % plot specifics
    xlim([-3 3])
    legend({'Night', 'Day'})
    title('Height Error Histogram for day/night')
    ylabel('pdf normalized frequency')
    xlabel('Height Error (m)')
    hold off
else
end

%% clouds
if ExecCloud == 1
    figure(fig) % new figure window
    fig = fig+1;
    % setup counts
    count = 0;
    count_3 = 0;
    count_4 = 0;
    cloudstruct = struct('nrflag', [], 'strflag', [], 'height', []); % setup struct
    nrflags = unique(clouds); % get options
    strflags = {'High Confidence clear skies', 'Medium Confidence clear skies', ...
        'Low Confidence clear skies', 'Low Confidence cloudy skies', 'Medium ...
        Confidence cloudy skies', 'High Confidence cloudy skies'}; % create ...
        label strings
    for i = 1:length(nrflags)
        count = count+1; % count
        % save in struct
```

```
        cloudstruct(i).strflag = strflags(i);
        cloudstruct(i).nrflag = nrflags(i);
        cloudstruct(i).height = HeightDifferenceInterp(find(clouds==i-1));
        subplot(length(unique(clouds))/2,2,i); % begin subplot
        % fit t-distribution
        pdcloud = fitdist(double(cloudstruct(i).height'), 'tlocationscale');
        mus(i) = pdcloud.mu;
        sigmas(i) = pdcloud.sigma;
        XGrid = limits(1):0.01:limits(2);
        YPlot = pdf(pdcloud,XGrid);
        hold on
        plot(XGrid, YPlot, '-r', 'LineWidth', 2); % plot fit
        histogram(cloudstruct(i).height, xbin, 'Normalization', 'pdf') % plot ...
            histogram
        cloudprec(i) = precision(perc, pdcloud, XGrid); % Calculate precision
        % plot text
        titlestr = {sprintf('%s with %d points', ...
            string(cloudstruct(i).strflag), length(cloudstruct(i).height))};
        title(titlestr)
        infostr = {num2str(pdcloud.mu,'\\mu = %1.3f m'), num2str(pdcloud.sigma, ...
            '\\sigma = %1.3f m'), num2str(pdcloud.nu, '\\nu = %1.3f')};
        if mod(count,2) == 1
            count_3 = count_3+1;
            annotation('TextBox', [0.14 1-(-0.15+count_3/3.35) 0.5 0.5], ...
                'String', infostr, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                'EdgeColor', 'none')
        else
            count_4 = count_4+1;
            annotation('TextBox', [0.585 1-(-0.15+count_4/3.35) 0.5 0.5], ...
                'String', infostr, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                'EdgeColor', 'none')
        end
        % plot specifics
        xlim(limits)
        xlabel('Height Error in meter')
        ylabel('PDF')
    end

    figure(fig) % new figure window
    fig = fig+1;
    hold on
    boxplot(HeightDifferenceInterp, clouds) % create boxplot
    % reference line at 0
    ylim([-3 3])
    y0 = refline(0);
    y0.LineWidth = 1;
    y0.Color = 'g';
    % plot specifics
    title('Boxplots of Height error per Cloud flag')
```

```
    ylabel('Height Error (m)')
    xticklabels(strflags)
    xtickangle(45)
    hold off
else
end

%% landcover
if ExecLandcover == 1
    figure(fig) % new figure window
    fig = fig+1;
    % land type names
    LCclas = {'Water';
    'Evergreen Needleleaf forest';
    'Evergreen Broadleaf forest';
    'Deciduous Needleleaf forest';
    'Deciduous Broadleaf forest';
    'Mixed forest';
    'Closed shrublands';
    'Open shrublands';
    'Woody savannas';
    'Savannas';
    'Grasslands';
    'Permanent wetlands';
    'Croplands';
    'Urban and built-up';
    'Cropland/Natural vegetation mosaic';
    'Snow and ice';
    'Barren or sparsely vegetated'};
    LCstruct = struct('name', LCclas, 'height', []); % setup struct
    count = 0; % setup count
    minlen = 100; % threshold
    for i = 1:length(LCclas)
        % get data points
       LCstruct(i).height = HeightDifferenceInterp(find(landcover == (i-1)));
       LCstruct(i).height = LCstruct(i).height(¬isnan(LCstruct(i).height));
       if length(LCstruct(i).height) ≤ minlen % index data vectors
           LCind(i) = 0;
       else
           count = count+1;
           LCind(i) = 1;
       end
    end
    % setup counts
    count_2 = 0;
    count_3 = 0;
    count_4 = 0;
    for i = 1:length(LCclas)
        if LCind(i) == 1
```

```
            count_2 = count_2+1; % count
            subplot(round(count/2),2,count_2) % begin subplot
            % fit t-distribution
            pdLC = fitdist(double(LCstruct(i).height'), 'tlocationscale');
            len(count_2) = length(LCstruct(i).height);
            mus(count_2) = pdLC.mu;
            sigmas(count_2) = pdLC.sigma;
            nus(count_2) = pdLC.nu;
            XGrid = limits(1):0.01:limits(2);
            YPlot = pdf(pdLC,XGrid);
            hold on
            plot(XGrid, YPlot, '-r', 'LineWidth', 2); % plot fit
            histogram(LCstruct(i).height, xbin, 'Normalization', 'pdf') % plot ...
                histogram
            LCprec(count_2) = precision(perc, pdLC, XGrid); % calculate precision
            % plot specifics
            xlim(limits)
            xlabel('Height Error in meter')
            ylabel('PDF')
            % plot text
            titlestr = {sprintf('%s with %d points', LCstruct(i).name, ...
                length(LCstruct(i).height))};
            title(titlestr)
            if mod(count_2,2) == 1
                count_3 = count_3+1;
                annotation('TextBox', [0.14 1-(-0.08+count_3/4.5) 0.5 0.5], ...
                    'String', {num2str(pdLC.mu,'\\mu = %1.3f m'), ...
                    num2str(pdLC.sigma, '\\sigma = %1.3f m'), num2str(pdLC.nu, ...
                    '\\nu = %1.3f')}, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                    'EdgeColor', 'none')
            else
                count_4 = count_4+1;
                annotation('TextBox', [0.585 1-(-0.08+count_4/4.5) 0.5 0.5], ...
                    'String', {num2str(pdLC.mu,'\\mu = %1.3f m'), ...
                    num2str(pdLC.sigma, '\\sigma = %1.3f m'), num2str(pdLC.nu, ...
                    '\\nu = %1.3f')}, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                    'EdgeColor', 'none')
            end
            hold off
        else
        end
    end
    figure(fig) % new figure window
    fig = fig+1;
    hold on
    % setup vectors
    LCx = [];
    LCg = [];
    for i = 1:length(LCclas)
```

```
        if length(LCstruct(i).height) ≤ minlen
        else
            % fill vectors with data
            LCx = [LCx; LCstruct(i).height'];
            LCg = [LCg; i*ones(length(LCstruct(i).height),1)];
        end
    end
    boxplot(LCx, LCg) % create boxplot
    % create reference line at 0
    y0 = refline(0);
    y0.LineWidth = 1;
    y0.Color = 'g';
    % plot specifics
    ylim(limits)
    title('Boxplots of Height error per Land Cover')
    ylabel('Height Error (m)')
    xticklabels(LCclas(find(LCind==1)))
    xtickangle(45)
    set(gca, 'FontSize', 12)
    hold off

    figure(fig) % new figure window
    fig = fig+1;
    x = 1:length(mus); % x-axis placing
    mustd = mus./sqrt(len); % mu std
    LCprecstd = LCprec./sqrt(len); % precision std
    hold on
    errorbar(x,mus,2*sigmas, '.', 'CapSize',18, 'LineWidth', 3, 'MarkerSize', ...
        1) % plot errorbar 2x sigma
    errorbar(x,mus,sigmas, 'x', 'CapSize',18, 'LineWidth', 3, 'MarkerSize', 15) ...
        % plot errorbar 1x
    % plot specifics
    ylim(limits/2)
    xlim([0 length(mus)+1])
    ylabel('Height Error (m)')
    xticklabels([{''}; LCclas(find(LCind==1))])
    xtickangle(45)
    set(gca, 'FontSize', 12)
    hold off
else
end

%% Weak v Strong
if ExecBeam == 1
    % setup vectors
    strong = [];
    weak = [];
    limits = [-2 2];
    for i = 1:length(Data)
```

```
        for j = 1:6
            side = char(Data(i).beam(j).track); % get left/right beam
            if side(2) == 'r'
                strong = [strong ...
                    Data(i).beam(j).DiffInterp(find(Data(i).beam(j).landcover≠0))]; ...
                    % get data points
            elseif side(2) == 'l'
                weak = [weak ...
                    Data(i).beam(j).DiffInterp(find(Data(i).beam(j).landcover≠0))]; ...
                    % get data points
            else
            end
        end
    end
    figure(fig) % new figure window
    fig = fig+1;
    hold on
    % get bar heights
    hstrong = histcounts(strong,[xbin Inf], 'Normalization','pdf');
    hweak = histcounts(weak,[xbin Inf], 'Normalization','pdf');
    % fit t-distributions
    pdstrong = fitdist(double(strong'), 'tlocationscale');
    pdweak = fitdist(double(weak'), 'tlocationscale');
    XGrid = limits(1):0.01:limits(2);
    Ystrong = pdf(pdstrong,XGrid);
    Yweak = pdf(pdweak,XGrid);
    plot(XGrid, Ystrong, 'LineWidth', 3, 'Color', [0 0.4470 0.7410]); % plot fit
    plot(XGrid, Yweak, 'LineWidth', 3, 'Color', [0.8500    0.3250    0.0980]); ...
        % plot fit
    wsbar = bar(xbin,[hstrong;hweak]', 'BarWidth', 1); % plot bars
    % bar specifics
    wsbar(1).FaceColor = [0         0.4470    0.7410];
    wsbar(1).FaceAlpha = 0.8;
    wsbar(2).FaceColor = [0.8500    0.3250    0.0980];
    wsbar(2).FaceAlpha = 0.8;
    % plot text
    infostrstrong = {num2str(pdstrong.mu,'\\mu = %1.3f m'), ...
        num2str(pdstrong.sigma, '\\sigma = %1.3f m'), num2str(pdstrong.nu, '\\nu ...
        = %1.3f')};
    annotation('TextBox', [0.2 0.6 0.5 0.5], 'String', infostrstrong, 'vert', ...
        'bottom', 'FitBoxToText', 'on', 'EdgeColor', [0 0.4470 0.7410])
    infostrweak = {num2str(pdweak.mu,'\\mu = %1.3f m'), num2str(pdweak.sigma, ...
        '\\sigma = %1.3f m'), num2str(pdweak.nu, '\\nu = %1.3f')};
    annotation('TextBox', [0.65 0.6 0.5 0.5], 'String', infostrweak, 'vert', ...
        'bottom', 'FitBoxToText', 'on', 'EdgeColor', [0.8500    0.3250    0.0980])
    % plot specifics
    xlim(limits)
    title('Strong Beam vs Weak Beam');
    ylabel('PDF')
```

```
    xlabel('Height Error in meter')
    legend({'strong', 'weak'})
    set(gca, 'FontSize', 12)
    hold off
else
end

%% Pass v Pass
if ExecPass == 1
    figure(fig) % new figure window
    fig = fig+1;
    passes = struct('pass', [], 'lat', [], 'lon', []); % setup struct
    for i = 1:length(Data)
        % setup vectors
        pass = [];
        latpass = [];
        lonpass = [];
        LCpass = [];
        for j = 1:6
            % fill vectors
            pass = [pass ...
                Data(i).beam(j).DiffInterp(find(Data(i).beam(j).landcover≠0))];
            latpass = [latpass ...
                Data(i).beam(j).lat(find(Data(i).beam(j).landcover≠0))'];
            lonpass = [lonpass ...
                Data(i).beam(j).lon(find(Data(i).beam(j).landcover≠0))'];
            LCpass = [LCpass ...
                Data(i).beam(j).landcover(find(Data(i).beam(j).landcover≠0))'];
        end
        % fill struct
        passes(i).pass = pass;
        passes(i).lat = latpass;
        passes(i).lon = lonpass;
        passes(i).lc = LCpass;
        clear pass
    end
    histocol = 2; % determine subplot colomn amount
    count = 0; % setup count
    for i = 1:length(Data)
        fill(i) = length(passes(i).pass) ≤ 100; % count data below thershold
    end
    %setup counts
    count_3 = 0;
    count_4 = 0;
    for i = 1:length(Data)
        if length(passes(i).pass) ≤ 100
        else
            count = count + 1; % count
            subplot(round((length(Data)-sum(fill))/histocol),histocol,count) % ...
```

```
                being subplot
            % fit t-distribution
            pdpass = fitdist(double(passes(i).pass'), 'tlocationscale');
            XGrid = limits(1):0.01:limits(2);
            YPlot = pdf(pdpass,XGrid);
            hold on
            plot(XGrid, YPlot, '-r', 'LineWidth', 2); % plot fit
            histogram(passes(i).pass, xbin,'Normalization','pdf'); % plot histogram
            % plot text
            titlestr = sprintf('Pass: RGT %s of Cycle %s with %d points', ...
                Data(i).file(22:25), Data(i).file(26:27), length(passes(i).pass));
            title(titlestr);
            infostr = {num2str(pdpass.mu,'\\mu = %1.3f m'), ...
                num2str(pdpass.sigma, '\\sigma = %1.3f m'), num2str(pdpass.nu, ...
                '\\nu = %1.3f')};
            if mod(count,2) == 1
                count_3 = count_3+1;
                annotation('TextBox', [0.14 1-count_3/7 0.5 0.5], 'String', ...
                    infostr, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                    'EdgeColor', 'none')
            else
                count_4 = count_4+1;
                annotation('TextBox', [0.585 1-count_4/7 0.5 0.5], 'String', ...
                    infostr, 'vert', 'bottom', 'FitBoxToText', 'on', ...
                    'EdgeColor', 'none')
            end
            % plot specifics
            xlim(limits)
            ylabel('PDF')
            xlabel('Height Error in meter')
            hold off
            % precision
            passprec(i,1) = precision(perc, pdpass, XGrid)*100;
            passprec(i,2) = passprec(i,1)/(length(passes(i).pass));
            passprec(i,3) = pdpass.mu*100;
            passprec(i,4) = pdpass.mu/(length(passes(i).pass));
        end
    end

    figure(fig) % new figrue window
    fig = fig+1;
    for i = 1:length(passes)
        [Npass(:,i),edges] = histcounts(double(passes(i).pass), xbin); % get ...
            bar heights
        strpass(i) = {sprintf('RGT %s of Cycle %s', Data(i).file(22:25), ...
            Data(i).file(26:27))}; % plot legend labels
    end
    bar(Npass, 'stacked') % plot bar
    %plot specifics
```

```
    title('Height Error Histogram with passes split')
    ylabel('Frequency')
    xticks([0 10 20 30 40 50 60])
    xticklabels({string(limits(1)), string(2*limits(1)/3), string(limits(1)/3), ...
        string(0), string(limits(2)/3), string(2*limits(2)/3), string(limits(2))})
    xlabel('Height Error in meter')
    legend(strpass)
    set(gca, 'FontSize', 12)
else
end
```