ICE, CLOUD, and Land Elevation Satellite (ICESat-2)

Algorithm Theoretical Basis Document (ATBD) for

Land - Vegetation Along-track products (ATL08)

Contributions by Land/VEG SDT Team Members and ICESAt-2 Project Science Office

(Amy Neuenschwander, Sorin Popescu, Ross Nelson, David Harding, Katherine Pitts, John Robbins, Dylan Pederson, and Ryan Sheridan)

ATBD Document prepared by

Amy Neuenschwander

June 2018

Content reviewed: technical approach, assumptions, scientific soundness, maturity, scientific utility of the data product

Contents

1 INTRODUCTION

This document describes the theoretical basis and implementation of the processing algorithms and data parameters for Level 3 land and vegetation heights for the non-polar regions of the Earth. The ATL08 product contains heights for both terrain and canopy in the along-track direction as well as other descriptive parameters derived from the measurements. At the most basic level, a derived surface height from the ATLAS instrument at a given time is provided relative to the WGS-84 ellipsoid. Height estimates from ATL08 can be compared with other geodetic data and used as input to higher-level ICESat-2 products, namely ATL13 and ATL18. ATL13 will provide estimates of inland water-related heights and associated descriptive parameters. ATL18 will consist of gridded maps for terrain and canopy features.

The ATL08 product will provide estimates of terrain heights, canopy heights, and canopy cover at fine spatial scales in the along-track direction. Along-track is defined as the direction of travel of the ICESat-2 satellite in the velocity vector. Parameters for the terrain and canopy will be provided at a fixed step-size of 100 m along the ground track referred to as a segment. A fixed segment size of 100 m was chosen to provide continuity of data parameters on the ATL08 data product. From an analysis perspective, it is difficult and cumbersome to attempt to relate canopy cover over variable lengths. Furthermore, a segment size of 100 m will facilitate a simpler combination of along-track data to create the gridded products.

We anticipate that the signal returned from the weak beam will be sufficiently weak and may prohibit the determination of both a terrain and canopy segment height, particularly over areas of dense vegetation. However, in more arid regions we anticipate producing a terrain height for both the weak and strong beams.

In this document, section 1 provides a background of lidar in the ecosystem community as well as describing photon counting systems and how they differ from discrete return lidar systems. Section 2 provides an overview of the Land and

Vegetation parameters and how they are defined on the data product. Section 3 describes the basic methodology that will be used to derive the parameters for ATL08. Section 4 describes the processing steps, input data, and procedure to derive the data parameters. Section 5 will describe the test data and specific tests that NASA's implementation of the algorithm should pass in order to determine a successful implementation of the algorithm.

1.1. Background

The Earth's land surface is a complex mosaic of geomorphic units and land cover types resulting in large variations in terrain height, slope, roughness, vegetation height and reflectance, often with the variations occurring over very small spatial scales. Documentation of these landscape properties is a first step in understanding the interplay between the formative processes and response to changing conditions. Characterization of the landscape is also necessary to establish boundary conditions for models which are sensitive to these properties, such as predictive models of atmospheric change that depend on land-atmosphere interactions. Topography, or land surface height, is an important component for many height applications, both to the scientific and commercial sectors. The most accurate global terrain product was produced by the Shuttle Radar Topography Mission (SRTM) launched in 2000; however, elevation data are limited to non-polar regions. The accuracy of SRTM derived elevations range from 5 – 10 m, depending upon the amount of topography and vegetation cover over a particular area. ICESat-2 will provide a global distribution of geodetic measurements (of both the terrain surface and relative canopy heights) which will provide a significant benefit to society through a variety of applications including sea level change monitoring, forest structural mapping and biomass estimation, and improved global digital terrain models.

In addition to producing a global terrain product, monitoring the amount and distribution of above ground vegetation and carbon pools enables improved characterization of the global carbon budget. Forests play a significant role in the terrestrial carbon cycle as carbon pools. Events, such as management activities (Krankina et al. 2012) and disturbances can release carbon stored in forest above ground biomass (AGB) into the atmosphere as carbon dioxide, a greenhouse gas that contributes to climate change (Ahmed et al. 2013). While carbon stocks in nations with continuous national forest inventories (NFIs) are known, complications with NFI carbon stock estimates exist, including: (1) ground-based inventory measurements are time consuming, expensive, and difficult to collect at large-scales (Houghton 2005; Ahmed et al. 2013); (2) asynchronously collected data; (3) extended time between repeat measurements (Houghton 2005); and (4) the lack of information on the spatial distribution of forest AGB, required for monitoring sources and sinks of carbon (Houghton 2005). Airborne lidar has been used for small studies to capture canopy height and in those studies canopy height variation for multiple forest types is measured to approximately 7 m standard deviation (Hall et al., 2011).

Although the spatial extent and changes to forests can be mapped with existing satellite remote sensing data, the lack of information on forest vertical structure and biomass limits the knowledge of biomass/biomass change within the global carbon budget. Based on the global carbon budget for 2015 (Quere et al., 2015), the largest remaining uncertainties about the Earth's carbon budget are in its terrestrial components, the global residual terrestrial carbon sink, estimated at 3.0 ± 0.8 GtC/year for the last decade (2005-2014). Similarly, carbon emissions from land-use changes, including deforestation, afforestation, logging, forest degradation and shifting cultivation are estimated at 0.9 ± 0.5 GtC /year. By providing information on vegetation canopy height globally with a higher spatial resolution than previously afforded by other spaceborne sensors, the ICESat-2 mission can contribute significantly to reducing uncertainties associated with forest vegetation carbon.

Although ICESat-2 is not positioned to provide global biomass estimates due to its profiling configuration and somewhat limited detection capabilities, it is anticipated that the data products for vegetation will be complementary to ongoing biomass and vegetation mapping efforts. Synergistic use of ICESat-2 data with other space-based mapping systems is one solution for extended use of ICESat-2 data. Possibilities include NASA's Global Ecosystems Dynamics Investigation (GEDI) lidar planned to fly onboard the International Space Station (ISS) or imaging sensors, such as Landsat 8, or NASA/ISRO –NISAR radar mission.

1.2 Photon Counting Lidar

Rather than using an analog, full waveform system similar to what was utilized on the ICESat/GLAS mission, ICESat-2 will employ a photon counting lidar. Photon counting lidar has been used successfully for ranging for several decades in both the science and defense communities. Photon counting lidar systems operate on the concept that a low power laser pulse is transmitted and the detectors used are sensitive at the single photon level. Due to this type of detector, any returned photon whether from the reflected signal or solar background can trigger an event within the detector. A discussion regarding discriminating between signal and background noise photons is discussed later in this document. A question of interest to the ecosystem community is to understand where within the canopy is the photon likely to be reflected. Figure 1.1 is an example of three different laser detector modalities: full waveform, discrete return, and photon counting. Full waveform sensors record the entire temporal profile of the reflected laser energy through the canopy. In contrast, discrete return systems have timing hardware that record the time when the amplitude of the reflected signal energy exceeds a certain threshold amount. A photon counting system, however, will record the arrival time associated with a single photon detection that can occur anywhere within the vertical distribution of the reflected signal. If a photon counting lidar system were to dwell over a surface for a significant number of shots (i.e. hundreds or more), the vertical distribution of the reflected photons will resemble a full waveform. Thus, while an individual photon could be reflected from anywhere within the vertical canopy, the probability distribution function (PDF) of that reflected photon would be the full waveform. Furthermore, the probability of detecting the top of the tree is not as great as detecting reflective surfaces positioned deeper into the canopy where the bulk of leaves and branches are located. As one might imagine, the PDF will differ according to canopy structure and vegetation physiology. For example, the PDF of a conifer tree will look different than broadleaf trees.

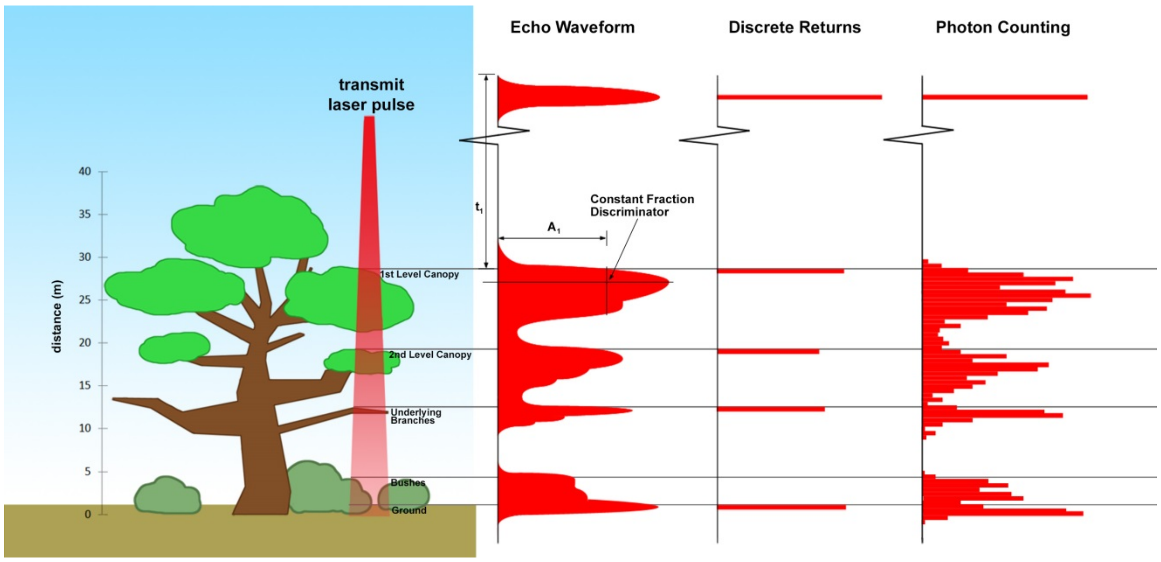

Figure 1.1. Various modalities of lidar detection. Adapted from Harding, 2009.

A cautionary note, the photon counting PDF that is illustrated in Figure 1.1 is merely an illustration if enough photons (i.e. hundreds of photons or more) were to be reflected from a target. In reality, due to the spacecraft speed, ATLAS will record 0 – 4 photons per transmit laser pulse over vegetation.

1.3 The ICESat-2 concept

The Advanced Topographic Laser Altimeter System (ATLAS) instrument designed for ICESat-2 will utilize a different technology than the GLAS instrument used for ICESat. Instead of using a high-energy, single-beam laser and digitizing the entire temporal profile of returned laser energy, ATLAS will use a multi-beam, micropulse laser (sometimes referred to as photon-counting). The travel time of each detected photon is used to determine a range to the surface which, when combined with satellite attitude and pointing information, can be geolocated into a unique XYZ location on or near the Earth's surface. For more information on how the photons from ICESat-2 are geolocated, refer to ATL03 ATBD. The XYZ positions from ATLAS are subsequently used to derive surface and vegetation properties. The ATLAS instrument will operate at 532 nm in the green range of the electromagnetic (EM) spectrum and will have a laser repetition rate of 10 kHz. The combination of the laser repetition rate and satellite velocity will result in one outgoing laser pulse approximately every 70 cm on the Earth's surface and each spot on the surface is ~13 m in diameter. Each transmitted laser pulse is split by a diffractive optical element in ATLAS to generate six individual beams, arranged in three pairs (Figure 1.2). The beams within each pair have different transmit energies ('weak' and 'strong', with an energy ratio of approximately 1:4) to compensate for varying surface reflectance. The beam pairs are separated by ~3.3 km in the across-track direction and the strong and weak beams are separated by ~2.5 km in the along-track direction. As ICESat-2 moves along its orbit, the ATLAS beams describe six tracks on the Earth's surface; the array is rotated slightly with respect to the satellite's flight direction so that tracks for the fore and aft beams in each column produce pairs of tracks – each separated by approximately 90 m.

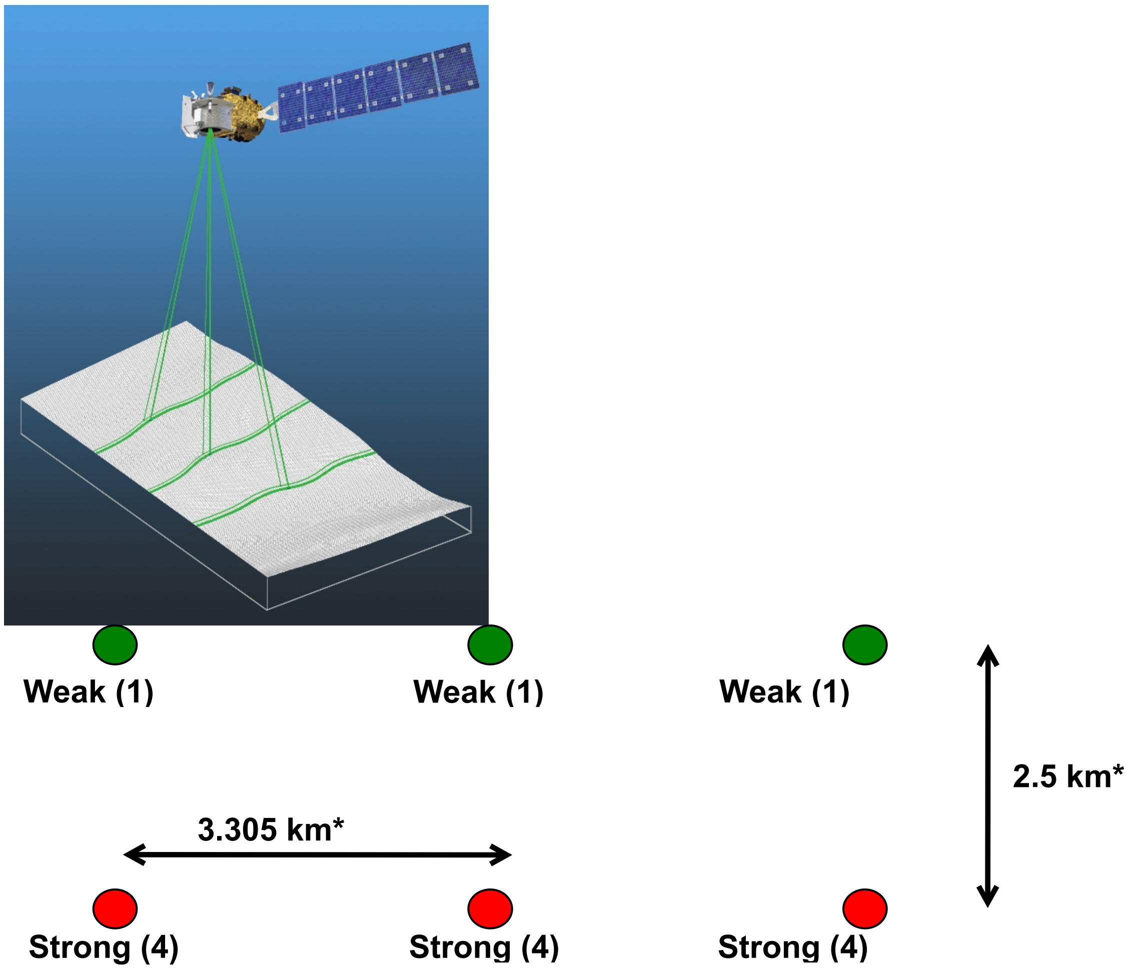

Figure 1.2. Schematic of 6-beam configuration for ICESat-2 mission. The laser energy will be split into 3 laser beam pairs – each pair having a weak spot (1X) and a strong spot (4X).

The motivation behind this multi-beam design is its capability to compute cross-track slopes on a per-orbit basis, which contributes to an improved understanding of ice dynamics. Previously, slope measurements of the terrain were determined via repeat-track and crossover analysis. The laser beam configuration as proposed for ICESat-2 is also beneficial for terrestrial ecosystems compared to GLAS as it enables a denser spatial sampling in the non-polar regions. To achieve a spatial sampling goal of no more than 2 km between equatorial ground tracks, ICESat-2 will be off-nadir pointed a maximum of 1.8 degrees from the reference ground track during the entire mission.

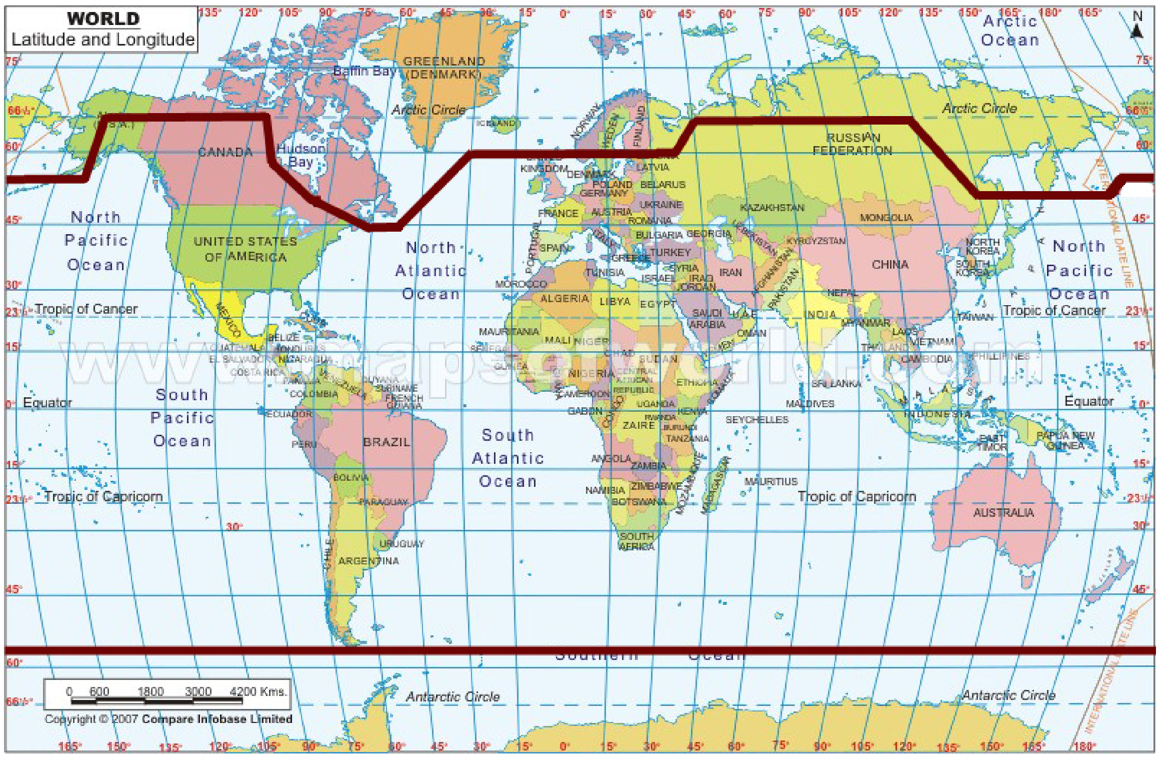

Figure 1.3. Illustration of off-nadir pointing scenarios. Over land in the mid-latitudes, ICESat-2 will be pointed away from the repeat ground tracks to increase the density of measurements over terrestrial surfaces.

ICESat-2 is designed to densely sample the Earth's surface, permitting scientists to measure and quantitatively characterize vegetation across vast expanses, e.g., nations, continents, globally. ICESat-2 will acquire synoptic measurements of vegetation canopy height, density, the vertical distribution of photosynthetically active material, leading to improved estimates of forest biomass, carbon, and volume. In addition, the orbital density, i.e., the number of orbits per unit area, at the end of the three year mission will facilitate the production of gridded global products. ICESat-2 will provide the means by which an accurate "snapshot" of global biomass and carbon may be constructed for the mission period.

1.4 Height Retrieval from ATLAS

Light from the ATLAS lasers reaches the earth's surface as flat disks of down-traveling photons approximately 50 cm in vertical extent and spread over approximately 12 m horizontally. Upon hitting the earth's surface, the photons are reflected and scattered in every direction and a handful of photons return to the ATLAS telescope's focal plane. The number of photon events per laser pulse is a function of outgoing laser energy, surface reflectance, solar conditions, and scattering and attenuation in the atmosphere. For highly reflective surfaces (such as land ice) and clear skies, approximately 10 signal photons from a single strong beam are expected to be recorded by the ATLAS instrument for a given transmit laser pulse. Over vegetated land where the surface reflectance is considerably less than snow or ice surfaces, we expect to see fewer returned photons from the surface. Whereas snow and ice surfaces have high reflectance at 532 nm (typical Lambertian reflectance between 0.8 and 0.98 (Martino, GSFC internal report, 2010)), canopy and terrain surfaces have much lower reflectance (typically around 0.3 for soil and 0.1 for vegetation) at 532 nm. As a consequence we expect to see 1/3 to 1/9 as many photons returned from terrestrial surfaces as from ice and snow surfaces. For vegetated surfaces, the number of reflected signal photon events per transmitted laser pulse is estimated to range between 0 to 4 photons.

The time measured from the detected photon events are used to compute a range, or distance, from the satellite. Combined with the precise pointing and attitude information about the satellite, the range can be geolocated into a XYZ point (known as a geolocated photon) above the WGS-84 reference ellipsoid. In addition to recording photons from the reflected signal, the ATLAS instrument will detect background photons from sunlight which are continually entering the telescope. A primary objective of the ICESat-2 data processing software is to correctly discriminate between signal photons and background photons. Some of this processing occurs at the ATL03 level and some of it also occurs within the software for ATL08. At ATL03, this discrimination is done through a series of three steps of progressively finer resolution with some processing occurring onboard the satellite prior to downlink of the raw data. The ATL03 data product produces a classification between signal and background (i.e. noise) photons, and further discussion on that classification process can be read in the ATL03 ATBD. In addition, all geophysical corrections (e.g. solid earth tide models, etc.) are applied to the position of the individual geolocated photons at the ATL03 level.

1.5 Accuracy Expected from ATLAS

There are a variety of elements that contribute to the elevation accuracy that are expected from ATLAS and the derived data products. Elevation accuracy is a composite of ranging precision of the instrument, radial orbital uncertainty, geolocation knowledge, forward scattering in the atmosphere, and tropospheric path delay uncertainty. The ranging precision seen by ATLAS will be a function of the laser pulse width, the surface area potentially illuminated by the laser, and uncertainty in the timing electronics. The requirement on radial orbital uncertainty is specified to be less than 4 cm and tropospheric path delay uncertainty is estimated to be 3 cm. In the case of ATLAS, the ranging precision for flat surfaces, is expected to have a standard deviation of approximately 25 cm. The composite of each of the errors can also be thought of as the spread of photons about a surface (see Figure 1.4) and is referred to as the point spread function or Znoise.

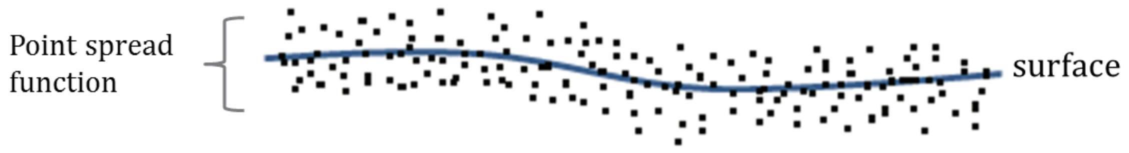

Figure 1.4. Illustration of the point spread function, also referred to as Znoise, for a series of photons about a surface.

The estimates of for a photon will be represented on the ATL03 data product as the final geolocated accuracy in the X, Y, and Z (or height) direction. In reality, these parameters have different temporal and spatial scales, however until ICESat-2 is on orbit, it is uncertain how these parameters will vary over time. As such, Equation 1.1 may change once the temporal aspects of these parameters are better understood. For a preliminary quantification of the uncertainties, Equation 1.1 is valid to incorporate the instrument related factors.

Eqn. 1.1

Although on the ATL03 product represents the best understanding of the uncertainty for each geolocated photon, it does not incorporate the uncertainty associated with local slope of the topography. The slope component to the geolocation uncertainty is a function of both the geolocation knowledge of the pointing (which is required to be less than 6.5 m) multiplied by the tangent of the surface slope. In a case of flat topography (<=1 degree slope), <= 25 cm, whereas in the case of a 10 degree surface slope, 119 cm. The uncertainty associated with the local slope will be combined with to produce the term . multiplied by the 6.5 m geolocation uncertainty factor will be used to update the expected uncertainty of the Z heights for ATLAS, .

Eqn. 1.2

Eqn. 1.3

Ultimately, the uncertainty that will be reported on the data product ATL08 will include the term and the local rms values of heights computed within each data parameter segment. For example, calculations of terrain height will be made on photons classified as terrain photons (this process is described in the following sections). The uncertainty of the terrain height for a segment is described in Equation 1.4, where the root mean square term of and rms of terrain heights are normalized by the number of terrain photons for that given segment.

Eqn. 1.4

1.6 Additional Potential Height Errors from ATLAS

Some additional potential height errors in the ATL08 terrain and vegetation product can come from a variety of sources including:

a. Vertical sampling error. ATLAS height estimates are based on a random sampling of the surface height distribution. Photons may be reflected from anywhere within the PDF of the reflecting surface; more specifically, anywhere from within the canopy.

b. Background noise. Random noise photons are mixed with the signal photons so classified photons will include random outliers.

c. Complex topography. The along-track product may not always represent complex surfaces, particularly if the density of ground photons does not support an accurate representation.

d. Vegetation. Dense vegetation may preclude reflected photon events from reaching the underlying ground surface. An incorrect estimation of the underlying ground surface will subsequently lead to an incorrect canopy height determination.

e. Misidentified photons. The product from ATL03 combined with additional noise filtering may not identify the correct photons as signal photons.

1.7 Dense Canopy Cases

Although the height accuracy produced from ICESat-2 is anticipated to be superior to other global height products (e.g. SRTM), for certain biomes photon counting lidar data as it will be collected by the ATLAS instrument present a challenge for extracting both the terrain and canopy heights, particularly for areas of dense vegetation. Due to the relatively low laser power, we anticipate that the along-track signal from ATLAS may lose ground signal under dense forest (e.g.

>96% canopy closure) and in situations where cloud cover obscures the terrestrial signal. In areas having dense vegetation, it is likely that only a handful of photons will be returned from the ground surface with the majority of reflections occurring from the canopy. A possible source of error can occur with both the canopy height estimates and the terrain heights if the vegetation is particularly dense and the ground photons were not correctly identified.

1.8 Sparse Canopy Cases

Conversely, sparse canopy cases also pose a challenge to vegetation height retrievals. In these cases, expected reflected photon events from sparse trees or shrubs may be difficult to discriminate between solar background noise photons. The algorithms being developed for ATL08 operate under the assumption that signal photons are close together and noise photons will be more isolated in nature. Thus, signal (in this case canopy) photons may be incorrectly identified as solar background noise on the data product. Due to the nature of the photon counting processing, canopy photons identified in areas that have extremely low canopy cover <15% will be filtered out and reassigned as noise photons.

2.

ATL08: DATA PRODUCT

The ATL08 product will provide estimates of terrain height, canopy height, and canopy cover at fine spatial scales in the along-track direction. In accordance with the HDF-driven structure of the ICESat-2 products, the ATL08 product will characterize each of the six Ground Tracks (GT) associated with each Reference Ground Track (RGT) for each cycle and orbit number (see Appendix 1 – ATBD Lexicon). Each ground track group has a distinct beam number, distance from the reference track, and transmit energy strength, but all beams will be processed independently using the same sequence of steps described within ATL08. Each ground track group (GT) on the ATL08 product contains subgroups for land and canopy heights segments as well as beam and reference parameters useful in the ATL08 processing. In addition, the labeled photons that are used to determine the data parameters will be indexed back to the ATL03 products such that they are available for further, independent analysis. A layout of the ATL08 HDF product is shown in Figure 2.1. The six GTs are numbered from left to right, regardless of satellite orientation.

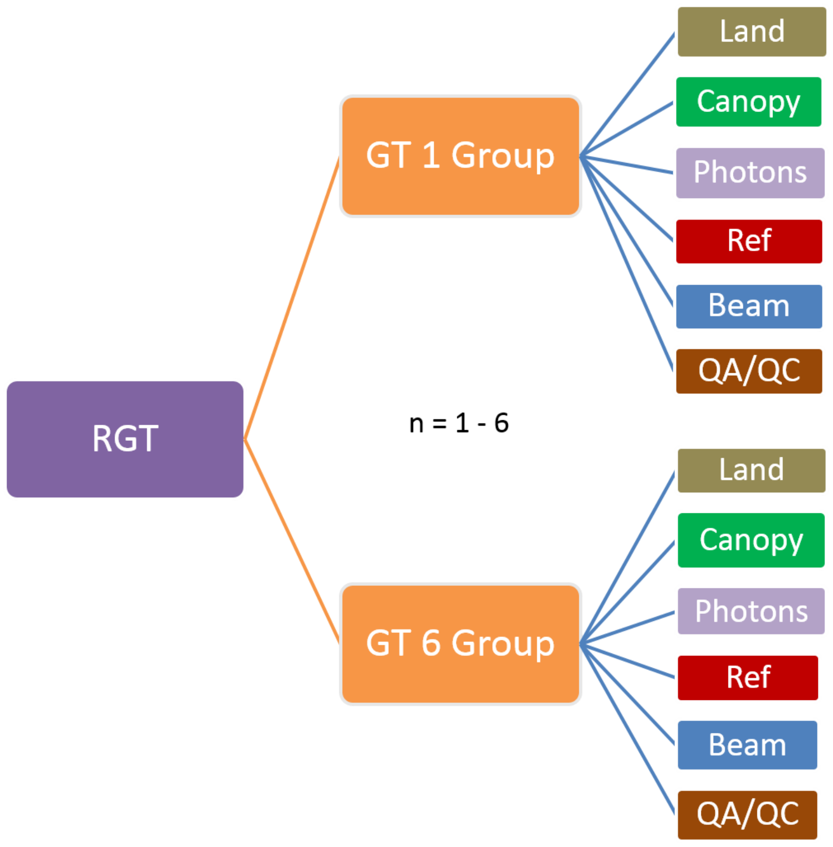

Figure 2.1. HDF Data structure for ATL08 products

For each data parameter, heights for the terrain surface and canopy heights will be provided at a fixed segment size of 100 meters along the ground track. Based on the satellite velocity and the expected number of reflected photons for land surfaces, each segment should have more than 100 signal photons, but in some instances there may be less than 100 signal photons per segment. If a segment has less than 50 (TBD, preliminary minimum photon recommendation) signal photons we feel this would not accurately represent the scene. Thus, an invalid value will be reported in all height fields except for an interpolated surface height. Each data parameter step will include a latitude and longitude based on the center position within that segment. In the event that a terrain height cannot be determined from a sufficient number of ground photons, (e.g. lack of photons penetrating through dense canopy), the reported terrain height will be the interpolated surface height. In those cases, a null value in the vegetation and canopy height product will be reported.

The ATL08 product can be referenced by region, which are roughly assigned by continent, as shown by Figure 2.2. For the regions covering Antarctica (regions 7, 8, 9, 10) and Greenland (region 11), the ATL08 algorithm will assume that no canopy is present. Note that the regions for each ICESat-2 product are not the same.

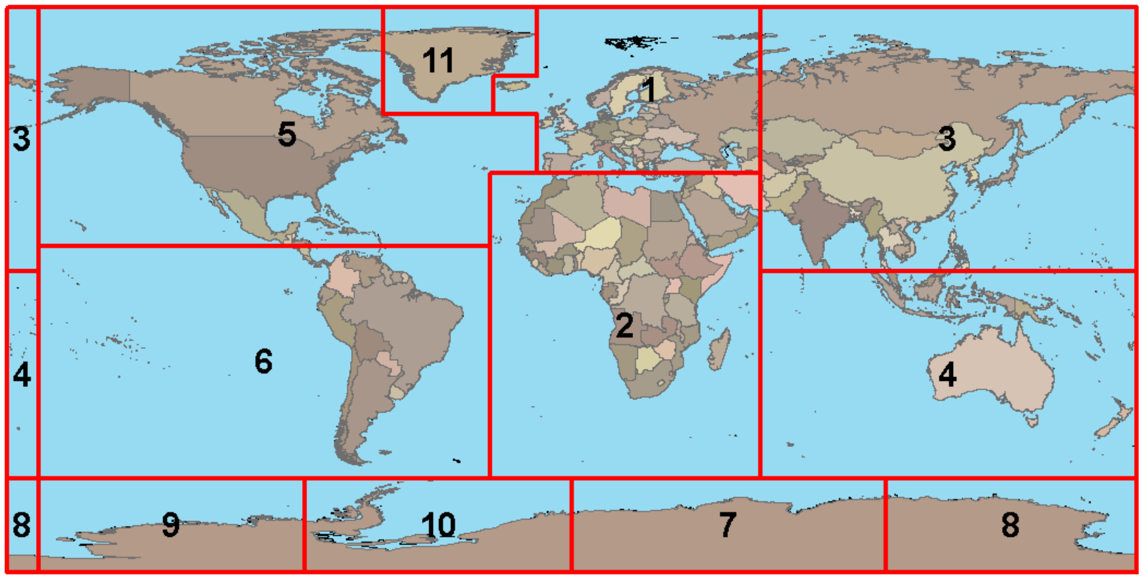

Figure 2.2. ATL08 product regions.

2.1 *Subgroup: Land Parameters*

ATL08 terrain height parameters are defined in terms of the absolute height above the reference ellipsoid.

Table 2.1. Summary Table of Land parameters on ATL08

Group	Data type	Description	Source
segment_id_beg	Integer	First along-track segment_id	ATL03

		number in 100-m segment	
segment_id_end	Integer	Last along-track segment_id number in 100-m segment	ATL03
h_te_mean	Float	Mean terrain height for segment	computed
h_te_median	Float	Median terrain height for segment	computed
h_te_min	Float	Minimum terrain height for segment	computed
h_te_max	Float	Maximum terrain height for segment	computed
h_te_mode	Float	Mode of terrain height for segment	computed
h_te_skew	Float	Skew of terrain height for segment	computed
n_te_photons	Integer	Number of ground photons in segment	computed
h_te_interp	Float	Interpolated terrain surface height at mid-point of segment	computed
h_te_std	Float	Standard deviation of ground heights about the interpolated ground surface	computed
h_te_uncertainty	Float	Uncertainty of ground height estimates. Includes all known uncertainties such as geolocation, pointing angle, timing, radial orbit errors, etc.	computed from Equation 1.4
terrain_slope	Float	Slope of terrain within segment	computed
n_photons	Integer	Total number of classed photons for 100 m segment	computed
h_te_best_fit	Float	Best fit terrain elevation at the 100 m segment mid-point location	computed

2.1.1 Georeferenced_segment_number_beg

(parameter = segment_id_beg). The first along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.1.2 Georeferenced_segment_number_end

(parameter = segment_id_end). The last along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.1.3 Segment_terrain_height_mean

(parameter = h_te_mean). Estimated mean of the terrain height above the reference ellipsoid derived from classified ground photons within the 100 m segment. If a terrain height cannot be directly determined within the segment (i.e. there are not a sufficient number of ground photons), only the interpolated terrain height will be reported. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons.

2.1.4 Segment_terrain_height_med

(parameter = h_te_median). Median terrain height above the reference ellipsoid derived from the classified ground photons within the 100 m segment. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons.

2.1.5 Segment_terrain_height_min

(parameter = h_te_min). Minimum terrain height above the reference ellipsoid derived from the classified ground photons within the 100 m segment. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons.

2.1.6 Segment_terrain_height_max

(parameter = h_te_max). Maximum terrain height above the reference ellipsoid derived from the classified ground photons within the 100 m segment. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons.

2.1.7 Segment_terrain_height_mode

(parameter = h_te_mode). Mode of the classified ground photon heights above the reference ellipsoid within the 100 m segment. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons.

2.1.8 Segment_terrain_height_skew

(parameter = h_te_skew). The skew of the classified ground photons within the 100 m segment. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons.

2.1.9 Segment_number_terrain_photons

(parameter = n_te_photons). Number of terrain photons identified in segment.

2.1.10 Segment height_interp

(parameter = h_te_interp). Interpolated terrain surface height above the reference ellipsoid from ATL08 processing at the mid-point of each segment. This interpolated surface is the FINALGROUND estimate (described in section 4.9).

2.1.11 Segment h_te_std

(parameter = h_te_std). Standard deviations of terrain points about the interpolated ground surface within the segment. Provides an indication of surface roughness.

2.1.12 Segment_terrain_height_uncertainty

(parameter = h_te_uncertainty). Uncertainty of the mean terrain height for the segment. This uncertainty incorporates all systematic uncertainties (e.g. timing, orbits, geolocation, etc.) as well as uncertainty from errors of identified photons. This parameter is described in Section 1, Equation 1.4. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing). This parameter will be derived from only classified ground photons. The term in Equation 1.4 represents the standard deviation of the terrain height residuals about the FINALGROUND estimate.

2.1.13 Segment_terrain_slope

(parameter = terrain_slope). Slope of terrain within each segment. Slope is computed from a linear fit of the terrain photons. It estimates the rise [m] in relief over each segment [100 m]; e.g., if the slope value is 0.04, there is a 4 m rise over the 100 m segment. Required input data are the classified terrain photons.

2.1.14 Segment number_of_photons

(parameter = n_photons). Number of classed photons within each 100 m segment.

2.1.15 Segment_terrain_height_best_fit

(parameter = h_te_best_fit). The best fit terrain elevation at the mid-point location of each 100 m segment. The mid-segment terrain elevation is determined by selecting the best of three fits – linear, 3rd order and 4th order polynomials – to the terrain photons and interpolating the elevation at the mid-point location of the 100 m segment. For the linear fit, a slope correction and weighting is applied to each ground photon based on the distance to the slope height at the center of the segment.

2.2 Subgroup: Vegetation Parameters

Canopy parameters will be reported on the ATL08 data product in terms of both the absolute height above the reference ellipsoid as well as the relative height above an estimated ground. The relative canopy height, H_i, is computed as the height from an identified canopy photon minus the interpolated ground surface for the same horizontal geolocation (see Figure 2.3). Thus, each identified signal photon above an interpolated surface (including a buffer distance based on the instrument point spread function) is by default considered a canopy photon. Canopy parameters will only be computed for segments where more than 5% of the classed photons are classified as canopy photons.

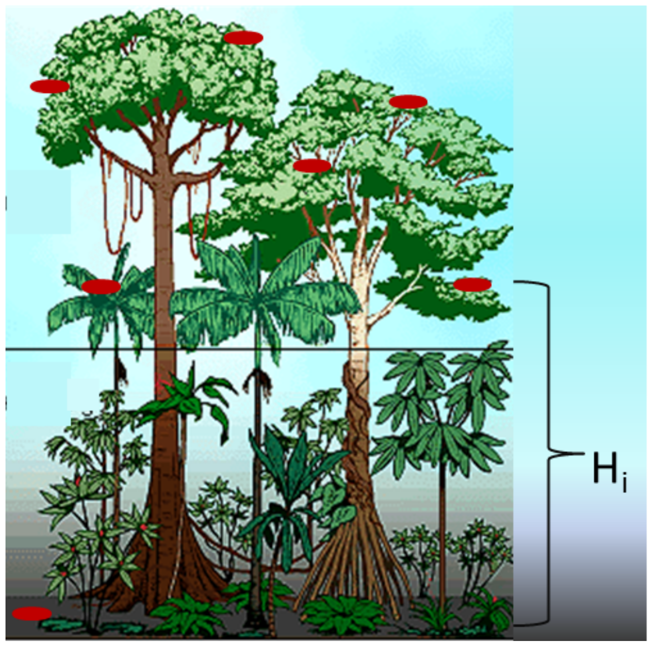

Figure 2.3. Illustration of canopy photons (red dots) interaction in a vegetated area. Relative canopy heights, H_i, are computed by differencing the canopy photon height from an interpolated terrain surface.

Table 2.2. Summary table of canopy parameters on ATL08

Group	Data type	Description	Source
segment_id_beg	Integer	First along-track segment_id number in 100-m segment	ATL03
segment_id_end	Integer	Last along-track segment_id number in 100-m segment	ATL03
canopy_h_metrics	Float	Absolute (H##) and relative (RH##) canopy height metrics calculated at the following percentiles: 25, 50, 60, 70, 75, 80, 85, 90, 95, 99.	computed
h_canopy_abs	Float	95% height of all the individual absolute canopy heights for segment. Should be equivalent to H95.	computed
h_canopy	Float	95% height of all the individual relative canopy heights for segment. Should be equivalent to RH95.	computed
h_mean_canopy_abs	Float	Mean of individual absolute	computed

		canopy heights within segment	
h_mean_canopy	Float	Mean of individual relative canopy heights within segment	computed
h_dif_canopy	Float	Difference between h_canopy and h_median_canopy	computed
h_median_canopy_abs	Float	Median of individual absolute canopy heights within segment. Should be equivalent to H50	computed
h_median_canopy	Float	Median of individual relative canopy heights within segment. Should be equivalent to RH50	computed
h_min_canopy_abs	Float	Minimum of individual absolute canopy heights within segment	computed
h_min_canopy	Float	Minimum of individual relative canopy heights within segment	computed
h_max_canopy_abs	Float	Maximum of individual absolute canopy heights within segment. Should be equivalent to H100	computed
h_max_canopy	Float	Maximum of individual relative canopy heights within segment. Should be equivalent to RH100	computed
h_canopy_uncertainty	Float	Uncertainty of the relative canopy height (h_canopy)	computed
canopy_openness	Float	STD of relative heights for all photons classified as canopy photons within the segment to provide inference of canopy openness	computed
toc_roughness	Float	STD of relative heights all photons classified as top of canopy within the segment	computed
h_canopy_quad	Float	Quadratic mean canopy height	computed
n_ca_photons	Integer4	Number of canopy photons within 100 m segment	computed
n_toc_photons	Integer4	Number of top of canopy photons within 100 m segment	computed
canopy_closure	Float	Relative canopy closure	computed
centroid_height	Float	Absolute height above reference ellipsoid associated with the centroid of all signal photons	computed
n_photons	Integer	Number of classed photons in 100 m segment	computed
canopy_flag	Integer	Flag indicating that canopy was detected using the Landsat Continuous Cover data product	computed

landsat_flag	Integer	Flag indicating that Landsat continuous cover data product had more than 50% values >100 for L-km segment	computed

2.2.1 Georeferenced_segment_number_beg

(parameter = segment_id_beg). The first along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.2.2 Georeferenced_segment_number_end

(parameter = segment_id_end). The last along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.2.3 Canopy_height_metrics

(parameter = canopy_h_metrics). The absolute height metrics (H##) and relative height metrics above the estimated terrain surface (RH##) of classified canopy photons above the ellipsoid. The height metrics are sorted based on a cumulative distribution and calculated at the following percentiles: 25, 50, 60, 70, 75, 80, 85, 90, 95, 99. These height metrics are often used in the literature to characterize vertical structure of vegetation. One important distinction of these canopy height metrics compared to those derived from other lidar systems (e.g. LVIS or GEDI) is that the ICESat-2 canopy height metrics are heights above the ground surface. These metrics do not include the ground photons. Required input data are the absolute canopy heights of all canopy photons and the relative canopy heights of the estimated terrain surface.

2.2.4 Absolute_segment_canopy_height

(parameter = h_canopy_abs). The absolute 95% height of classified canopy photon heights above the ellipsoid. The absolute height from classified canopy photons are sorted into a cumulative distribution, and the height associated with the 95% height is reported. This parameter is equivalent to H95.

2.2.5 Segment_canopy_height

(parameter = h_canopy). The relative 95% height of classified canopy photon heights above the estimated terrain surface. Relative canopy heights have been computed by differencing the canopy photon height from the estimated terrain surface in the ATL08 processing. The relative canopy heights are sorted into a cumulative distribution, and the height associated with the 95% height is reported. This parameter is equivalent to RH95.

2.2.6 Absolute_segment_mean_canopy

(parameter = h_mean_canopy_abs). The absolute mean canopy height for the segment. Absolute canopy heights are the photons heights above the reference ellipsoid. These heights are averaged.

2.2.7 Segment_mean_canopy

(parameter = h_mean_canopy). The mean canopy height for the segment. Relative canopy heights have been computed by differencing the canopy photon height from the estimated terrain surface in the ATL08 processing. These heights are averaged.

2.2.8 Segment_dif_canopy

(parameter = h_dif_canopy). Difference between h_canopy and h_median_canopy. This parameter is one metric to describe the vertical distribution of the canopy within the segment.

2.2.9 Absolute_segment_median_canopy

(parameter = h_median_canopy_abs). The median absolute canopy height for the segment. This parameter is equivalent to H50. The absolute canopy heights are sorted into a cumulative distribution, and the height associated with the 50% height is reported. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.10 Segment_median_canopy

(parameter = h_median_canopy). The median relative canopy height for the segment. This parameter is equivalent to RH50. The relative canopy heights are sorted into a cumulative distribution, and the height associated with the 50% height is reported. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.11 Absolute_segment_min_canopy

(parameter = h_min_canopy_abs). The minimum absolute canopy height for the segment. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.12 Segment_min_canopy

(parameter = h_min_canopy). The minimum relative canopy height for the segment. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.13 Absolute_segment_max_canopy

(parameter = h_max_canopy_abs). The maximum absolute canopy height for the segment. This product is equivalent to H100 metric reported in the literature. This parameter, however, has the potential for error as random solar background noise may not have been fully rejected. It is recommended that h_canopy or h_canopy_abs (i.e., the 95% canopy height) be considered as the top of canopy measurement. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.14 Segment_max_canopy

(parameter = h_max_canopy). The maximum relative canopy height for the segment. This product is equivalent to RH100 metric reported in the literature. This parameter, however, has the potential for error as random solar background noise may not have been fully rejected. It is recommended that h_canopy or h_canopy_abs (i.e., the 95% canopy height) be considered as the top of canopy measurement. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.15 Segment_canopy_height_uncertainty

(parameter = h_canopy_uncertainty). Uncertainty of the relative canopy height for the segment. This uncertainty incorporates all systematic uncertainties (e.g. timing, orbits, geolocation, etc.) as well as uncertainty from errors of identified photons. This parameter is described in Section 1, Equation 1.4. If there are not a sufficient number of ground photons, an invalid value will be reported –no interpolation will be done. In the case for canopy height uncertainty, the parameter is comprised of both the terrain uncertainty within the segment but also the top of canopy residuals. Required input data is classified point cloud (i.e. photons labeled as either top of canopy or ground in the ATL08 processing). This parameter will be derived from only classified top of canopy photons. The canopy height uncertainty is derived from Equation 1.4, shown below as Equation 1.5, represents the standard deviation of the terrain points and the standard deviation of the top of canopy height photons.

Eqn 1.5

2.2.16 Segment_canopy_openness

(parameter = canopy_openness). Standard deviation of relative canopy heights within each segment. This parameter will potentially provide an indicator of canopy openness as a greater standard deviation of heights indicates greater penetration of the laser energy into the canopy. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.17 Segment_top_of_canopy_roughness

(parameter = toc_roughness). Standard deviation of relative top of canopy heights within each segment. This parameter will potentially provide an indicator of canopy variability. Required input data is classified point cloud (i.e. photons labeled as the top of the canopy in the ATL08 processing).

2.2.18 Segment_canopy_quadratic_height

(parameter = h_canopy_quad). The relative quadratic mean height of relative classified canopy photons above the reference DEM available for the product. The quadratic mean height is computed as:

2.2.19 Segment_number_canopy_photons

(parameter = n_ca_photons). Number of canopy photons within each segment. Required input data is classified point cloud (i.e. photons labeled as either canopy or ground in the ATL08 processing).

2.2.20 Segment_number_top_canopy_photons

(parameter = n_toc_photons). Number of top of canopy photons within each segment. Required input data is classified point cloud (i.e. photons labeled as top of canopy in the ATL08 processing).

2.2.21 Segment_rel_canopy_closure

(parameter = canopy_closure). Relative canopy closure within the segment

2.2.22 Centroid_height

(parameter = centroid_height). Optical centroid of all photons classified as either canopy or ground points within a segment. The heights used in this calculation are absolute heights above the reference ellipsoid. This parameter is equivalent to the centroid height produced on ICESat GLA14.

2.2.23 Segment_number_of_photons

(parameter = n_photons). Number of classed photons within 100 m segment.

2.2.24 Canopy_flag

(parameter = canopy_flag). Flag indicating that canopy was detected using the Landsat Continuous Cover product for the *L-km* segment. Currently, if more than 5% of the Landsat CC pixels along the profile have canopy in them, we make the assumption canopy is present along the entire *L-km* segment.

2.2.25 LANDSAT_flag

(parameter = landsat_flag). Flag indicating that more than 50% of the Landsat Continuous Cover product have values >100 for the *L-km* segment which then we assume canopy in present along the *L-km* segment.

2.3 *Subgroup: Photons*

The subgroup for photons contains the classified photons that were used to generate the parameters within the land or canopy subgroups. Each photon that is identified as being likely signal will be classified as: 0 = noise, 1 = ground, 2 = canopy, or 3 = top of canopy. The index values for each classified photon will be provided such that they can be extracted from the ATL03 data product for independent evaluation.

Table 2.3. Summary Table for photon parameters for the ATL08 product

Group	Data Type	Description	Source
classed_PC_indx	Float	Indices of photons tracking back to ATL03 that surface finding software identified and used within the creation of the data products.	ATL03
classed_PC_flag	Integer	Classification flag for each photon as either noise, ground, canopy, or top of canopy.	computed
ph_segment_id	Integer	Georeferenced bin number (20-m) associated with each photon	ATL03
segment_id_beg	Integer	First along-track segment_id number in 100-m segment	ATL03
segment_id_end	Integer	Last along-track segment_id number in 100-m segment	ATL03
d_flag	Integer	Flag indicating whether DRAGANN labeled the photon as noise or signal	computed
n_photons	integer	Number of classed photons in 100 m segment	computed

2.3.1 Indices_of_classed_photons

(parameter = classed_PC_indx). Indices of photons tracking back to ATL03 that surface finding software identified and used within the creation of the data products for a given segment.

2.3.2 Photon_class

(parameter = classed_PC_flag). Classification flags for a given segment. 0 = noise, 1 = ground, 2 = canopy, 3 = top of canopy. The final ground and canopy classification are flags 1-3. The full canopy is the combination of flags 2 and 3.

2.3.3 Georeferenced_segment_number

(parameter = ph_segment_id). The segment_id associated with every photon in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.3.4 Georeferenced_segment_number_beg

(parameter = segment_id_beg). The first along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.3.5 Georeferenced_segment_number_end

(parameter = segment_id_end). The last along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.3.6 DRAGANN_flag

(parameter = d_flag). Flag indicating the labeling of DRAGANN noise filtering for a given photon. 0 = noise, 1=signal.

2.3.7 Segment_number_of_photons

(parameter = n_photons). Number of classed photons within 100 m segment.

2.4 Subgroup: Reference data

The reference data subgroup contains parameters and information that are useful for determining the terrain and canopy heights that are reported on the product. In addition to position and timing information, these parameters include the reference DEM height, reference landcover type, and flags indicating water or snow.

Table 2.4. Summary Table for reference parameters for the ATL08 product

Group	**Data Type**	**Description**	**Source**
segment_id_beg	Integer	First along-track segment_id number in 100-m segment	ATL03
segment_id_end	Integer	Last along-track segment_id number in 100-m segment	ATL03
latitude	Float	Center latitude of signal photons within each segment	ATL03
longitude	Float	Center longitude of signal photons within each segment	ATL03
delta_time	Float	Mid-segment GPS time in seconds past an epoch. The epoch is provided in the metadata at the file level	ATL03

delta_time_beg	Float	Delta time of the first photon in the segment	ATL03
delta_time_end	Float	Delta time of the last photon in the segment	ATL03
night_flag	Integer	Flag indicating whether the measurements were acquired during night time conditions	computed
n_photons	Float	Number of classed photons per 100 m segment	computed
dem_h	Float4	Reference DEM elevation	external
dem_flag		Source of reference DEM	external
h_dif_ref	Float4	Difference between h_te_median and dem_h	computed
terrain_flg	Integer	Terrain flag quality check to indicate a deviation from the reference DTM	computed
segment_landcover	Integer4	Reference landcover for segment derived from best global landcover product available	external
segment_watermask	Integer4	Water mask indicating inland water produced from best sources available	external
segment_snowmask	Integer4	Daily snow cover mask derived from best sources	external
urban_flag	Integer	Flag indicating segment is located in an urban area	external
surf_type	Integer1	Flags describing surface types: 0=not type, 1=is type. Order of array is land, ocean, sea ice, land ice, inland water.	ATL03

2.4.1 Georeferenced_segment_number_beg

(parameter = segment_id_beg). The first along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.4.2 Georeferenced_segment_number_end

(parameter = segment_id_end). The last along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.4.3 Segment_latitude

(parameter = latitude). Center latitude of signal photons within each segment

2.4.4 Segment_longitude

(parameter = longitude). Center longitude of signal photons within each segment

2.4.5 Delta time

(parameter = delta_time). Mean GPS time for the segment in seconds past an epoch. The epoch is listed in the metadata at the file level.

2.4.6 Delta_time_beg

(parameter = delta_time_beg). Delta time for the first photon in the segment in seconds past an epoch. The epoch is listed in the metadata at the file level.

2.4.7 Delta_time_end

(parameter = delta_time_end). Delta time for the last photon in the segment in seconds past an epoch. The epoch is listed in the metadata at the file level.

2.4.8 Night_Flag

(parameter = night_flag). Flag indicating the data were acquired in night conditions: 0 = day, 1 = night. Night flag is set when solar elevation is below 0.0 degrees.

2.4.9 Segment_number_of_photons

(parameter = n_photons). Number of classed photons within 100 m segment.

2.4.10 Segment_reference_DTM

(parameter = dem_h). Reference terrain height value for segment determined by the "best" DEM available based on data location. All heights in ICESat-2 are referenced to the WGS 84 ellipsoid unless clearly noted otherwise. DEM is taken from a variety of ancillary data sources: GIMP, GMTED, MSS. The DEM source flag indicates which source was used.

2.4.11 Segment_reference_DEM_source

(parameter = dem_flag). Indicates source of the reference DEM height. Values: 0=None, 1=GIMP, 2=GMTED, 3=MSS.

2.4.12 Segment_terrain_difference

(parameter = h_dif_ref). Difference between h_te_median and dem_h. Since the mean terrain height is more sensitive to outliers, the median terrain height will be evaluated against the reference DEM. This parameter will be used as an internal data quality check with the notion being that if the difference exceeds a threshold (TBD) a terrain quality flag (terrain_flg) will be triggered.

2.4.13 Segment_terrain flag

(parameter = terrain_flg). Terrain flag to indicate confidence in the derived terrain height estimate. If h_dif_ref exceeds a threshold (TBD) the terrain_flg parameter will be set to 1. Otherwise, it is 0.

2.4.14 Segment_landcover

(parameter = segment_landcover). Segment landcover will be based on best available global landcover product used for reference. One potential source is the 0.5 km global mosaics of the standard MODIS land cover product (Channan et al, 2015; Friedl et al, 2010; available online at http://glcf.umd.edu/data/lc/index.shtml). Here, 17 classes are defined ranging from evergreen (needle and broadleaf forest), deciduous (needle and broadleaf forest), shrublands, woodlands, savanna and grasslands, agriculture, to urban. The most current year processed for this product is based on MODIS measurements from 2012.

2.4.15 Segment_watermask

(parameter = segment_watermask). Water mask (i.e., flag) indicating inland water as referenced from the Global Raster Water Mask at 250 m spatial resolution (Carroll et al, 2009; available online at http://glcf.umd.edu/data/watermask/). 0 = no water; 1 = water.

2.4.16 Segment_snowcover

(parameter = segment_snowcover). Daily snowcover mask (i.e. flag) indicating a likely presence of snow within each segment produced from best available source used for reference (e.g. MODIS or Landsat). The snow mask will be the same snow mask as used for ATL09 Atmospheric Products. 0 = no snow; 1 = snow

2.4.17 Urban_flag

(parameter = urban_flag). The urban flag indicates that a segment is likely located over an urban area. In these areas, buildings may be misclassified as canopy, and thus the canopy products may be incorrect. The urban flag is sourced from the "urban and built up" classification on the MODIS land cover product (Channan et al, 2015; Friedl et al, 2010; available online at http://glcf.umd.edu/data/lc/index.shtml). 0 = not urban; 1 = urban.

2.4.18 Surface_type

(parameter = surf_type). The surface type for a given segment is determined at the major frame rate (every 200 shots, or ~140 meters along-track) and is a two-dimensional array surf_type(n, nsurf), where n is the major frame number, and nsurf is the number of possible surface types such that surf_type(n,isurf) is set to 0 or 1 indicating if surface type isurf is present (1) or not (0), where isurf = 1 to 5 (land, ocean, sea ice, land ice, and inland water) respectively.

2.5 Subgroup: Beam data

The subgroup for beam data contains basic information on the geometry and pointing accuracy for each beam.

Table 2.5. Summary table for beam parameters for the ATL08 product

Group	Data Type	Units	Description	Source
segment_id_beg	Integer		First along-track segment_id number in 100-m segment	ATL03
segment_id_end	Integer		Last along-track segment_id number in 100-m segment	ATL03
atlas_pa	Float		Off nadir pointing angle of the spacecraft	ATL03
beam_number	Integer		Laser beam number. 1, 3, 5 are strong beams. 2, 4, 6 are weak beams.	ATL03
rgt	Integer		The reference ground track (RGT) is the track on the earth at which the vector bisecting	ATL03

		laser beams 3 and 4 is pointed during repeat operations	
sigma_h	Float	Total vertical uncertainty due to PPD and POD	ATL03
sigma_along	Float	Total along-track uncertainty due to PPD and POD knowledge	ATL03
sigma_across	Float	Total cross-track uncertainty due to PPD and POD knowledge	ATL03
sigma_topo	Float	Uncertainty of the geolocation knowledge due to local topography (Equation 1.3)	computed
sigma_atlas_land	Float	Total uncertainty that includes sigma_h plus the geolocation uncertainty due to local slope Equation 1.2	computed
cloud_flag_asr	Integer	Cloud confidence flag from ATL09 indicating clear skies	ATL09
msw_flag	Integer	Multiple scattering warning product produced on ATL09	ATL09
asr	Float	Apparent surface reflectance	ATL09
snr	Float	Background signal to noise level	Computed
solar_azimuth	Float	The azimuth (in degrees) of the sun position vector from the reference photon bounce point position in the local ENU frame. The angle is measured from North and is positive towards East.	ATL03g
solar_elevation	Float	The elevation of the sun position vector from the reference photon bounce point position in the local ENU frame. The angle is measured from the East-North plane and is positive Up.	ATL03g

n_seg_ph	Integer	Number of photons within each land segment	computed

2.5.1 Georeferenced_segment_number_beg

(parameter = segment_id_beg). The first along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.5.2 Georeferenced_segment_number_end

(parameter = segment_id_end). The last along-track segment_id in each 100-m segment. Each 100-m segment consists of five sequential 20-m segments provided from the ATL03 product, which are labeled as segment_id. The segment_id is a seven digit number that uniquely identifies each along track segment, and is written at the along-track geolocation segment rate (i.e. ~20m along track). The four digit RGT number can be combined with the seven digit segment_id number to uniquely define any along-track segment number. Values are sequential, with 0000001 referring to the first segment after the equatorial crossing of the ascending node.

2.5.3 ATLAS Pointing Angle

(parameter = atlas_pa). Off nadir pointing angle (in radians) of the satellite to increase spatial sampling in the non-polar regions.

2.5.4 Beam_number

(parameter = beam_number). The sequential laser pulses emitted from the ATLAS instrument that illuminate spots on the earth's surface are called laser beams. ATLAS generates six laser beams. The laser beam numbering convention follows the ATLAS instrument convention with strong beams numbered 1, 3, and 5 and weak beams numbered 2, 4, and 6.

2.5.5 Reference ground track

(parameter = rgt). The reference ground track (RGT) is the track on the earth at which the vector bisecting laser beams 3 and 4 (or GT2L and GT2R) is pointed during repeat operations. Each RGT spans the part of an orbit between two ascending equator crossings and are numbered sequentially. The ICESat-2 mission has 1387 RGTs, numbered from 0001xx to 1387xx. The last two digits refer to the cycle number.

2.5.6 Sigma_h

(parameter = sigma_h). Total vertical uncertainty due to PPD (Precise Pointing Determination), POD (Precise Orbit Determination), and geolocation errors. Specifically, this parameter includes radial orbit error, , tropospheric errors, , forward scattering errors, , instrument timing errors, , and off-nadir pointing geolocation errors . The component parameters are pulled from ATL03 and ATL09. Sigma_h is the root sum of squares of these terms as detailed in Equation 1.1.

2.5.7 Sigma_along

(parameter = sigma_along). Total along-track uncertainty due to PPD and POD knowledge. This parameter is pulled from ATL03.

2.5.8 Sigma_across

(parameter = sigma_across). Total cross-track uncertainty due to PPD and POD knowledge. This parameter is pulled from ATL03.

2.5.9 Sigma_topo

(parameter = sigma_topo). Uncertainty in the geolocation due to local surface slope as described in Equation 1.3. The local slope is multiplied by the 6.5 m geolocation uncertainty factor that will be used to determine the geolocation uncertainty. The geolocation error will be computed from a 100 m sample due to the local slope calculation at that scale.

2.5.10 Sigma_ATLAS_LAND

(parameter = sigma_atlas_land). Total vertical geolocation error due to ranging, and local surface slope. The parameter is computed for ATL08 as described in Equation 1.2. The geolocation error will be computed from a 100 m sample due to the local slope calculation at that scale.

2.5.11 Cloud flag

(parameter = cloud_flag_asr). Cloud confidence flag from ATL09. Flag indicates potential clear skies from ATL09. Cloud Product Flags:

0	= High confidence clear skies
1	= Medium confidence clear skies
2	= Low confidence clear skies
3	= Low confidence cloudy skies
4	= Medium confidence cloudy skies
5	= High confidence cloudy skies

2.5.12 MSW

(parameter = msw_flag). Multiple scattering warning computed in the ATL09 atmospheric processing and delivered on the ATL09 data product.

2.5.13 Computed_Apparent_Surface_Reflectance

(parameter = asr). Apparent surface reflectance computed in the ATL09 atmospheric processing and delivered on the ATL09 data product.

2.5.14 Signal_to_Noise_Ratio

(parameter = snr). The Signal to Noise Ratio of geolocated photons as determined **by the ratio** of the superset of ATL03 signal and DRAGANN found signal photons used for processing the ATL08 segments **to the background photons (i.e., noise)** within the same ATL08 segments.

2.5.15 Solar Azimuth

(parameter = solar_azimuth). The azimuth (in degrees) of the sun position vector from the reference photon bounce point position in the local ENU frame. The angle is measured from North and is positive towards East.

2.5.16 Solar Elevation

(parameter = solar_elevation). The elevation of the sun position vector from the reference photon bounce point position in the local ENU frame. The angle is measured from the East-North plane and is positive up.

2.5.17 Number of segment photons

(parameter = n_seg_ph). Number of photons in each land segment.

3

ALGORITHM METHODOLOGY

For the ecosystem community, identification of the ground and canopy surface is by far the most critical task, as meeting the science objective of determining global canopy heights hinges upon the ability to detect both the canopy surface and the underlying topography. Since a space-based photon counting laser mapping system is a relatively new instrument technology for mapping the Earth's surface, the software to accurately identify and extract both the canopy surface and ground surface is described here. The methodology adopted for ATL08 establishes a framework to potentially accept multiple approaches for capturing both the upper and lower surface of signal photons. One method used is an iterative filtering of photons in the along-track direction. This method has been found to preserve the topography and capture canopy photons, while rejecting noise photons. An advantage of this methodology is that it is self-parameterizing, robust, and works in all ecosystems if sufficient photons from both the canopy and ground are available. For processing purposes, along-track data signal photons are parsed into L-km segment of the orbit which is recommended to be 10 km in length.

3.1 Noise Filtering

Solar background noise is a significant challenge in the analysis of photon counting laser data. Range measurement data created from photon counting lidar detectors typically contain far higher noise levels than the more common photon integrating detectors available commercially in the presence of passive, solar background photons. Given the higher detection sensitivity for photon counting devices, a background photon has a greater probability of triggering a detection event over traditional integral measurements and may sometimes dominate the dataset. Solar background noise is a function of the surface reflectance, topography, solar elevation, and atmospheric conditions. Prior to running the surface finding algorithms used for ATL08 data products, the superset of output from the GSFC medium-high confidence classed photons (ATL03 signal_conf_ph: flags 3-4) and the output from DRAGANN will be considered as the input data set. ATL03 input data requirements include the latitude, longitude, height, segment delta time, segment ID, and a preliminary signal classification for each photon. The motivation behind combining the results from two different noise filtering methods is to ensure that all of the potential signal photons for land surfaces will be provided as input to the surface finding software. The description of the methodology for the ATL03 classification is described separately in the ATL03 ATBD. The methodology behind DRAGANN is described in the following section.

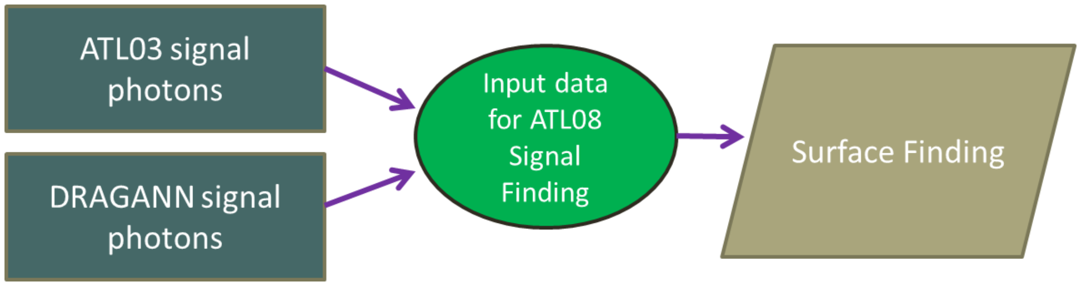

Figure 3.1. Combination of noise filtering algorithms to create a superset of input data for surface finding algorithms.

3.1.1 DRAGANN

The Differential, Regressive, and Gaussian Adaptive Nearest Neighbor (DRAGANN) filtering technique was developed to identify and remove noise photons from the photon counting data point cloud. DRAGANN utilizes the basic premise that signal photons will be closer in space than random noise photons. The first step of the filtering is to implement an adaptive nearest neighbor search. By using an adaptive method, different thresholds can be applied to account for variable amounts of background noise and changing surface reflectance along the data profile. This search finds an effective radius by computing the probability of finding P number of points within a search area. For MABEL and mATLAS, P=20 points within the search area was empirically derived but found to be an effective and efficient number of neighbors.

There may be cases, however, where the value of P needs to be changed. For example, during night acquisitions it is anticipated that the background noise rate will be considerably low. Since DRAGANN is searching for two distributions in neighborhood searching space, the software could incorrectly identify signal photons as noise photons. The parameter P, however, can be determined dynamically from estimations of the signal and noise rates from the photon cloud. In cases of low background noise (night), P would likely be changed to a value lower than 20. Similarly, in cases of high amounts of solar background, P may need to be increased to better capture the signal and avoid classifying small, dense clusters of noise as signal. In this case, however, it is likely that noise photons near signal photons will also be misclassified as signal. The method for dynamically determining a P value is explained further in section 4.2.1.

After P is defined, a histogram of the number of neighbors within a search radius for each point is generated. The distribution of neighbor radius occurrences is analyzed to determine the noise threshold.

Eqn. 3.1

where N_{total} is the total number of photons in the point cloud, V is the volume of the nearest neighborhood search, and V_{total} is the bounding volume of the enclosed point cloud. For a 2-dimensional data set, V becomes

Eqn. 3.2

where r is the radius. A good practice is to first normalize the data set along each dimension before running the DRAGANN filter. Normalization prevents the algorithm from favoring one dimension over the others in the radius search (e.g., when the latitude and longitude are in degrees and height is in meters).

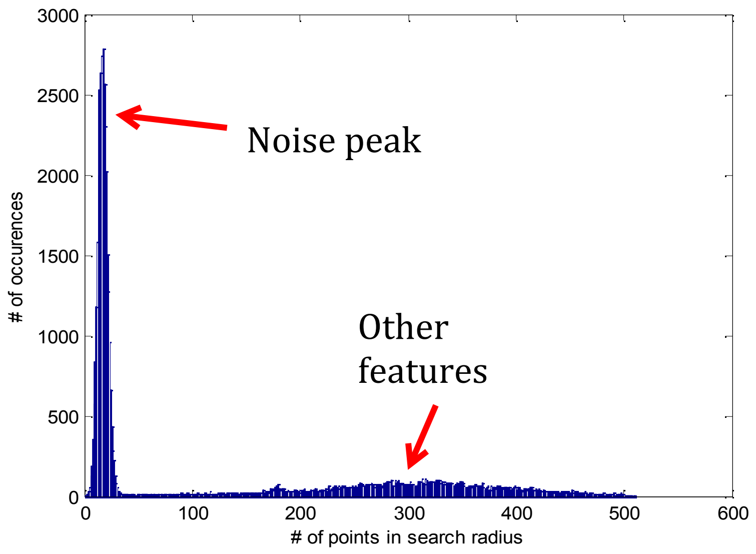

Figure 3.2. Histogram of the number of photons within a search radius. This histogram is used to determine the threshold for the DRAGANN approach.

Once the radius has been computed, DRAGANN counts the number of points within the radius for each point and histograms that set of values. The distribution of the number of points, Figure 3.2, reveals two distinct peaks; a noise peak and a signal peak. The motivation of DRAGANN is to isolate the signal photons by determining a threshold based on the number of photons within the search radius. The noise peak is characterized as having a large number of occurrences of photons with just a few neighboring photons within the search radius. The signal photons comprise the broad second peak. The first step in determining the threshold between the noise and signal is to implement Gaussian fitting to the number of photons distribution (i.e., the distribution shown in Figure 3.2). The Gaussian function has the form

Eqn. 3.3

where a is the amplitude of the peak, b is the center of the peak, and c is the standard deviation of the curve. A first derivative sign crossing method is one option to identify peaks within the distribution. As previously described, the noise peak is the leftmost peak in the distribution because noise is characterized as having fewer neighboring points. A Gaussian curve is fit to the noise peak which is subsequently removed from the distribution via subtraction. Next, up to ten Gaussian curves are fit to the histogram using an iterative process of fitting and subtracting the max-amplitude peak component from the histogram until all peaks have been extracted. Then, the potential Gaussians pass through a rejection process to eliminate those with poor statistical fits or other apparent errors (Goshtasby and O'Neill, 1994; Chauve et al. 2008). A Gaussian with an amplitude less than 1/5 of the previous Gaussian and within two standard deviations of the previous Gaussian should be rejected. Once the errant Gaussians are rejected, the noise and signal are separated based on the remaining two Gaussian components within the histogram using the logic that the leftmost Gaussian is noise (low neighbor counts) and the other is signal (high neighbor counts).

The intersection of these two Gaussians (noise and signal) determines a data threshold value. The threshold value is the parameter used to distinguish between noise points and signal points when the point cloud is re-evaluated for surface finding. In the event that only one curve passes the rejection process, the threshold is set at 4 above the center of the noise peak.

An example of the noise filtered product from DRAGANN is shown in Figure 3.3. The signal photons identified in this process will be combined with the coarse signal finding output available on the ATL03 data product.

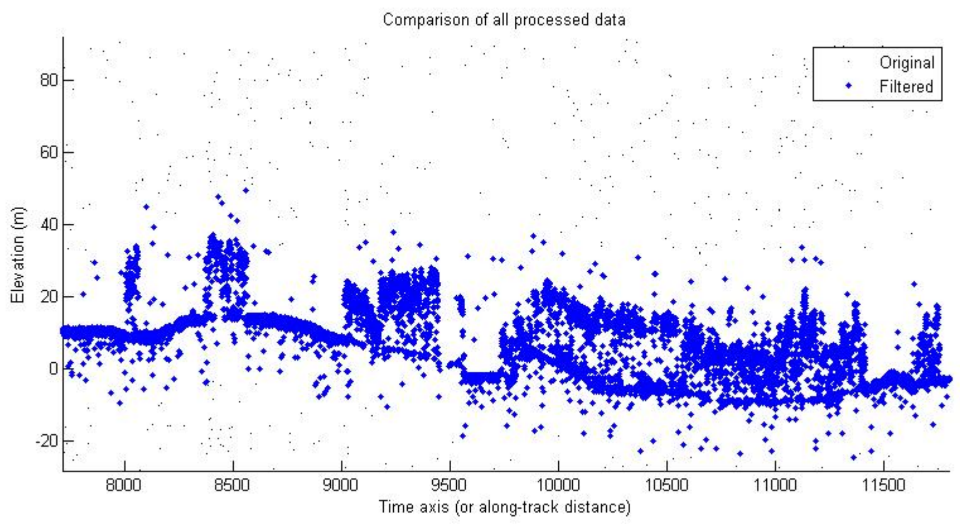

Figure 3.3. Output from DRAGANN filtering. Signal photons are shown as blue.

Figure 3.3 provides an example of along-track (profiling) height data collected in September 2012 from the MABEL (ICESat-2 simulator) over vegetation in North Carolina. The photons have been filtered such that the signal photons returned from vegetation and the ground surface are remaining. Noise photons that are adjacent to the signal photons are also retained in the input dataset; however, these should be classified as noise photons during the surface finding process. It is possible that some additional outlying noise may be retained during the DRAGANN process when noise photons are densely grouped, and these photons should be filtered out before the surface finding process. Estimates of the ground surface and canopy height can then be derived from the signal photons.

3.2 Surface Finding

Once the signal photons have been determined, the objective is to find the ground and canopy photons from within the point cloud. With the expectation that one algorithm may not work everywhere for all biomes, we are employing a framework that will allow us to combine the solutions of multiple algorithms into one final composite solution for the ground surface. The composite ground surface solution will then be utilized to classify the individual photons as ground, canopy, top of canopy, or noise. Currently, the framework described here utilizes one algorithm for finding the ground surface and canopy surface. Additional methods, however, could be integrated into the framework at a later time. Figure 3.4 below describes the framework.

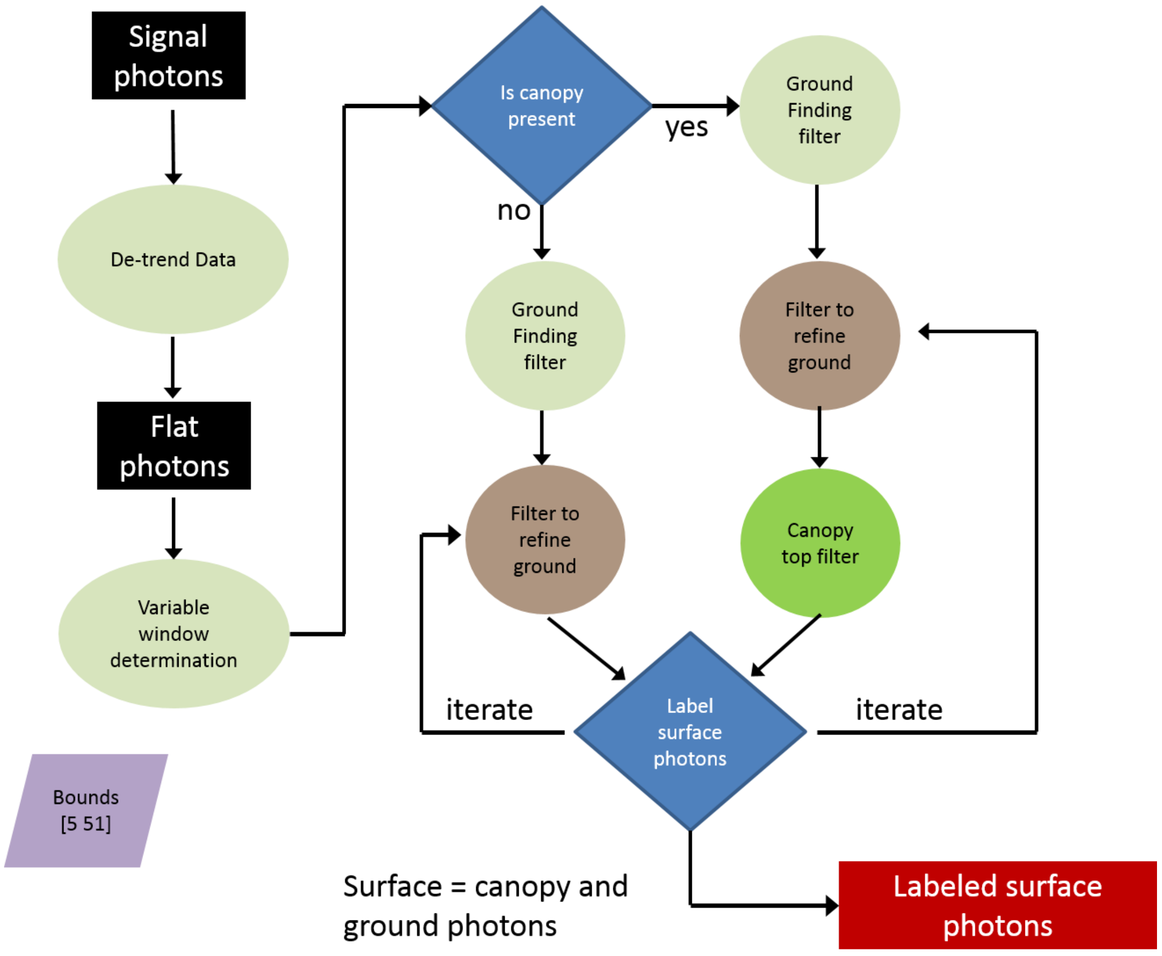

Figure 3.4. Flowchart of overall surface finding method.

3.2.1 De-trending the Signal Photons

An important step in the success of the surface finding algorithm is to remove the effect of topography on the input data, thus improving the performance of the algorithm. This is done by de-trending the input signal photons by subtracting a heavily smoothed “surface” that is derived from the input data. Essentially, this is a low pass filter of the original data and most of the analysis to detect the canopy and ground will subsequently be implemented on the high pass data. The amount of smoothing that is implemented in order to derive this first surface is dependent upon the relief. For segments where the relief is high, the smoothing window size is decreased so topography isn’t over-filtered.

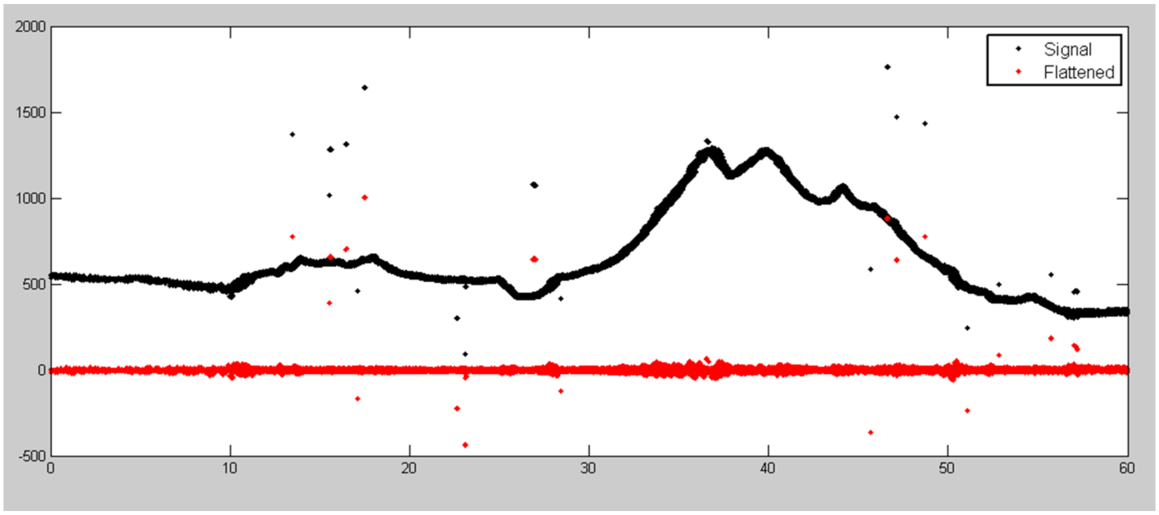

Figure 3.5. Plot of Signal Photons (black) from 2014 MABEL flight over Alaska and de-trended photons (red).

3.2.2 Canopy Determination

A key factor in the success of the surface finding algorithm is for the software to automatically **account for the presence of canopy** along a given *L*-km segment. Due to the large volume of data, this process has to occur in an automated fashion, allowing the correct methodology for extracting the surface to be applied to the data. In the absence of canopy, the iterative filtering approach to finding ground works extremely well, but if canopy does exist, we need to accommodate for that fact when we are trying to recover the ground surface.

Currently, the Landsat Tree Cover Continuous Fields dataset from the 2000 epoch is used to set a canopy flag within the ATL08 algorithm. Each of these Landsat Tree Cover tiles contain 30 m pixels indicating the percentage canopy cover for vegetation over 5 m high in that pixel area. The 2000 epoch is used over the newer 2005 epoch due to "striping" in the 2005 tiles, caused by the failure of the scan line corrector (SLC) in 2003. The striping artifacts result in inconsistent pixel values across a landscape which in turn can result in a tenfold difference in the average canopy cover percentage calculated between the epochs for a flight segment. There is currently available a 2015 Tree Cover Beta Release that utilizes Landsat 8 data. This new release of the 2015 Tree Cover product will replace the 2000 epoch for setting the canopy flag in the ATL08 algorithm. The Tree Cover data are available via ftp at http://glcf.umd.edu/data/landsatTreecover/.

For each *L-km* segment of ATLAS data, a comparison is made between the midpoint location of the segment and the midpoint locations of the WRS Landsat tiles to find the closest tile that encompasses the *L-km* segment. Using the closest found tile, each signal photon's X-Y location is used to identify the corresponding Landsat pixel. Multiple instances of the same pixels found for the *L-km* segment are discarded, and the percentage canopy values of the unique pixels determined to be under the *L-km* segment are averaged to produce an average canopy cover percentage for that segment. If the average canopy cover percentage for a segment is over 5% (threshold subject to change under further testing), then the ATL08 algorithm will assume the presence of canopy and identify both ground and vegetation photons in that segment's output. Else, the ATL08 algorithm uses a simplified calculation to identify only ground photons in that segment.

The canopy flag determines if the algorithm will calculate only ground photons (canopy flag = 0) or both ground and vegetation photons (canopy flag = 1) for each *L-km* segment.

For ATL08 product regions over Antarctica (regions 7, 8, 9, 10) and Greenland (region 11), the algorithm will assume only ground photons (canopy flag = 0) (see Figure 2.2).

3.2.3 Variable Window Determination

The method for generating a best estimated terrain surface will vary depending upon whether canopy is present. *L-km segment*s without canopy are much easier to analyze because the ground photons are usually continuous. *L-km* segments with canopy, however, require more scrutiny as the number of signal photons from ground are fewer due to occlusion by the vegetation.

There are some common elements for finding the terrain surface for both cases (canopy/no canopy) and with both methods. In both cases, we will use a variable windowing span to compute statistics as well as filter and smooth the data. For clarification, the window size is variable for each *L-km* segment, but it is constant within the *L-km* segment. For the surface finding algorithm, we will employ a Savitzky-Golay smoothing/median filtering method. Using this filter, we compute a variable smoothing parameter (or window size). It is important to bound the filter appropriately as the output from the median filter can lose fidelity if the scan is over-filtered.

We have developed an empirically-determined shape function, bound between [5 51], that sets the window size (Sspan) based on the number of photons within each *L-km* segment.

Eqn. 3.4

Eqn. 3.5

where a is the shape parameter and length is the total number of photons in the *L-km* segment. The shape parameter, a, was determined using data collected by MABEL and is shown in Figure 3.6. It is possible that the model of the shape function, or the filtering bounds, will need to be adjusted once ICESat-2/ATLAS is on orbit and collecting data.

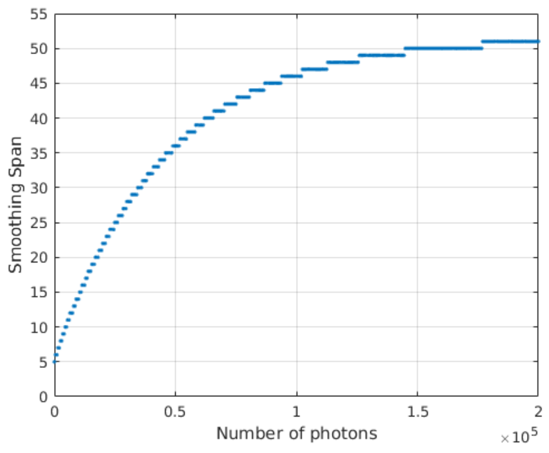

Figure 3.6. Shape Parameter for variable window size.

3.2.4 Compute descriptive statistics

To help characterize the input data and initialize some of the parameters used in the algorithm, we employ a moving window to compute descriptive statistics on the de-trended data. The moving window's width is the smoothing span function computed in Equation 5 and the window slides ¼ of its size to allow of overlap between windows. By moving the window with a large overlap helps to ensure that the approximate ground location is returned. The statistics computed for each window step include:

- Mean height
- Min height
- Max height
- Standard deviation of heights

Dependent upon the amount of vegetation within each window, the estimated ground height is estimated using different statistics. A standard deviation of the photon elevations computed within each moving window are used to classify the vertical spread of photons as belonging to one of four classes with increasing amounts of variation: open, canopy level 1, canopy level 2, canopy level 3. The canopy indices are defined in Table 3.1.

Table 3.1. Standard deviation ranges utilized to qualify the spread of photons within moving window.

Name	Definition	Lower Limit	Upper Limit
Open	Areas with little or no spread in signal photons determined due to low standard deviation	N/A	Photons falling within 1st quartile of Standard deviation
Canopy Level 1	Areas with small spread in signal photons	1st quartile	Median
Canopy Level 2	Areas with a medium amount of spread	Median	3rd quartile
Canopy Level 3	Areas with high amount of spread in signal photons	3rd quartile	N/A

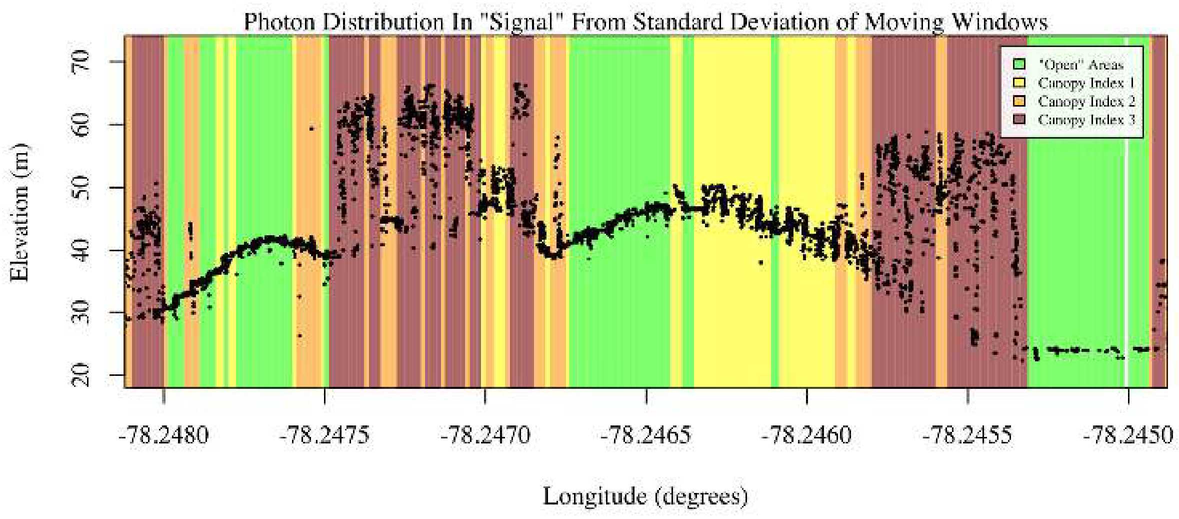

Figure 3.7. Illustration of the standard deviations calculated for each moving window to identify the amount of spread of signal photons within a given window.

3.2.5 Ground Finding Filter (Iterative median filtering)

A combination of an iterative median filtering and smoothing filter approach will be employed to derive the output solution of both the ground and canopy surfaces. The input to this process is the set of de-trended photons. Finding the ground in the presence of canopy often poses a challenge because often there are fewer ground photons underneath the canopy. The algorithm adopted here uses an iterative median filtering approach to retain/eliminate photons for ground finding in the presence of canopy. When canopy exists, a smoothed line will lay somewhere between the canopy top and the ground. This fact is used to iteratively label points above the smoothed line as canopy. The process is repeated four times to eliminate canopy points that fall above the estimated surface as well as noise points that fall below the ground surface. An example of iterative median filtering is shown in Figure 3.8. The final median filtered line is the preliminary surface estimate. A limitation of this approach, however, is in cases of dense vegetation and few photons reaching the ground surface. In these instances, the output of the median filter may lie within the canopy.

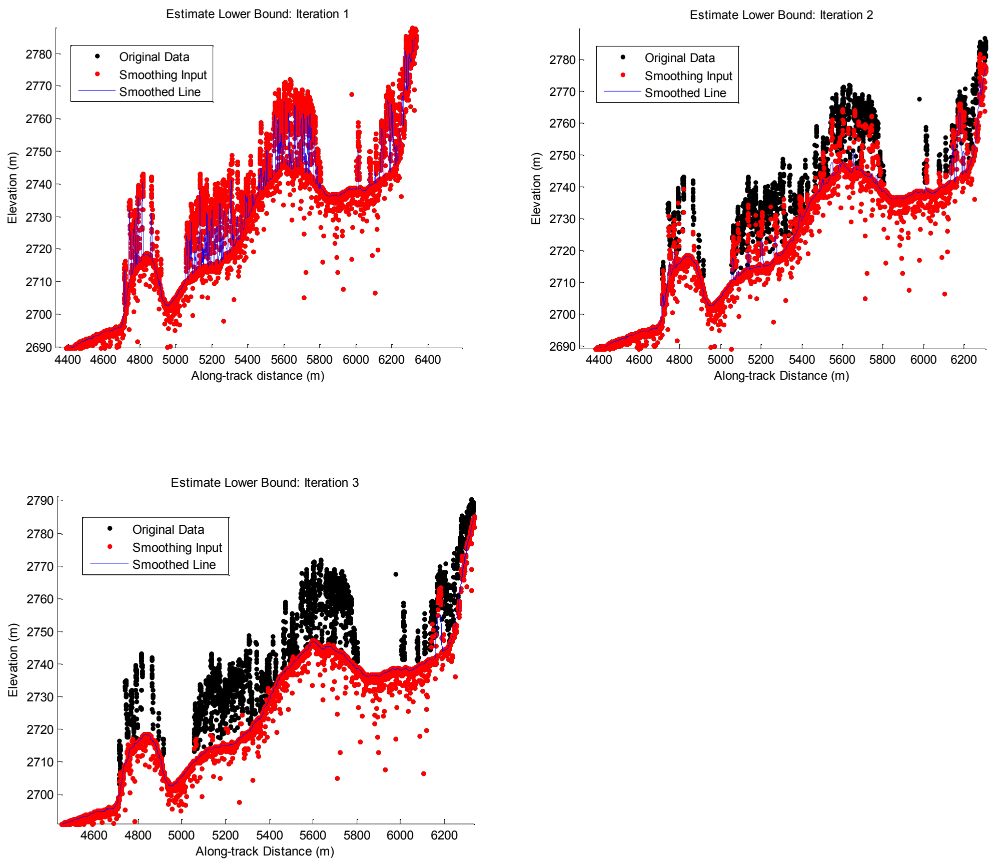

Figure 3.8. Three iterations of the ground finding concept for *L-km* segments with canopy.

3.3 Top of Canopy Finding Filter

Finding the top of the canopy surface uses the same methodology as finding the ground surface, except now the de-trended data are “flipped” over. The “flip” occurs by multiplying the photons heights by -1 and adding the mean of all the heights back to the data. The same procedure used to find the ground surface can be used to find the indices of the top of canopy points.

3.4 Classifying the Photons

Once a composite ground surface is determined, photons falling within the point spread function of the surface are labeled as ground photons. Based on the expected performance of ATLAS, the point spread function should be approximately 35 cm rms. Signal photons that are not labeled as ground and are below the ground surface (buffered with the point spread function) are considered noise, but keep the signal label.

The top of canopy photons that are identified can be used to generate an upper canopy surface through a spline function or other surface fitting methods. All signal photons that are not labeled ground and lie above the ground surface (buffered with the point spread function) and below the upper canopy surface are considered to be canopy photons (and thus labeled accordingly). Signal photons that lie above the top of canopy surface are considered noise, but keep the signal label.

FLAGS,	0 = noise
	1 = ground
	2 = canopy
	3 = TOC (top of canopy)

The final ground and canopy classifications are flags 1 – 3. The full canopy is the combination of flags 2 and 3.

3.5 Refining the Photon Labels

During the first iteration of the algorithm, it is possible that some photons are mislabeled; most likely this would be noise photons mislabeled as canopy. To reject these mislabeled photons, we apply three criteria:

a) If top of canopy photons are 2 standard deviations above a smoothed median top of canopy surface
b) If there are less than 3 canopy indices within a 15m radius

c) If, for 500 signal photon segments, the number of canopy photons is < 5% of the total (when SNR > 1), or < 10% of the total (when SNR <= 1). This minimum number of canopy indices criterion implies a minimum amount of canopy cover within a region.

There are also instances where the ground points will be redefined. This reassigning of ground points is based on how the final ground surface is determined. Following the "iterate" steps in the flowchart shown in Figure 3.4, if there are no canopy indices identified for the *L-km* segment, the final ground surface is interpolated from the identified ground photons and then will undergo a final round of median filtering and smoothing.

If canopy photons are identified, the final ground surface is interpolated based upon the level/amount of canopy at that location along the segment. The final ground surface is a composite of various intermediate ground surfaces, defined thusly:

ASmooth	heavily smoothed surface used to de-trend the signal data
Interp_Aground	interpolated ground surface based upon the identified ground photons
AgroundSmooth	median filtered and smoothed version of Interp_Aground

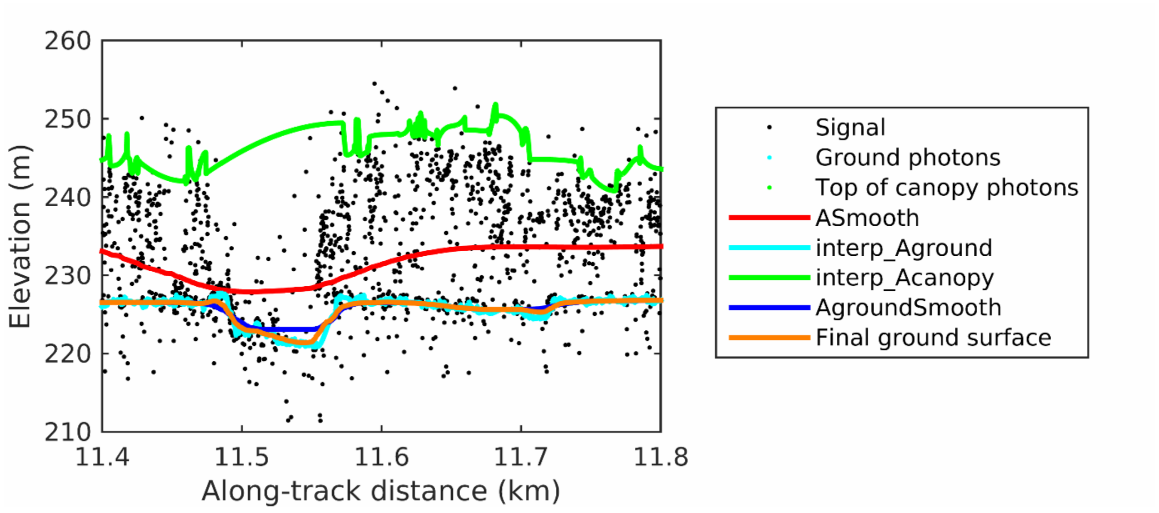

Figure 3.9. Example of the intermediate ground and top of canopy surfaces calculated from MABEL flight data over Alaska during July 2014.

During the first round of ground surface refinement, where there are canopy photons identified in the segment, the ground surface at that location is defined by the smoothed ground surface (AgroundSmooth) value. Else, if there is a location along-track where the standard deviation of the ground-only photons is greater than the 75% quartile for all signal photon standard deviations (i.e., canopy level 3), then the ground surface at that location is a weighted average between the interpolated ground surface (Interp_Aground*1/3) and the smoothed interpolated ground surface (AgroundSmooth*2/3). For all remaining locations long the segment, the ground surface is the average of the interpolated ground surface (Interp_Aground) and the heavily smoothed surface (Asmooth).

The second round of ground surface refinement is simpler than the first. Where there are canopy photons identified in the segment, the ground surface at that location is defined by the smoothed ground surface (AgroundSmooth) value again. For all other locations, the ground surface is defined by the interpolated ground surface (Interp_Aground). This composite ground surface is run through the median and smoothing filters again.

The pseudocode for this surface refining process can be found in section 4.10.

Examples of the ground and canopy photons for several MABEL lines are shown in Figures 3.10 – 3.12.

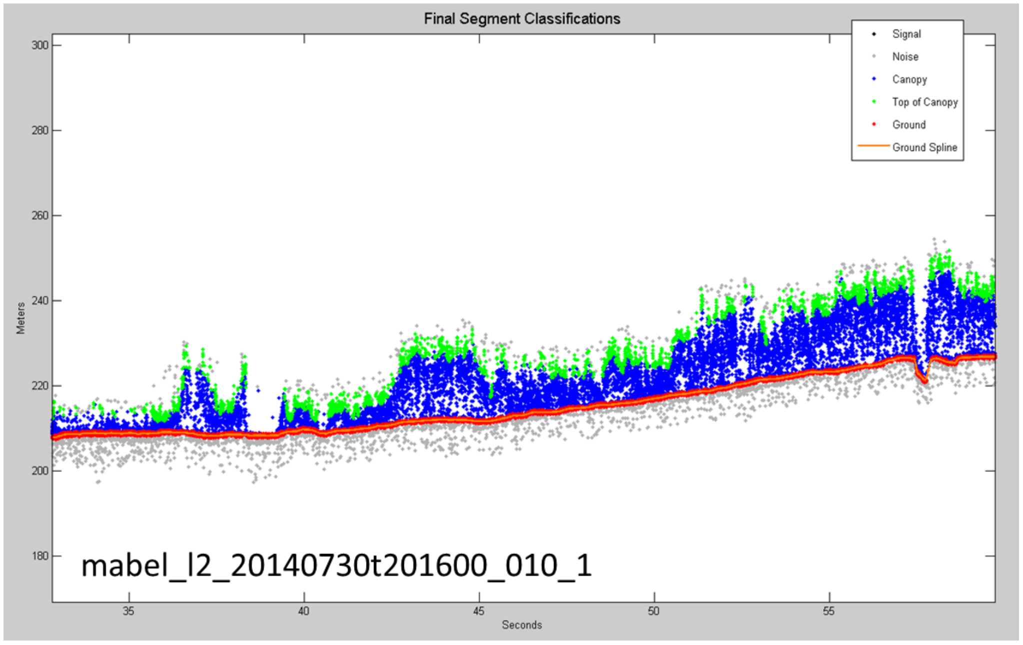

Figure 3.10. Example of classified photons from MABEL data collected in Alaska 2014. Red photons are photons classified as terrain. Green photons are classified as top of canopy. Canopy photons (shown as blue) are considered as photons lying between the terrain surface and top of canopy.

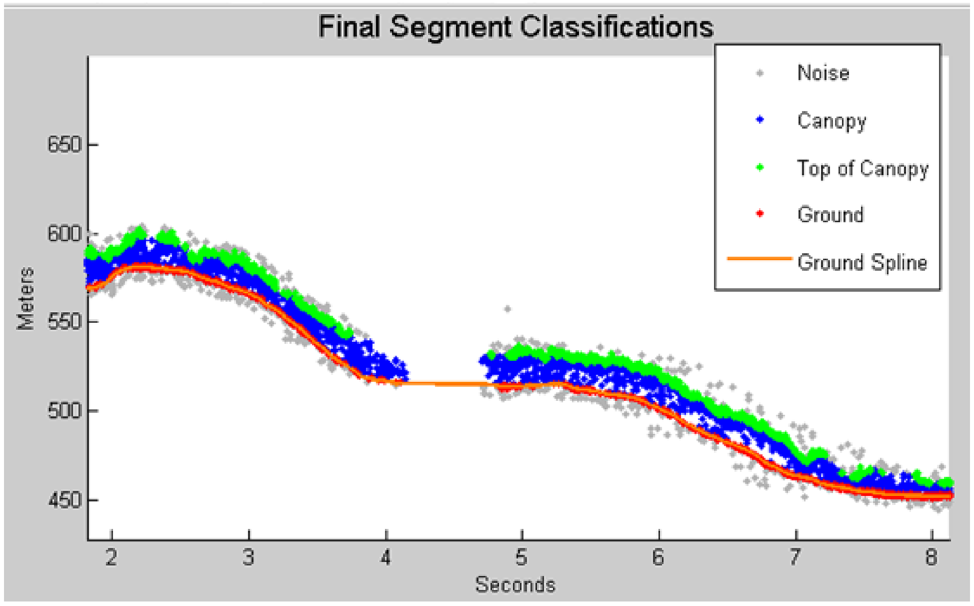

Figure 3.11. Example of classified photons from MABEL data collected in Alaska 2014. Red photons are photons classified as terrain. Green photons are classified as top of canopy. Canopy photons (shown as blue) are considered as photons lying between the terrain surface and top of canopy.

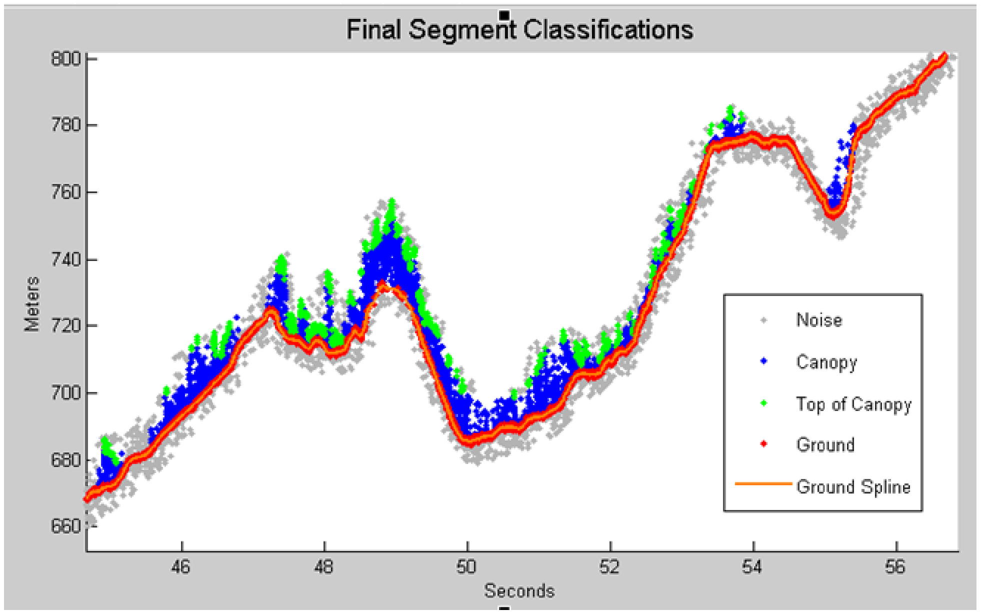

Figure 3.12. Example of classified photons from MABEL data collected in Alaska 2014. Red photons are photons classified as terrain. Green photons are classified as top of

canopy. Canopy photons (shown as blue) are considered as photons lying between the terrain surface and top of canopy.

3.6 Canopy Height Determination

Once a final ground surface is determined, canopy heights for individual photons are computed by removing the ground surface height for that photon's latitude/longitude. These relative canopy height values will be used to compute the canopy statistics on the ATL08 data product.

3.7 Link Scale for Data products

The link scale for each segment within which values for vegetation parameters will be derived will be defined over a fixed distance of 100 m. A fixed segment length ensures that canopy and terrain metrics are consistent between segments, in addition to increased ease of use of the final products. A size of 100 m was selected as it should provide approximately 140 photons (a statistically sufficient number) from which to make the calculations for terrain and canopy height.

4. ALGORITHM IMPLEMENTATION

Prior to running the surface finding algorithms used for ATL08 data products, the superset of output from the GSFC medium-high confidence classed photons (ATL03 signal_conf_ph: flags 3-4) and the output from DRAGANN will be considered as the input data set. ATL03 input data requirements include the latitude, longitude, height, and classification for each photon. The motivation behind combining the results from two different noise filtering methods is to ensure that all of the potential signal photons for land surfaces will be provided as input to the surface finding software.

Table 4.1. Input parameters to ATL08 classification algorithm.

Name	Data Type	Long Name	Units	Description	Source
delta_time	DOUBLE	GPS elapsed time	seconds	Elapsed GPS seconds since start of the granule for a given photon. Use the metadata attribute granule_start_seconds to compute full gps time.	ATL03
lat_ph	FLOAT	latitude of photon	degrees	Latitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point.	ATL03
lon_ph	FLOAT	longitude of photon	degrees	Longitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point.	ATL03
h_ph	FLOAT	height of photon	meters	Height of each received photon, relative to the WGS-84 ellipsoid.	ATL03
sigma_h	FLOAT	height uncertainty	m	Estimated height uncertainty (1-sigma) for the reference photon.	ATL03
signal_conf_ph	UINT_1_LE	photon signal confidence	counts	Confidence level associated with each photon event selected as signal (0-noise. 1- added to allow for buffer but algorithm classifies as background, 2-low, 3-med, 4-high).	ATL03
segment_id	UNIT_32	along-track	unitless	A seven-digit number	ATL03

		segment ID number		uniquely identifying each along-track segment. These are sequential, starting with one for the first segment after an ascending equatorial crossing node.	
Landsat tree cover	UINT_8	Landsat Tree Cover Continuous Fields	percentage	Percentage of woody vegetation greater than 5 meters in height across a 30 meter pixel	Global Land Cover Facility (Sexton , 2013)

Table 4.2. Additional external parameters referenced in ATL08 product.

Name	Data Type	Long Name	Units	Description	Source
atlas_pa				Off nadir pointing angle of the spacecraft	
ground_track				Ground track, as numbered from left to right: 1 = 1L, 2 = 1R, 3 = 2L, 4 = 2R, 5 = 3L, 6 = 3R	
dem_h				Reference DEM height	
ref_azimuth	FLOAT	azimuth	radians	Azimuth of the unit pointing vector for the reference photon in the local ENU frame in radians. The angle is measured from north and positive towards east.	ATL03
ref_elev	FLOAT	elevation	radians	Elevation of the unit pointing vector for the reference photon in the local ENU frame in radians. The angle is measured from east-north plane and positive towards up.	ATL03
rgt	INTEGER_2	reference ground track	unitless	The reference ground track (RGT) is the track on the Earth at which a specified unit vector within the observatory is pointed. Under nominal operating conditions, there will be no data collected along the RGT, as the RGT is spanned by GT2L and GT2R. During slews or off-pointing, it is	ATL03

				possible that ground tracks may intersect the RGT. The ICESat-2 mission has 1,387 RGTs.	
sigma_along	DOUBLE	along-track geolocation uncertainty	meters	Estimated Cartesian along-track uncertainty (1-sigma) for the reference photon.	ATL03
sigma_across	DOUBLE	across-track geolocation uncertainty	meters	Estimated Cartesian across-track uncertainty (1-sigma) for the reference photon.	ATL03
surf_type	INTEGER_1	surface type	unitless	Flags describing which surface types this interval is associated with. 0=not type, 1=is type. Order of array is land, ocean, sea ice, land ice, inland water.	ATL03, Section 4
cloud_flag_asr	Integer(3)	Cloud probability from ASR	unitless	Cloud confidence flag, from 0 to 5, indicating low, med, or high confidence of clear or cloudy sky	ATL09
msw_flag	Byte(3)	Multiple scattering warning flag	unitless	Flag with values from 0 to 5 indicating presence of multiple scattering, which may be due to blowing snow or cloud/aerosol layers.	ATL09
asr	Float(3)	Apparent surface reflectance	unitless	Surface reflectance as modified by atmospheric transmission	ATL09

4.1 Preparing data for input to ATL08 algorithm

1. Break up data into *L-km* segments. Segments equivalent of 10 km in along-track distance of an orbit would be appropriate.
2. Add a buffer of 200 m (or 10 segment_id's) to both ends of each *L-km* segment. The total processing segment length is (*L-km* + 2*buffer), but will be referred to as *L-km* segments for simplicity.
3. The input data for ATL08 algorithm is X, Y, Z, T (where T is time).
4. Determine a relative along-track distance, ATD, of each geolocated photon from the beginning of each *L-km* segment.

5. Normalize the lat/lon/height data from 0 – 1 for each *L-km* segment based on the min/max of each field. So, normlat = (lat-minlat)/(maxlat – minlat).
6. Build a kd-tree based on normalized Z and ATD.

4.1.1 Option for cloud flag based filtering

It is possible for the presence of clouds to affect the number of surface photon returns, or to cause false positive classifications of ground or canopy photons on low cloud returns. Either of these cases would lower the quality of the ATL08 product. Currently, ATL08 provides a cloud flag on its 100 m product and encourages the user to make note of the presence of clouds when using ATL08 output. However, if the presence of clouds is shown to significantly affect the ATL08 products once on-orbit data come in, an option to filter the ATL03 data before processing through the ATL08 algorithm is described here. **By default, this option should not be set; it is explained here in case on-orbit data show the need for it.**

1. The ATL09 cloud parameters of interest are solar_elevation, cloud_flag_asr, cloud_flag_atm, layer_con, and msw_flag. Interpolate these values to the ATL03 photon resolution.
2. Find the photons where solar_elevation > 0 to determine which photons are in daytime scenes.
3. For daytime scenes, the sky is considered clear if cloud_flag_asr < 3.
4. For nighttime scenes, the sky is considered clear if:
   ((cloud_flag_atm == 0) OR (cloud_flag_atm > 0 AND layer_con <= 10))
   AND
   ((msw_flag < 3) AND (msw_flag > -1))
5. The ATL03 photons which pass these clear sky criteria are then processed through the ATL08 algorithm.

4.2 Noise filtering via DRAGANN

DRAGANN will use ATL03 photons with all signal classification flags (0-4). These will include both signal and noise photons. See Appendix A for more details.

1. Determine the search radius starting with Equation 3.1. P=20, and V_{total} =1. N_{total} is the number of photons within the data *L-km* segment. Solve for V.
2. Now that you know V, determine the radius using Equation 3.2.
3. Compute the number of neighbors for each photon using this search radius.
4. Generate a histogram of the neighbor count distribution. As illustrated in Figure 3.2, the noise peak is the first peak (usually with the highest amplitude).
5. Compute the amplitude, a, of the first peak, which is located at position b.
6. Determine the width, c, of the first peak.
7. Use the amplitude and width to fit a Gaussian to the first peak of the histogram, as described in Equation 3.3. This fit is called Gaussian 1.
8. Remove Gaussian 1 from number count distribution.
9. Fit up to 10 Gaussians to the remaining distribution.
    a. Reject Gaussians that are too near (< 2 standard deviations) and amplitude too low (<1/5 previous amplitude) from the previous signal Gaussian.
10. Sort the Gaussians from largest to smallest area, estimated by a*c.
    a. Check in sorted order if one of the Gaussians are in the first 10% of the histogram. If so, it becomes the first Gaussian.
    b. Reject any Gaussians with imaginary components.
    c. Reject any Gaussians that are fully contained within another.
    d. Reject Gaussians whose centers are within 3 standard deviations of another, unless only two Gaussians remain
11. The last remaining Gaussian should represent the bulk of the signal photons within the distribution. This fit is referred to as Gaussian 2.
12. The intersection of Gaussian1 and Gaussian 2 is the threshold value.
    a. If the right edge of Gaussian 1 reaches near zero, defined as max(a)/100, for 5 consecutive bins, set that 5th bin location as the threshold instead of the intersection point.
    b. If there is only one Gaussian, it is assumed to be the noise Gaussian, and the threshold is set to b + c.

13. Label all photons having a neighbor count above the threshold as signal.
14. Label all photons having a neighbor count below the threshold as noise.
15. Reject noise photons.
16. Retain signal photons for feeding into next step of processing.
17. Use Logical OR to combine DRAGANN signal photons with ATL03 medium-high confidence signal photons (flags 3-4) as ATL08 signal photons.
18. Calculate a signal to noise ratio (SNR) for the *L-km* segment by dividing the number of ATL08 signal photons by the number of noise (i.e., all – signal) photons.

4.2.1 Alternative way of classifying signal with DRAGANN

If the default value of P=20 is found to not be sufficient, an alternative way of calculating P is described in this subsection. This assumes *L-km* to be 10 km (with additional *L-km* buffering).

1. Define a DRAGANN processing window of 170 segments (~3.4 km), and a buffer of 10 segments (~200 m).
2. The buffer is applied to both sides of each DRAGANN processing window to create buffered DRAGANN processing windows (referenced as “buffered window” for the rest of this section) that will overlap the DRAGANN processing windows next to them.
3. For each buffered window within the *L-km* segment, calculate a histogram of points with 1 m elevation bins.
4. For each buffered window histogram, calculate the median counts.
5. Bins with counts below the buffered window median count value are estimated to be noise. Calculate the **mean** count of noise bins.
6. Bins with counts above the buffered window median count value are estimated to be signal. Calculate the **mean** count of signal bins.
7. Determine the time elapsed over the buffered window.
8. Calculate estimated noise and signal rates for each buffered window by multiplying each window’s **mean** counts of noise bins and signal

bins, determined from steps 5 and 6 above, by 1/(elapsed time) to return the rates in terms of points/meter[elevation]/second[across].

9. Calculate a noise ratio for each window by dividing the noise rate by the signal rate.
10. If, for all the buffered windows in the *L-km* segment, the noise rate is less than 20 and the noise ratio is less than 0.15; OR any noise rate is 0; OR any signal rate is greater than 1000: re-calculate steps 3-9 using the entire *L-km* segment. **Continue with the following steps using results from the one *L-km* window (instead of multiple buffered windows).**
11. Now, determine the DRAGANN parameter, P, for each buffered window based on the following conditionals:
    a. **If the signal rate is NaN (i.e., an invalid value), set the signal index array to empty and move on to the next buffered window**.
    b. If noise rate < 20 || noise ratio < 0.15:

       P = signal rate

       If signal rate is < 5, P = 5; if signal rate > 20, P = 20
    c. Else P = 20.
12. Run DRAGANN on the buffered window points using the calculated P.
13. If DRAGANN fails to find a signal (i.e., only one Gaussian found), run DRAGANN again with P = 10.
14. If DRAGANN still fails to find a signal, try to determine P a second time using the following conditionals:
    a. If (noise rate >= 20) …

       && (signal rate > 100) …

       && (signal rate < 250),

       P = (signal rate)/2
    b. Else if signal rate >= 250,

       if noise rate >= 250,

       P = (noise rate)*1.1

```
        else,
          P = 250
    c. Else, P = mean(noise rate, signal rate)
```

15. Run DRAGANN on the buffered window points using the newly calculated P.
    a. If still no signal points are found, set a dragannError flag.
16. If signal points were found by DRAGANN, for each buffered window calculate a signal check by dividing the number of signal points found via DRAGANN by the number of total points in the buffered window.
17. **If dragannError has been set, or there are suspect signal statistics, the following snippet of pseudocode will check those conditionals and try to iteratively find a better P value to run DRAGANN with:**

```
try_count = 0

While dragannError ...
|| (  (noise rate >= 30) ...
    && (signal check > noise ratio) ...
    && (noise ratio >= 0.15) ) ...
|| (signal check < 0.001):

  if P < 3,
    break
  else,
    P = P*0.75
  end

  if  try_count < 2
      Clear out signal index results from previous DRAGANN run
      Re-run DRAGANN with new P value
      Recalculate the signal check
  end

  if no signal index results are returned
      P = P*0.75
  end

  try_count = try_count + 1

end
```

18. If no signal photons are found by DRAGANN because only one Gaussian was found, set the threshold as b+c (i.e., one standard deviation away from the Gaussian peak location) **for a final DRAGANN run**. Otherwise, set the signal index array to empty and move on to the next buffered window.
19. Assign the signal values found from DRAGANN for each buffered window to the original DRAGANN processing window range of points.
20. Combine signal points from each DRAGANN processing window back into one *L-km* array of signal points for further processing.

4.2.2 Iterative DRAGANN processing

It is possible in processing segments with high noise rates that DRAGANN will incorrectly identify clusters of noise as signal. One way to reduce these false positive noise clusters is to run the alternative DRAGANN process (Sec 4.2.1) again with the input being the signal output photons from the first run through alternative DRAGANN. Note that this methodology is still being tested, so by default this option should not be set.

1. If SNR < 1 (TBD) from alternative DRAGANN run, run alternative DRAGANN process again using the output signal photons from first DRAGANN run as the input to the second DRAGANN run.
2. Recalculate SNR based on output of second DRAGANN run.

4.3 Is Canopy Present

1. If *L-km* segment is within an ATL08 region encompassing Antarctica (regions 7, 8, 9, 10) or Greenland (region 11), assume no canopy is present: canopy flag = 0. Else:
2. Determine the center Latitude/Longitude position for the *L-km* segment.

3. Determine the corresponding tile from the Landsat continuous cover product.
4. For each unique XY position in the ATLAS segment, extract the canopy cover value from the Landsat CC product
5. Compute the average canopy cover value for the L-km segment (based on the Landsat values).
6. If canopy cover > 5%, set canopy flag = 1 (assumes canopy is present)
7. If canopy cover <= 5%, set canopy flag = 0 (assumes no canopy is present)

4.4 Compute Filtering Window

1. Next step is to run a surface filter with a variable window size (variable in that it will change from *L-km* segment to *L-km* segment). The window-size is denoted as Window.
2. , where *length* is the number of photons in the segment.
3. , where *a* is the shape parameter for the window span.

4.5 De-trend Data

1. The input data are the signal photons identified by DRAGANN and the ATL03 classification (signal_conf_ph) values of 3-4.
2. Generate a rough surface by connecting all unique (time) photons to each other. Let's call this surface interp_A.
3. Run a median filter through interp_A using the window size set by the software. Output = Asmooth.
4. Define a reference DEM limit (ref_dem_limit) as 120 m (TBD).
5. Remove any Asmooth values further than the ref_dem_limit threshold from the reference DEM, and interpolate the Asmooth surface based on the remaining Asmooth values.

6. Compute the approximate relief of the *L-km* segment using the 95th - 5th percentile heights of the signal photons. We are going to filter Asmooth again and the smoothing is a function of the relief.
7. Define the SmoothSize using the conditional statements below. The SmoothSize will be used to detrend the data as well as to create an interpolated ground surface later.

   SmoothSize = 2 * Window

   - If relief>=900, SmoothSize= round(SmoothSize/4)
   - If relief>=400 && <=900, SmoothSize=round(SmoothSize/3)
   - If relief>=200 && <=400, SmoothSize=round(SmoothSize/2)
8. Greatly smooth Asmooth by first running Asmooth 10 times through a median filter then a smoothing filter with a moving average method on the result. Both the median filter and the smoothing filter use a window size of SmoothSize.

4.6 Filter outlier noise from signal

1. If there are any signal data that are 150 meters above Asmooth, remove them from the signal data set.
2. If the standard deviation of the detrended signal is greater than 10 meters, remove any signal value from the signal data set that is 2 times the standard deviation of the detrended signal below Asmooth.
3. Calculate a new Asmooth surface by interpolating a surface from the remaining signal photons and median filtering using the Window size, then median filter and smooth 10 times again using the SmoothSize.
4. Detrend the signal photons by subtracting the signal height values from the Asmooth surface height values. Use the detrended heights for surface finding.

4.7 *Finding the initial ground estimate*

1. At this point, the initial signal photons have been noise filtered and de-trended and should have the following format: X, Y, detrended Z, T (T=time). From this, the input data into the ground finding will be the ATD (along track distance) metric (such as time) and the detrended Z height values.
2. Define a medianSpan as Window*2/3.
3. Identifying the ground surface is an iterative process. Start by assuming that all the input signal height photons are the ground. The first goal is the cut out the lower height excess photons in order to find a lower bound for potential ground photons. This process is done 5 times and an offset of 4 meters is subtracted from the resulting lower bound. The smoothing filter uses a moving average again:

```
for j=1:5
        cutOff = median filter (ground, medianSpan)
        cutOff = smooth filter (cutOff, Window)
        ground = ground( (cutOff – ground) > -1 )
    end
    lowerbound = median filter (ground, medianSpan*3)
    middlebound = smooth filter (lowerbound, Window)
    lowerbound = smooth filter (lowerbound, Window) – 4
end;
```

4. Create an interpolated surface along the lower bound points and only keep input photons above that line as potential ground points:

```
top = input( input > interp(lowerbound) )
```

5. The next goal is to cut out excess higher elevation photons in order to find an upper bound to the ground photons. This process is done 3 times and an offset of 1 meter is added to the resulting upper bound:

```
for j = 1:3
```

```
        cutOff = median filter (top, medianSpan)
        cutOff = smooth filter (cutOff, Window)
        top  = top( (cutOff – top) > -1 )
    end
    upperbound = median filter (top, medianSpan)
    upperbound = smooth filter (upperbound, Window) + 1
```

6. Create an interpolated surface along the upper bound points and extract the points between the upper and lower bounds as potential ground points:

```
    ground = input( ( input > interp(lowerbound) ) & …
                    ( input < interp(upperbound) ) )
```

7. Refine the extracted ground points to cut out more canopy:

```
    For j = 1:2
            cutOff = median filter (ground, medianSpan)
            cutOff = smooth filter (cutOff, Window)
            ground = ground( (cutOff – ground) > -1 )
    end
```

8. Run the ground output once more through a median filter using window side medianSpan and a smoothing filter using window size Window, but this time with the Savitzky-Golay method.
9. Finally, linearly interpolate a surface from the ground points.
10. The first estimate of canopy points are those indices of points that are between 2 and 150 meters above the estimated ground surface. Save these indices for the next section on finding the top of canopy.
11. The output from the final iteration of ground points is temp_interpA – an interpolated ground estimate.
12. Find ground indices that lie within +/- 0.5 m of temp_interpA.
13. Apply the ground indices to the original heights (i.e., not the de-trended data) to label ground photons.

14. Interpolate a ground surface based on the ground photons. Output is interp_Aground.

4.8 Find the top of the canopy (if canopy_flag = 1)

1. The input are the ATD metric (i.e., time), and the de-trended Z values indexed by the canopy indices extracted from step 4.7(10).
2. Flip this data over so that we can find a canopy "surface" by multiplying the de-trended canopy heights by -1.0 and adding the mean(heights).
3. Finding the top of canopy is also an iterative process. Follow the same steps described in 4.7(2) – 4.7(9), but use the canopy indexed and flipped Z values in place of the ground input.
4. Final retained photons are considered top of canopy photons. Use the indices of these photons to define top of canopy photons in the original (not de-trended) Z values.
5. Build a kd-tree on canopy indices.
6. If there are less than three canopy indices within a 15m radius, reassign these photons to noise photons.

4.9 Compute statistics on de-trended data

1. The input data have been noise filtered and de-trended and should have the following input format: X, Y, detrended Z, T.
2. The input data will contain signal photons as well as a few noise photons near the surface.
3. Compute statistics of heights in the along-track direction using a sliding window. Using the window size (window), compute height statistics for all photons that fall within each window. These include max height, median height, mean height, min height, and standard deviation of all photon heights. Additionally, in each window compute the median height and standard

deviation of just the initially classified top of canopy photons, and the standard deviation of just the initially classified ground photon heights. Currently only the median top of canopy, and all STD variables are being utilized, but it's possible that other statistics may be incorporated as changes/improvements are made to the code.

4. Slide the window ¼ of the window span and recompute statistics along the entire *L-km* segment. This results in one value for each statistic for each window.
5. Determine canopy index categories for each window based upon the total distribution of STD values for all signal photons along the *L-km* segment based on STD quartiles.
6. Open canopy have STD values falling within the 1st quartile.
7. Canopy Level 1 has STD values falling from 1st quartile to median STD value.
8. Canopy Level 2 has STD values falling from median STD value to 3rd quartile.
9. Canopy Level 3 has STD values falling from 3rd quartile to max STD.
10. Interpolate the window STD values (both for all photons and ground-only photons) back to the native along-track resolution and calculate the interpolated all-photon STD quartiles to create an interpolated canopy level index. This will be used later for interpolating a ground surface.

4.10 Refine Ground Estimates

1. Smooth the interpolated ground surface 10 times:

```
For j= 1:10
        AgroundSmooth = median filter (interp_Aground, SmoothSize*5)
        AgroundSmooth = smooth filter (AgroundSmooth, SmoothSize)
End
```

2. This output (AgroundSmooth) from the filtering/smoothing function is an intermediate ground solution and it will be used to estimate the final solution.
3. If there are **no canopy indices** identified along the entire segment (OR canopy_flag = 0) AND relief >400 m

```
        FINALGROUND = median filter (Asmooth, SmoothSize)
        FINALGROUND = smooth filter (FINALGROUND, SmoothSize)
    Else
        FINALGROUND = AgroundSmooth
    end
```

4. If there are **canopy indices** identified along the segment:

   If there is a canopy photon identified at a location along-track above the ground surface, then at that location along-track

```
        FINALGROUND = AgroundSmooth
 else if there is a location along-track where the interpolated ground STD has
an interpolated canopy level>=3
        FINALGROUND = Interp_Aground*1/3 + AgroundSmooth*2/3

else

        FINALGROUND = Interp_Aground*1/2 + Asmooth*1/2

end
```

5. Smooth the resulting interpolated ground surface (FINALGROUND) once using a median filter with window size of SmoothSize, then a smoothing filter twice with window size of SmoothSize. Select ground photons that lie within the point spread function (PSF) of FINALGROUND.
6. PSF is determined by sigma_atlas_land (Eq. 1.2) calculated at the photon resolution and thresholded between 0.5 to 1 m.
   a. Estimate the terrain slope by taking the gradient of FINALGROUND.
   b. Interpolate the sigma_h values to the photon resolution.
   c. Calculate sigma_topo (Eq. 1.3) at the photon resolution.

   d. Calculate sigma_atlas_land at the photon resolution using the sigma_h and sigma_topo values at the photon resolution.
   e. Set PSF equal to sigma_atlas_land.
      i. Any PSF < 0.5 m is set to 0.5 m as the minimum PSF.
      ii. Any PSF > 1 m is set to 1 m as the maximum PSF. Set psf_flag to true.

4.11 Canopy Photon Filtering

1. The first canopy filter will remove photons classified as top of canopy that are significantly above a smoothed median top of canopy surface. To calculate the smoothed median top of canopy surface:
   a. Interpolate the median and standard deviation canopy window statistics, calculated from 4.9 (3), to the top of canopy photon resolution. Output variables: interpMedianC, interpStdC.
   b. Calculate a canopy window size using Eq. 3.4, where *length* = number of top of canopy photons. Output variable: winC.
   c. Create the median filtered and smoothed top of canopy surface, smoothedC:

      smoothedC = median filter (interpMedianC, winC)

      if SNR > 1, canopySmoothSpan = winC*2;
      else, canopySmoothSpan = smoothSpan;

      smoothedC = smooth filter (smoothedC, canopySmoothSpan)

   d. Add the detrended heights back into the smoothedC surface:

      smoothedC = smoothedC + Asmooth

2. Set canopy height thresholds based on the interpolated top of canopy STD:

If SNR > 1, canopySTDthresh = 3; else, canopySTDthresh = 2;
canopy_height_thresh = canopySTDthresh*interpStdC

high_cStd = canopy_height_thresh > 10

low_cStd = canopy_height_thresh < 3

canopy_height_thresh(high_cStd) =
canopy_height_thresh(high_cStd)/2

canopy_height_thresh(low_cStd) = 3

3. Relabel as noise any top of canopy photons that are higher than smoothedC + canopy_height_thresh.
4. Next, interpolate a canopy spline using the remaining top of canopy photons (here we are trying to create an upper bound on canopy points). The interpolation method used is the shape-preserving piecewise cubic interpolation. This output is named interp_Acanopy.
5. Photons falling below interp_Acanopy and above FINALGROUND+PSF are labeled as canopy points.
6. For 500 signal photon segments, if number of all canopy photons (i.e., canopy and top of canopy) is:

   < 5% of the total (when SNR > 1), OR

   < 10% of the total (when SNR <= 1),

   relabel the canopy photons as noise.
7. Interpolate a new canopy spline using the filtered **top of canopy** photons. This output is again named interp_Acanopy.
8. Again, label photons that lie between interp_Acanopy and FINALGROUND+PSF as canopy photons.
9. Since the canopy points have been relabeled, we need to do a final refinement of the ground surface:

   If canopy is present at any location along-track

FINALGROUND = AgroundSmooth (at that location)

Else if canopy is not present at a location along-track

FINALGROUND = interp_Aground

Smooth the resulting interpolated ground surface (FINALGROUND) once using a median filter with window size of SmoothSize, then a smoothing filter twice with window size of SmoothSize.

10. Relabel ground photons based on this new (and last) FINALGROUND solution +/- a recalculated PSF. Points falling below the buffer are labeled as noise.
11. Using Interp_Acanopy and this last FINALGROUND solution + PSF buffer, label all photons that lie between the two as canopy photons.
12. Repeat the canopy cover filtering: For 500 signal photon segments, if number of all canopy photons (i.e., canopy and top of canopy) is:

    < 5% of the total (when SNR > 1), OR

    < 10% of the total (when SNR <= 1),

    relabel the canopy photons as noise. This is the last canopy labeling step.

4.12 Compute individual Canopy Heights

1. At this point, each photon will have its final label assigned in **classed_pc_flag**: 0 = noise, 1 = ground, 2 = canopy, 3 = top of canopy.
2. For each individual photon labeled as canopy or top of canopy, subtract the Z height value from the interpolated terrain surface, FINALGROUND, at that particular position in the along-track direction.
3. The relative height for each individual canopy or top of canopy photon will be used to calculate canopy products described in Section 4.15. Additional canopy products will be calculated using the absolute heights, as described in Section 4.15.1.

4.13 Final photon classification QA check

1. Find any ground, canopy, or top of canopy photons that have elevations further than the ref_dem_limit from the reference DEM elevation value. Convert these to the noise classification.
2. Find any relative heights of canopy or top of canopy photons that are greater than 150 m above the interpolated ground surface, FINALGROUND. Convert these to the noise classification.
3. Find any FINALGROUND elevations that are further than the ref_dem_limit from the reference DEM elevation value. Convert those FINALGROUND elevations to an invalid value, and convert any classified photons at the same indices to noise.
4. If more than 50% of photons are removed in a segment, set ph_removal_flag to true.

4.14 Compute segment parameters for the Land Products

1. For each 100 m segment, determine the classed photons (photons classified as ground, canopy, or top of canopy) and extract the ground photons.
2. If the number of ground photons > 5% of the total number of classed photons within the segment (this control value of 5% can be modified once on orbit):
    a. Compute statistics on the ground photons: mean, median, min, max, standard deviation, mode, and skew. These heights will be reported on the product as **h_te_mean, h_te_median, h_te_min, h_te_max, h_te_mode**, and **h_te_skew** respectively described in Table 2.1.
    b. Compute the standard deviation of the ground photons about the interpolated terrain surface, FINALGROUND. This value is reported as **h_te_std** in Table 2.1.
    c. Compute the residuals of the ground photon Z heights about the interpolated terrain surface, FINALGROUND. The product is the root

sum of squares of the ground photon residuals combined with the **sigma_ATLAS_LAND_h** term in Table 2.5 normalized by the number of ground photons in the segment as described in Equation 1.4. This parameter reported as **h_te_uncertainty** in Table 2.1.

d. Compute a linear fit on the ground photons and report the slope. This parameter is **terrain_slope** in Table 2.1.
    i. If there are fewer than two ground photons in the segment, or if the number of ground photons in the segment <= 5% of total number of classified photons in the segment, and there are at least two classified (ground or canopy) photons, compute terrain_slope via a linear fit of the interpolated ground surface, FINALGROUND, instead of the ground photons.

e. Calculate a best fit terrain elevation at the mid-point location of the 100 m segment:
    i. Calculate each terrain photon's distance along-track into the 100 m segment using the corresponding ATL03 20 m products segment_length and dist_ph_along, and determine the mid-segment distance (expected to be 50 m ± 0.5 m).
        1. Use the mid-segment distance to interpolate a mid-segment time (**delta_time** in Table 2.4). Use the mid-segment time to interpolate other mid-segment parameters: interpolated terrain surface, FINALGROUND, as **h_te_interp** (Table 2.1); **latitude** and **longitude** (Table 2.4).

    ii. For the linear fit, use terrain_slope to apply a slope correction to each terrain photon, and use a linear weighting on each photon based on its distance to the mid-segment location: 1 / sqrt((photon distance along – mid-segment distance)^2).

    iii. Calculate a 3rd and 4th order polynomial fit to the terrain photons in the segment.

iv. Determine which of the three fits is best by calculating the mean and standard deviation of the fit errors. If one of the fits has both the smallest mean and standard deviations, use that fit. Else, use the fit with the smallest standard deviation. If more than one fit has the same smallest mean and/or standard deviation, use the fit with the higher polynomial.

v. Using the best fit, interpolate a mid-segment elevation. This parameter is **h_te_best_fit** in Table 2.1.

1. If h_te_best_fit is farther than 3 m from h_te_interp (best fit diff threshold), check if: there are terrain photons on both sides of the mid-segment location; or the mid-segment elevation of the linear fit is greater than the best fit diff threshold; or the number of ground photons in the segment <= 5% of total number of classified photons per segment. If any of those cases are present, use h_te_interp as the corrected h_te_best_fit. Otherwise use the linear fit as the corrected h_te_best_fit.

f. Compute the difference of the median ground height from the reference DTM height. This parameter is **h_dif_ref** in Table 2.4.

3. If the number of ground photons in the segment <= 5% of total number of classified photons per segment,
   a. Report an invalid value for all computed terrain products: **h_te_mean, h_te_median, h_te_min, h_te_max, h_te_mode**, **h_te_skew, h_te_roughness, h_te_uncertainty, and h_te_slope** respectively as described in Table 2.1.
   b. Report the interpolated terrain surface, FinalGround, as **h_te_interp** as described in Table 2.1, and report **h_te_best_fit** as the h_te_interp value.

4.15 Compute segment parameters for the Canopy Products

1. For each 100 m segment, determine the classed photons (photons classified as ground, canopy, or top of canopy) and extract all canopy photons (i.e., canopy and top of canopy; henceforth referred to as "canopy" unless otherwise noted).
2. Only compute canopy height products if the number of canopy photons is > 5% of the total number of classed photons within the segment (this control value of 5% can be modified once on orbit), and the number of ground photons is > 5%, or else report an invalid value for each canopy height variable.
3. Again, the relative heights (height above the interpolated ground surface, FINALGROUND) have been computed already. All parameters derived in the section are based on relative heights.
4. Sort the heights and compute a cumulative distribution of the heights. Select the height associated with the 95% maximum height. This value is **h_canopy** listed in Table 2.2.
5. Compute statistics on the relative canopy heights. Min, Mean, Median, Max and standard deviation. These values are reported on the product as **h_min_canopy, h_mean_canopy, h_median_canopy, h_max_canopy**, and **canopy_openness** respectively in Table 2.2.
6. Compute the difference between h_canopy and h_median_canopy. This parameter is **h_dif_canopy** reported in Table 2.2 and represents an amount of canopy depth.
7. Again, using the cumulative distribution of relative canopy heights, select the heights associated with the **canopy_h_metrics** percentile distributions (25, 50, 60, 70, 75, 80, 85, 90, 95, 99), and report as listed in Table 2.2.
8. Using the h_canopy height value computed in step 4, compute height bins as h_canopy/4. Next compute the number of heights (or photons) that fall within each height bin. Divide these number of photons by the total number of canopy photons for the segment. These percentages are the density values and reported as **canopy_d_quartile**, as described in Table 2.2.

9. Compute the standard deviation of all photons that were labeled as Top of Canopy (flag 3) in the photon labeling portion. This value is reported on the data product as **toc_roughness** listed in Table 2.2.
10. The quadratic mean height, **h_canopy_quad** is computed by

    where N_{ca} is the number of canopy photons in the segment and h_i are the individual canopy heights.
11. The **canopy_closure** parameter in Table 2.2 is computed by

4.15.1 Canopy Products calculated with absolute heights

1. The absolute canopy height products are calculated if the number of canopy photons is > 5% of the total number of classed photons within the segment. No number of ground photons threshold is applied for these.
2. The **centroid_height** parameter in Table 2.2 is represented by all the classed photons for the segment (canopy & ground). To determine the centroid height, compute a cumulative distribution of all absolute classified heights and select the median height.
3. Compute statistics on the absolute canopy heights: Min, Mean, Median, and Max. These values are reported on the product as **h_min_canopy_abs, h_mean_canopy_abs, h_median_canopy_abs,** and **h_max_canopy_abs**, respectively, as described in Table 2.2.
4. Again, using the cumulative distribution of absolute canopy heights, select the heights associated with the **canopy_h_metrics** percentile distributions (25, 50, 60, 70, 75, 80, 85, 90, 95, 99), and report as listed in Table 2.2.

4.16 Record final product without buffer

1. Now that all products have be determined via processing of the *L-km* segment with the buffer included, remove the products that lie within the buffer zone on each end of the *L-km* segment.
2. Record the final *L-km* products and move on to process the next *L-km* segment.

5

DATA PRODUCT VALIDATION STRATEGY

Although there are no Level-1 requirements related to the accuracy and precision of the ATL08 data products, we are presenting a methodology for validating terrain height, canopy height, and canopy cover once ATL08 data products are created. Parameters for the terrain and canopy will be provided at a fixed size of 100 m along the ground track referred to as a segment. Validation of the data parameters should occur at the 100 m segment scale and residuals of uncertainties are quantified (i.e. averaged) at the 5-km scale. This 5-km length scale will allow for quantification of errors and uncertainties at a local scale which should reflect uncertainties as a function of surface type and topography.

5.1 Validation Data

Swath mapping airborne lidar is the preferred source of validation data for the ICESat-2 mission due to the fact that it is widely available and the errors associated with most small-footprint, discrete return data sets are well understood and quantified. Profiling airborne lidar systems (such as MABEL) are more challenging to use for validation due to the low probability of exact overlap of flightlines between two profiling systems (e.g. ICESat-2 and MABEL). In order for the ICESat-2 validation exercise to be statistically relevant, the airborne data should meet the requirements listed in Table 5.1. Validation data sets should preferably have a minimum average point density of 5 pts/m^2. In some instances, however, validation data sets with a lower point density that still meet the requirements in Table 5.1 may be utilized for validation to provide sufficient spatial coverage.

Table 5.1. Airborne Lidar Data Vertical Height (Z accuracy) Requirements for validation data

ICESat-2 ATL08 Parameter	Airborne lidar (rms)
Terrain height	<0.3 m over open ground (vertical) <0.5 m (horizontal)

Canopy height	<2 m temperate forest, < 3 m tropical forest
Canopy cover	n/a

Terrain and canopy heights will be validated by computing the residuals between the ATL08 terrain and canopy height value, respectively, for a given 100 m segment and the terrain height (or canopy height) of the validation data for that same representative distance. Canopy cover on the ATL08 data product shall be validated by computing the relative canopy cover (cc = canopy returns/total returns) for the same representative distance in the airborne lidar data.

It is recommended that the validation process include the use of ancillary data sets (i.e. Landsat-derived annual forest change maps) to ensure that the validation results are not errantly biased due to non-equivalent content between the data sets.

Using a synergistic approach, we present two options for acquiring the required validation airborne lidar data sets.

Option 1:
We will identify and utilize freely available, open source airborne lidar data as the validation data. Potential repositories of this data include OpenTopo (a NSF repository or airborne lidar data), NEON (a NSF repository of ecological monitoring in the United States), and NASA GSFC (repository of G-LiHT data). In addition to small-footprint lidar data sets, NASA Mission data (i.e. ICESat and GEDI) can also be used in a validation effort for large scale calculations.

Option 2:
Option 2 will include Option 1 as well as the acquisition of additional airborne lidar data that will benefit multiple NASA efforts.

GEDI: With the launch of the Global Ecosystems Dynamic Investigation (GEDI) mission in 2018, there are tremendous synergistic activities for data validation between both the ICESat-2 and GEDI missions. Since the GEDI mission, housed on the International Space Station, has a maximum latitude of 51.6 degrees, much of the Boreal zone will not be mapped by GEDI. The density of GEDI data will increase as latitude increases north to 51.6 degrees. Since the data density for GEDI would be at its highest near 51.6 degrees, we would propose to acquire airborne lidar data in a "GEDI overlap zone" that would ample opportunity to have sufficient coverage of benefit to both ICESat-2 and GEDI for calibration and validation.

We recommend the acquisition of new airborne lidar collections that will meet our requirements to best validate ICESat-2 as well as be beneficial for the GEDI mission. In particular, we would like to obtain data over the following two areas:

1) Boreal forest (as this forest type will NOT be mapped with GEDI)
2) GEDI high density zone (between 50 to 51.6 degrees N). Airborne lidar data in the GEDI/ICESat-2 overlap zone will ensure cross-calibration between these two critical datasets which will allow for the creation of a global, seamless terrain, canopy height, and canopy cover product for the ecosystem community.

In both cases, we would fly data with the following scenario:

Small-footprint, full-waveform, dual wavelength (green and NIR), high point density (>20 pts/m^2) and, over low and high relief locations. In addition, the newly acquired lidar data must meet the error accuracies listed in Table 5.1.

Potential candidate acquisition areas include: Southern Canadian Rocky Mountains (near Banff), Pacific Northwest mountains (Olympic National Park, Mt. Baker-Snoqualmie National Forest), and Sweden/Norway. It is recommended that the airborne lidar acquisitions occur during the summer months to avoid snow cover in either 2016 or 2017 prior to launch of ICESat-2.

5.2 Internal QC Monitoring

In addition to the data product validation, internal monitoring of data parameters and variables is required to ensure that the final ATL08 data quality output is trustworthy. Table 5.2 lists a few of the computed parameters that should provide insight into the performance of the surface finding algorithm within the ATL08 processing chain.

Table 5.2. ATL08 Parameter Monitoring

Group	Description	Source	Monitor	Validate in Field
h_te_median	Median terrain height for segment	computed		Yes against airborne lidar data. The airborne lidar data should have an absolute accuracy of <30 cm rms.
n_photons	Number of classed photons in a 100 m segment	computed	Yes. Build an internal counter for the number of segments in a row where there aren't enough photons (currently a minimum of 50 photons per 100 m segment is used)	
h_te_interp	Interpolated terrain surface height, FINALGROUND	computed	Difference h_te_interp and h_te_median and determine if the value is > a specified threshold. 2 m is suggested as the threshold value. This is an internal check to evaluate whether the median elevation for a	

			segment is roughly the same as the interpolated surface height.	
h_dif_ref	Difference between h_te_median and ref_dem	computed	This value will be computed and flagged if the difference is > 25 m. The reference DEM is the onboard DEM.	
h_canopy	95% height of individual canopy heights for segment	computed	Yes, > a specified threshold (e.g. 60 m)	Yes against airborne lidar data. The canopy heights derived from airborne lidar data should have a relative accuracy <2 m in temperate forest, <3 m in tropical forest
h_dif_canopy	Difference between h_canopy and h_median_canopy	computed	Yes, this is more of an internal check to make sure the calculations on canopy height are not suspect	
psf_flag	Flag is set if computed PSF exceeds 1m	computed	Yes, this is more of an internal check to make sure the calculations are not suspect	
ph_removal_flag	Flag is set if more than 50% of photons in a segment is removed during final QA check	computed		

In addition to the monitoring parameters listed in Table 5.2, a plot such as what is shown in Figure 5.1 would be helpful for internal monitoring and quality assessment of the ATL08 data product. Figure 5.1 illustrates in graphical form what the input point cloud look like in the along-track direction, the classifications of each photon, and the estimated ground surface (FINALGROUND).

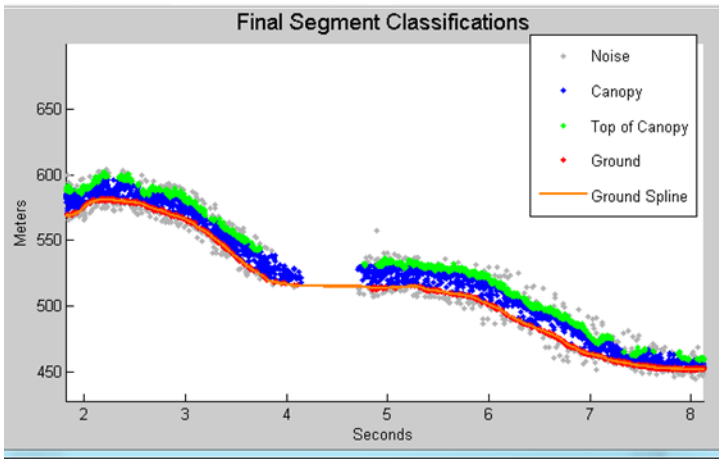

Figure 5.1 Example of *L-km* segment classifications and interpolated ground surface.

Parameters to be in the QA/QC group on the HDF5 data file based on Table 5.2 of the ATL08 ATBD. Statistics will be computed on a per-orbit basis and be reported on the data product.

1. Percentage of segments with > 50 photons
2. Max, median, and mean of the number of contiguous segments with < 50 photons.
3. Percentage of segments with difference in h_te_interp – h_te_median is greater than a specified threshold (2 m TBD). 50 segments in a row with this threshold exceed would send an alert to QAQC team.
4. Max, median, and mean of h_diff_ref, difference between h_te_median and dem_h, over all segments.
5. Percentage of segments where h_diff_ref > 25 m. If percentage is greater than 75%, send an alert to the QAQC team.
6. Percentage of segments where the h_canopy is < 60m
7. Max, median, and mean of h_diff
8. Number and percentage of Landsat continuous tree cover pixels per processing (*L-km*) segment with values > 100.
9. Percentage of segments where psf_flag is set. If percentage is greater than 75%, send an alert to the QAQC team.
10. Alert if more than 50% of the photons in a segment is being removed during final photon QA check (ph_removal_flag is set).

6 REFERENCES

Carroll, M. L., Townshend, J. R., DiMiceli, C. M., Noojipady, P., & Sohlberg, R. A., 2009. A new global raster water mask at 250 m resolution. International Journal of Digital Earth, 2(4), 291–308. http://doi.org/10.1080/17538940902951401

Channan, S., K. Collins, and W. R. Emanuel, 2014. Global mosaics of the standard MODIS land cover type data. University of Maryland and the Pacific Northwest National Laboratory, College Park, Maryland, USA.

Chauve, Adrien, et al. 2008. Processing full-waveform lidar data: modelling raw signals. *International archives of photogrammetry, remote sensing and spatial information sciences 2007*. pp 102- 107.

Friedl, M.A., D. Sulla-Menashe, B. Tan, A. Schneider, N. Ramankutty, A. Sibley and X. Huang, 2010. MODIS Collection 5 global land cover: Algorithm refinements and characterization of new datasets, 2001-2012, Collection 5.1 IGBP Land Cover, Boston University, Boston, MA, USA.

Goshtasby, A., and O'Neill, W.D. 1994. Curve fitting by a Sum of Gaussians. *Graphical Models and Image Processing*, 56:4, 281-288.

Goetz and Dubayah, 2011. Advances in remote sensing technology and implications for measuring and monitoring forest carbon stocks and change. Carbon Management, 2:3, 231-244. doi:10.4155/cmt.11.18

Hall, F.G., Bergen, K., Blair, J.B., Dubayah, R., Houghton, R., Hurtt, G., Kellndorfer, J., Lefsky, M., Ranson, J., Saatchi, S., Shugart, H., Wickland, D., 2011. Characterizing 3D vegetation structure from space: Mission requirements. Remote sensing of environment, 115:11, 2753-2775

Harding, D.J., 2009, Pulsed laser altimeter ranging techniques and implications for terrain mapping, in Topographic Laser Ranging and Scanning: Principles and Processing, Jie Shan and Charles Toth, eds., CRC Press, Taylor & Francis Group, pp. 173 – 194.

Olson, D. M., Dinerstein, E., Wikramanayake, E. D., Burgess, N. D., Powell, G. V. N., Underwood, E. C., D'Amico, J. A., Itoua, I., Strand, H. E., Morrison, J. C., Loucks, C. J., Allnutt, T. F., Ricketts, T. H., Kura, Y., Lamoreux, J. F., Wettengel, W. W., Hedao, P., Kassem, K. R. 2001. Terrestrial ecoregions of the world: a new map of life on Earth. Bioscience 51(11):933-938.

Sexton, J.O., Song, X.-P., Feng, M. Noojipady, P., Anand, A., Huang, C., Kim, D.-H., Collins, K.M., Channan, S., DiMiceli, C., Townshend, J.R.G. 2013. Global, 30-m resolution continuous fields of tree cover: Landsat-based rescaling of MODIS Vegetation Continuous Fields with lidar-based estimations of error. International Journal of Digital Earth, 130321031236007. doi:10.1080/17538947.2013.786146.

DRAGANN Gaussian Deconstruction

John Robbins
20151021 (updates 20170808)

Introduction

This document provides a verbal description of how the DRAGANN (Differential, Regressive, and Gaussian Adaptive Nearest Neighbor) filtering system deconstructs a histogram into Gaussian components, which can also be called *iteratively fitting a sum of Gaussian Curves*. The purpose is to provide enough detail for ASAS to create operational ICESat-2 code required for the production of the ATL08, Land and Vegetation product. This document covers the following Matlab functions within DRAGANN:

- mainGaussian_dragann
- findpeaks_dragann
- peakWidth_dragann
- checkFit_dragann

Components of the k-d tree nearest-neighbor search processing and histogram creation were covered in the document, *DRAGANN k-d Tree Investigations*, and have been determined to function consistently with UTexas DRAGANN Matlab software.

Histogram Creation

Steps to produce a histogram of nearest-neighbor counts from a normalized photon cloud segment have been completed and confirmed. Figure A.1 provides an example of such a histogram. The development, below, is specific to the two-dimensional case and is provided as a review.

The histogram represents the frequency (count) of the number of nearby photons within a specified radius, as ascertained for each point within the photon cloud. The radius, *R*, is established by first normalizing the photon cloud in time (x-axis) and in height (y-axis), i.e., both sets of coordinates (time & height) run from 0 to 1; then an average radius for finding 20 points is determined based on forming the ratio of 20 to the total number of the photons in the cloud (N_{total}): $20/N_{total}$.

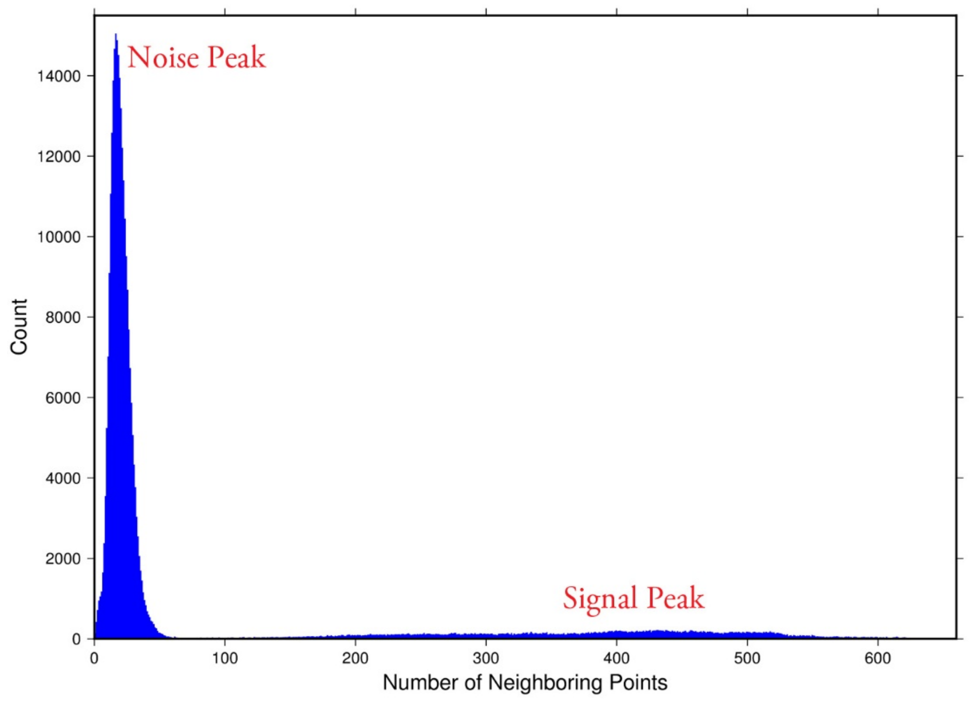

Figure A.1. Histogram for Mabel data, channel 43 from SE-AK flight on July 30, 2014 at 20:16.

Given that the total area of the normalized photon cloud is, by definition, 1, then this ratio gives the average area, A, in which to find 20 points. A corresponding radius is found by the square root of A/π. A single equation describing the radius, as a function of the total number of photons in the cloud (remembering that this is done in the cloud normalized, two-dimensional space), is given by

(A.1)

For the example in Figure A.1, R was found to be 0.00447122. The number of photons falling into this radius, at each point in the photon cloud, is given along the x-axis; a count of their number (or frequency) is given along the y-axis.

Gaussian Peak Removal

At this point, the function, mainGaussian_dragann, is called, which passes the histogram and the number of peaks to detect (typically set to 10).

This function essentially estimates (i.e., fits) a sequence of Gaussian curves, from larger to smaller. It determines a Gaussian fit for the highest histogram peak, then removes it before determining the fit for the next highest peak, etc. In concept, the process is an iterative sequential-removal of the ten largest Gaussian components within the histogram.

In the process of *sequential least-squares*, parameters are re-estimated when input data is incrementally increased and/or improved. The present problem operates in a slightly reverse way: the data set is fixed (i.e., the histogram), but components within the histogram (independent Gaussian curve fits) are removed sequentially from the histogram. The paper by *Goshtasby & O'Neill* (1994) outlines the concepts.

Recall that a Gaussian curve is typically written as

(A.2)

where a = the height of the peak; b = position of the peak; and c = width of the bell curve.

The function, mainGaussian_dragann, computes the [*a, b, c*] values for the ten highest peaks found in the histogram. At initialization, these [*a, b, c*] values are set to zero. The process begins by locating histogram peaks via the function, findpeaks_dragann.

Peak Finding

As input arguments, the findpeaks_dragann function receives the histogram and a minimum peak size for consideration (typically set to zero, which means all peaks will be found). An array of index numbers (i.e., the "number of neighboring points", values along x-axis of Figure A.1) for all peaks is returned and placed into the variable peaks.

The methodology for locating each peak goes like this: The function first computes the derivatives of the histogram. In Matlab there is an intrinsic function, called diff, which creates an array of the derivatives. Diff essentially computes the differences along sequential, neighboring values. "Y = diff(X) calculates differences between adjacent elements of X." [from Matlab Reference Guide] Once the derivatives are computed, then findpeaks_dragann enters a loop that looks for changes in the sign of the derivative (positive to negative). It skips any derivatives that equal zero.

For the kth derivative, the "*next*" derivative is set to k+1. A test is made whereby if the k+1 derivative equals zero and k+1 is less than the total number of histogram values, then increment "*next*" to k+2 (i.e., find the next negative derivative). The test is iterated until the start of the "down side" of the peak is found (i.e., these iterations handle cases when the peak has a flat top to it).

When a sign change (positive to negative) is found, the function then computes an approximate index location (variable *maximum*) of the peak via

(A.3)

These values of *maximum* are retained in the peaks array (which can be *grown* in Matlab) and returned to the function mainGaussian_dragann.

Next, back within mainGaussian_dragann, there are two tests to determine whether the first or last elements of the histogram are peaks. This is done since the findpeaks_dragann function will not detect peaks at the first or last elements, based solely on derivatives. The tests are:

If (histogram(1) > histogram(2) && max(histogram)/histogram(1) < 20) then insert a value of 1 to the very first element of the peaks array (again, Matlab can easily "grow" arrays). Here, max(histogram) is the highest peak value across the whole histogram.

For the case of the last histogram value (say there are N-bins), we have

If (histogram(N) > histogram(N-1) && max(histogram)/histogram(N) < 4) then insert a value of N to the very last element of the peaks array.

One more test is made to determine whether there any peaks were actually found for the whole histogram. If none were found, then the function, mainGaussian_dragann, merely exits.

Identifying and Processing upon the Ten Highest Peaks

The function, mainGaussian_dragann, now begins a loop to analyze the ten highest peaks. It begins the n^{th} loop (where *n* goes from 1 to 10) by searching for the largest peak among all remaining peaks. The index number, as well as the magnitude of the peak, are retained in a variable, called maximum, with dimension 2.

In each pass in the loop, the [*a,b,c*] values (see eq. 2) are retained as output of the function. The values of *a* and *b* are set equal to the index number and peak magnitude saved in maximum(1) and maximum(2), respectively. The *c*-value is determined by calling the function, peakWidth_dragann.

Determination of Gaussian Curve Width

The function, peakWidth_dragann, receives the whole histogram and the index number (maximum(1)) of the peak for which the value *c* is needed, as arguments. For a specific peak, the function essentially searches for the point on the histogram that is about ½ the size of the peak and that is furthest away from the peak being investigated (left and right of the peak). If the two sides (left and right) are equidistant from the peak, then the side with the smallest value is chosen (> ½ peak).

Upon entry, it first initializes *c* to zero. Then it initializes the index values left, xL and right, xR as index-1 and index+1, respectively (these will be used in a loop, described below). It next checks whether the n^{th} peak is the first or last value in the histogram and treats it as a special case.

At initialization, first and last histogram values are treated as follows:

If first bin of histogram (peak = 1), set left = 1 and xL = 1.

If last bin of histogram, set right = *m* and xR = *m*, where *m* is the final index of the histogram.

Next, a search is made to the left of the peak for a nearby value that is smaller than the peak value, but larger than half of the peak value. A while-loop does this, with the following conditions: (a) left > 0, (b) histogram value at left is ≥ half of histo value at peak and (c) histo value at left is ≤ histo value at peak. When these conditions are all true, then xL is set to left and left is decremented by 1, so that the test can be made again. When the conditions are no longer met (i.e., we've moved to a bin in the histogram where the value drops below half of the peak value), then the program breaks out of the while loop.

This is followed by a similar search made upon values to the right of the peak. When these two while-loops are complete, we then have the index numbers from the histogram representing bins that are above half the peak value. This is shown in Figure A.2.

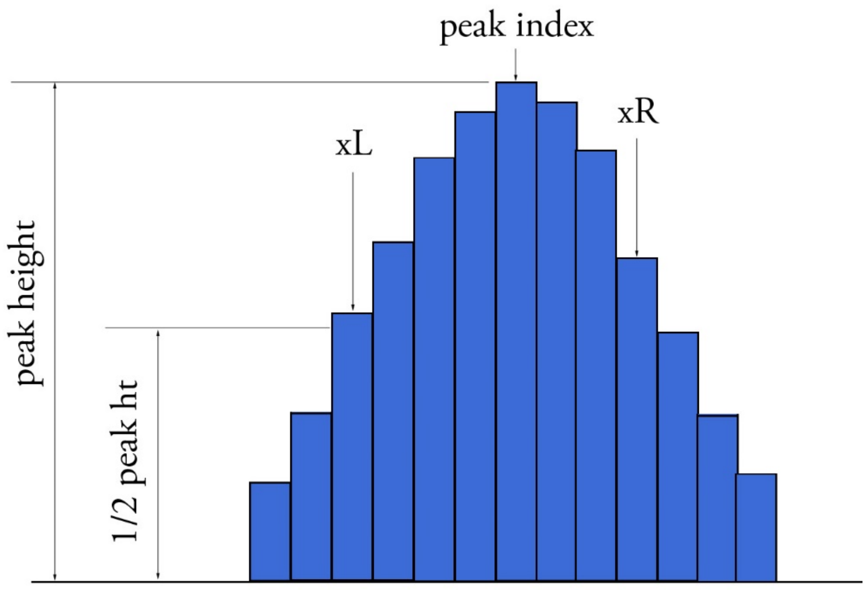

Figure A.2. Schematic representation of a histogram showing xL and xR parameters determined by the function peakWidth_dragann.

A test is made to determine which of these is furthest from the middle of the peak. In Figure A.2, xL is furthest away and the variable x is set to equal xL. The histogram "height" at x, which we call V_x, is used (as well as x) in an inversion of Equation A.2 to solve for *c*:

(A.4)

The function, peakWidth_dragann, now returns the value of *c* and control returns to the function, mainGaussian_dragann.

The mainGaussian_dragann function then picks-up with a test on whether the returned value of *c* is zero. If so, then use a value of 4, which is based on an *a priori* understanding that *c* usually falls between 4 and 6. If the value of *c* is not zero, then add 0.5 to *c*.

At this point, we have the [a,b,c] values of the Gaussian for the n^{th} peak. Based on these values, the Gaussian curve is computed (via Equation A.2) and it is removed (subtracted) from the current histogram (and put into a new variable called newWave).

Numeric Optimization Steps

The first of the optimization steps utilizes a Full Width Half Max (*FWHM*) approach, computed via

(A.5)

A left range, L_r, is computed by L_r=round(b-$FWHM$/2). This tested to make sure it doesn't go off the left edge of the histogram. If so, then it is set to 1.

Similarly, a right range, R_r, is computed by R_r=round(b+$FWHM$/2). This is also tested to be sure that it doesn't go off the right edge of the histogram. If so, then it is set to the index value for the right-most edge of the histogram.

Using these new range values, create a temporary segment (between L_r and R_r) of the newWave histogram, this is called errorWave. Also, set three delta parameters for further optimization:

DeltaC = 0.05; DeltaB = 0.02; DeltaA = 1

The temporary segment, errorWave is passed to the function checkFit_dragann, along with a set of zero values having the same number of elements as errorWave, the result, at this point, is saved into a variable called oldError. The function, checkFit_dragann, computes the sum of the squares of the difference between two histogram segments (in this case, errorWave and zeros with the same number of elements as errorWave). Hence, the result, oldError, is the sum of the squares of the values of errorWave. This function is applied in optimization loops, to refine the values of b and c, described below.

Optimization of the b-parameter. The do-loop operates at a maximum of 1000 times. It's purpose is to refine the value of *b*, in 0.02 increments. It increments the value of *b* by DeltaB, to the right, and computes a new Gaussian curve based on b+Δb, which is then removed from the histogram with the result going into the variable newWave. As before, checkFit_dragann is called by passing the range-limited part of newWave (errorWave) and returning a new estimate of the error (newError) which is then checked against oldError to determine which is smaller. If newError is ≥

oldError, then the value of *b* that produced oldError is retained, and the testing loop is exited.

Optimization of the c-parameter. Now the value of *c* is optimized, first to the left, then to the right. It is performed independently of, but similarly, to the *b*-parameter, using do-loops with a maximum of 1000 passes. These loops increment (to right) or decrement (to left) by a value of 0.05 (DeltaC) and use checkFit_dragann to, again, check the quality of the fit. The loops (right and left) kick-out when the fit is found to be smallest.

The final, optimized Gaussian curve is now removed (subtracted) from the histogram. After removal, a statement "corrects" any histogram values that may drop below zero, by setting them to zero. This could happen due to any mis-fit of the Gaussian.

The *n*th loop is concluded by examining the peaks remaining in the histogram without the peak just processed by sending the *n*th-residual histogram back into the function findpeaks_dragann. If the return of peak index numbers from findpeaks_dragann reveals more than 1 peak remaining, then the index numbers for peaks that meet these three criteria are retained in an array variable called these:

1. The peak must be located above *b(n)-2*c(n)*, and
2. The peak must be located below *b(n)+2*c(n)*, and
3. The height of the peak must be < *a(n)/5*.

The peaks meeting all three of these criteria are to be eliminated from further consideration. What this accomplishes is eliminate the nearby peaks that have a size lower than the peak just previously analyzed; thus, after their elimination, only leaving peaks that are further away from the peak just processed and are presumably "real" peaks. The *n*th iteration ends here, and processing begins with the revised histogram (after having removed the peak just analyzed).

Gaussian Rejection

The function mainGaussian_dragann returns the [*a,b,c*] parameters for the ten highest peaks from the original histogram. The remaining code in dragann examines each of the ten Gaussian peaks and eliminates the ones that fail to meet a variety of conditions. This section details how this is accomplished.

First, an approximate area, area1=a*c, is computed for each found peak and *b*, for all ten peaks, being the index of the peaks, are converted to an actual value via b+min(numptsinrad)-1 (call this allb).

Next, a rejection is made for all peaks that have any component of [*a,b,c*] that are imaginary (Matlab isreal function is used to confirm that all three components are real, in which case it passes). The product a*c then passes through a descending sort. So now, the [*a,allb,c*]-values are sorted from largest "area" to smallest, these are placed in arrays [a1, b1, c1].

Next, a test is made to ensure that at least one of the peaks is within the first 10% of the whole histogram. It is done inside a loop that works from peak 1 to the number of peaks left at this point. This loop first tests whether the first (sorted) peak is within the first 10% of the histogram; if so, then it simply kicks out of the loop. If not, then it places the loop's current peak into a holder (ihold) variable, increments the loop to the next peak and runs the same test on the second peak, etc. Here's a Matlab code snippet:

```
    inds = 1:length(a1);
    for i = 1:length(b1)
        if b1(i) <= min(numptsinrad) + 1/10*max(numptsinrad)
            if i==1
                break;
            end
            ihold = inds(i);
            for j = i:-1:2
                inds(j) = inds(j-1);
            end
            inds(1) = ihold;
            break
        end
    end
```

The j-loop expression gives the init_val:step_val:final_val. The semi-colon at the end of statements causes Matlab to execute the expression without printout to the user's screen. When this loop is complete, then the indexes (inds) are re-ordered and placed back into the [a1,b1,c1] and area1 arrays.

Next, are tests to reject any Gaussian peak that is entirely encompassed by another peak. A Matlab code snippet helps to describe the processing.

```
    % reject any gaussian if it is fully contained within another
    isR = true(1,length(a1));
    for i = 1:length(a1)
        ai = a1(i);
        bi = b1(i);
        ci = c1(i);
        aset = (1-(c1/ci).^2);
        bset = ((c1/ci).^2*2*bi - 2*b1);
        cset = -(2*c1.^2.*log(a1/ai)-b1.^2+(c1/ci).^2*bi^2);
        realset = (bset.^2 - 4*aset.*cset >= 0) | (a1 > ai);
        isR = isR & realset;
    end
    a2 = a1(isR);
    b2 = b1(isR);
    c2 = c1(isR);
```

The logical array isR is initialized to all be true. The i-do-loop will run through all peaks. The computations are done in array form with the variables aset,bset,cset all being arrays of length(a1). At the bottom of the loop, isR remains "true" when either of the conditions in the expression for realset is met (the single "|" is a logical "or"). Also, the nomenclature, ".*" and ".^", denote element-by-element array operations (not matrix operations). Upon exiting the i-loop, the array variables a2,b2,c2 are set to the a1,b1,c1 that remain as "true." [At this point, in our test case from channel 43 of East-AK Mable flight on 20140730 @ 20:16, six peaks are still retained: 18, 433, 252, 33, 44.4 and 54.]

Next, reject Gaussian peaks whose centers lay within 3σ of another peak, unless only two peaks remain. The code snippet looks like this:

```
    isR = true(1, length(a2));
    for i = 1:length(a2)
        ai = a2(i);
        bi = b2(i);
        ci = c2(i);
        realset = (b2 > bi+3*ci | b2 < bi-3*ci | b2 == bi);
        realset = realset | a2 > ai;
        isR = isR & realset;
    end
    if length(a2) == 2
        isR = true(1, 2);
    end
    a3 = a2(isR);
    b3 = b2(isR);
    c3 = c2(isR);
```

Once again, the isR array is initially set to "true." Now, the array, realset, is tested twice. In the first line, one of three conditions must be true. In the second line, if realset is true or a2 > ai, then it remains true. At this point, we've pared down, from ten Gaussian peaks, to two Gaussian peaks; one represents the noise part of the histogram; the other represents the signal part.

If there are less than two peaks left, a thresholding/histogram error message is printed out.

If there <u>are</u> two peaks left, then set the array [a,b,c] to those two peaks. [At this point, in our test case from channel 43 of East-AK Mable flight on 20140730 @ 20:16, the two peaks are: 18 and 433.]

Gaussian Thresholding

With the two Gaussian peaks identified as noise and signal, all that is left is to compute the threshold value between the Gaussians. This is done through a single loop, and two if-statements.

First, a counter is initialized to zero (zerCount=0). Then a variable called setPoint is established as the maximum of a divided by 100 (setPoint = max(a)/100). Based on our example in Figure A.1, the max(a) is going to be about 15,000, thus setPoint comes out to be 150.15.

Next, a do-loop executes with an index (idx) running from b(1)+2-min(numptsrad) to b(2)-min(numptsrad)+1. In our example, these values go from 20 to 434, and the heights of the histogram between these indices will be evaluated. Inside the loop, if the histogram height of index, idx is ≤ setPoint, then another if-statement is executed that looks at the idx-1 and idx+1 values; if either of these is ≤ setPoint, then increment zerCount by 1. When zerCount > 4, set variable cutoff=idx+min(numptsinrad)-1, and then exit the loop.

After finding four values that are below 0.01*max(a), the algorithm concludes that the histogram has "leveled-off" sufficiently enough after encountering the first noise peak. This value of cutoff is retained unless a better threshold is found, as described in the next three paragraphs, below.

An array of xvals is established running from min(numptsinrad) to max(numptsinrad). In our example, xvals has indices between 0 and 653. For each of these xvals, Gaussian curves (allGauss) are computed for the two Gaussian peaks [*a,b,c*] determined at the end of the previous section. This computation is performed via a function called gaussmaker which receives, as input, the xvals array and the [a,b,c] parameters for the two Gaussian curves. An array of heights of the Gaussian curves is returned by the function, computed with Equation A.2. In Matlab, the allGauss array has dimension 2x654. An array, noiseGauss is set to be equal to the 1st column of allGauss.

An if-statement checks whether the b array has more than 1 element (i.e., consisting of two peaks), if so, then nextGauss is set to the 2nd column of allGauss, and a difference, noiseGauss-nextGauss, is computed. The following step is restricted to be between the two main peaks. Now the point (i.e., index) is found of the minimum of the absolute value of the difference; this index is put into variable, signchanges. This point is where the sign changes from positive to negative as one moves left-to-right, up the Gaussian curve differences (noise minus next will be positive under the peak of the noise curve, and negative under the next (signal) curve). Figure A.3 (top) shows the two Gaussian curves. The bottom plot shows their differences. The index of the threshold is now saved into variable, threshNN, concluding the if-statement for b having more than 1 element.

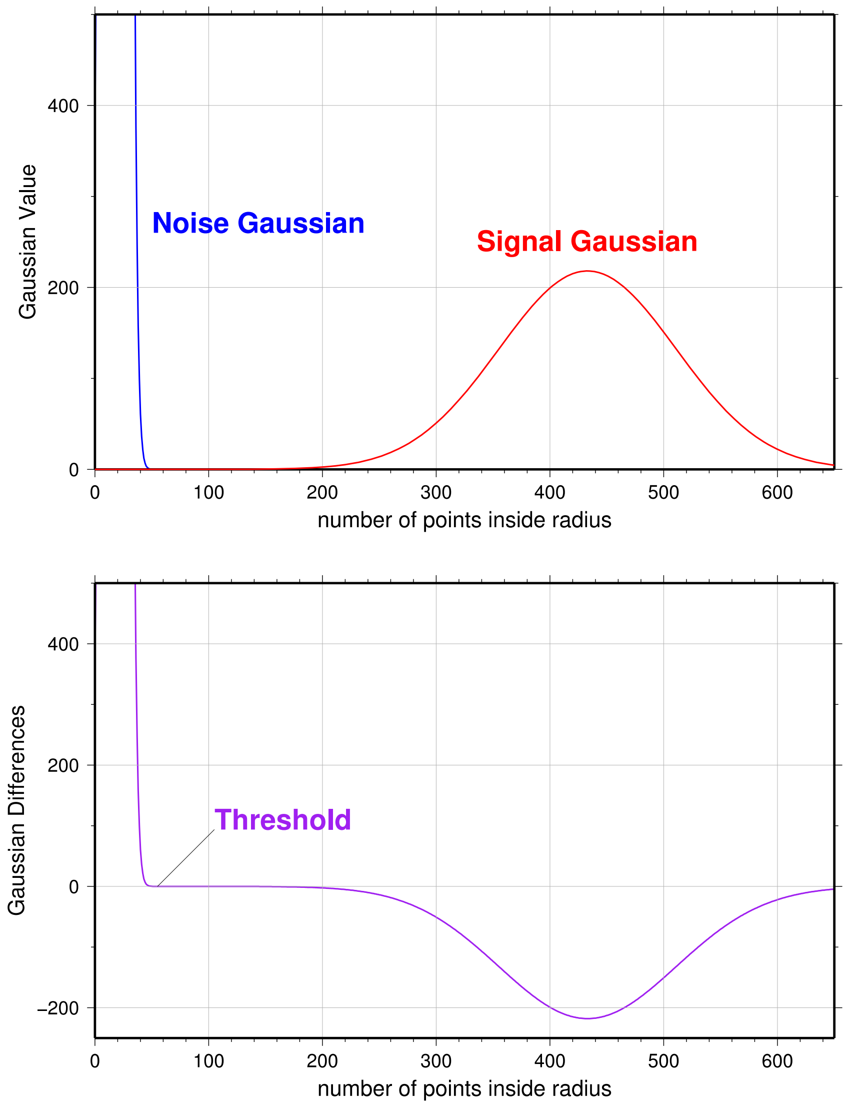

Figure A.3. Top: two remaining Gaussian curves representing the noise (blue) and signal (red) portions of the histogram in F1gure A.1. Bottom: difference noise – signal of the two Gaussian curves. The threshold is defined as the point where the sign of the differences change.

An else clause (b !> 1), merely sets threshNN to b+c, i.e., 1-standard deviation away from mean of the (presumably) noise peak.

Next, an if-statement checks whether the variable cutoff exists (which it would if the histogram remained near to zero for more than four bins in a row). When this is the case, then threshNN is set to the index saved in the variable, cutoff.

The final step is mask the signal part of the histogram where all indices above the threshNN index are set to logical 1 (true). This is applied to the numptsinrad array, which represents the photon cloud. After application, dragann returns the cloud with points in the cloud identified as "signal" points.

The Matlab code has a few debug statements that follow, along with about 40 lines for plotting.

References

Goshtasby, A & W. D. O'Neill, Curve Fitting by a Sum of Gaussians, *CVGIP: Graphical Models and Image Processing*, V. 56, No. 4, 281-288, 1994.