PNW-FIA LiDAR script information

All scripts are compiled for Windows operating systems and come into two formats: 64-bit and 32-bit. The '32'suffix is used to determine the one format from the other. They read and process binary LAS files in formats 1.0, 1.1, or 1.2. Access to the LAS 1.3 format specification has not been implemented. Efforts have been made to promote script scalability, parsimonious use of computer resources, and execution speed.

The scripts can be downloaded from http://pnwfia.info/dgatziolis/scripts.htm. With the exception of the LAS_prj.exe (and LAS_prj32.exe) there are no external dependencies. I have included 4 dynamic link libraries (DLLs) in the *dependencies.zip* file you can download from the link above in case your Windows system does not have them. Scripts are to be executed from the command line, as there is no graphical user interface (GUI). They are written in C programming language with parallelization enabled via the OPENMP API. For the majority of the scripts reading from and/or to the disk is by far the most time consuming operation. The actual data processing is only a fraction of the input/output (I/O) operations. For those scripts parallelization has not been implemented, since it yields a trivial improvement in performance.

Successful script execution is often conditioned upon the size of the input LAS file and the options specified by the user. Where possible data from LAS files is read and processed incrementally with minimal use of memory. Often though, the algorithms I am implementing require that the entire data in an LAS file is loaded into memory, which can become a limitation, especially for machines with 32-bit operating system or limited RAM. If you can error stating that the program "failed to allocate memory" for a file/vector/matrix/array, etc. you would other split the corresponding LAS into one or more parts or degrade the resolution, search radius, or other similar parameter. In the latter case execution will be slower but less susceptible to memory limitations.

If you have used my scripts and found them useful, please acknowledge my efforts wherever possible and warranted. Your support is very important and necessary for script maintenance and future development.

If you believe you have found an error please do the following:

- Read this guide, with close attention to the section that mentions the script in question,
- Thoroughly document the error (frequency, consistency, conditions, etc.) and send me a small piece of the data that triggers it. From past experience, 9 out of 10 script error reports received had to do with LAS file issues, mainly departures from the LAS specifications.
- Submit an email to lidar<the at symbol>pnwfia.info. Since even a piece of an LAS is still a large file, we will likely have to arrange an alternative way to transfer the file(s).

I will address your concerns as my time and work schedule permit.

LAS_Header_View.exe

```
Syntax: LAS_Header_View.exe <LAS File>
```

Every LAS file contains a header with summary information. The script sends to the standard output, usually your computer monitor, the contents of the header in a formatted layout. The actual data contents of the file are not queried.

LAS_Update_Header.exe

```
Syntax: LAS_Update_Header.exe <LAS File>
```

The scripts queries the contents of the data without considering the information present in the header of the LAS file. It then updates the header using the information read. No updates are made to the ‘Generating Software’ field of the header.

LAS2ASC.exe

```
Syntax: LAS2ASC.exe  <LAS File> <ASCII File> <ASCII Header File>
                     <Append FieldNames [Y/N]> <Format>

                      Format:
                      0 -- All
                      1 -- X
                      2 -- Y
                      3 -- Z
                      4 -- Intensity
                      5 -- Return Number
                      6 -- Number of Returns
                      7 -- Scan Direction
                      8 -- Edge of Flight Line
```

```
 9 -- Classification
10 -- Scan Angle Rank
11 -- File Marker / User Data
12 -- User Bit Field / Point Source ID
13 -- GPS Time (if available)
14 -- RGB (if available)
15 -- X Y Z
16 -- X Y Z Intensity
17 -- X Y Z GPS
18 -- X Y Z Intensity GPS Time
19 -- X Y Z RGB
```

A file in binary format is read typically an order of magnitude faster than if in text format, but viewing its contents usually requires specialized software. Text files can be accessed by editors available with all operating systems. LAS2ASC.exe converts the binary version of an LAS file to its text, or ASCII, equivalent. For every binary LAS, two text files are produced: The <ASCII File> contains the actual return data while <ASCII Header FILE> contains the header information. The user can choose whether to append the name of the fields in the first line of the header file and also the format. There are 20 format options. Some of them are available conditionally, that is if they are supported by the format of the LAS file. For instance, in LAS format 1.2, type 0, there is no GPS Time information. If format 18 is requested, which includes GPS Time, an error will be returned. All output files are space delimited. I have often used this script to get text versions of the entire LAS data, or certain data fields, and further process them in R.

LAS_Stats.exe

```
Syntax: LAS_stats.exe <Flags> <LAS FILE>
        Flags [Optional]
        -h Help
           -d Detailed Report to standard output
           -e Save report to LAS FILE_stats.asc
```

Queries the contents of an LAS file and creates a standard or detailed report. If the –e option is used the report is also sent to a text file.

The standard report shows

- Number of returns
- X range
- Y range
- Z range
- GPS Time range (if field is available)

- Number of 1st returns
- Number of 2nd returns
- Number of 3rd returns
- Number of 4th returns

- Number of 5th returns
- Number of 6th returns

- Number of returns in single-return pulses
- Number of returns in pulses with 2 returns
- Number of returns in pulses with 3 returns
- Number of returns in pulses with 4 returns
- Number of returns in pulses with 5 returns
- Number of returns in pulses with 6 returns

- Number of returns in classification 1
- Number of returns in classification 2
- ...
- Number of returns in classification N

The detailed report extends the standard report by including the following:

- Frequency of returns in each Intensity class [0, 65535]
- Frequency of returns in each Scan Angle Rank class, measured in degrees [-89, 89], with negative integers corresponding to return to the left of the platform's trajectory
- Frequency of returns in each User Data class [0, 255], which is sometimes used by LiDAR vendors to store the intensity gain effective on the laser instrument the moment a pulse in transmitted
- Frequency of returns in Point Source ID class [0, 65535], which is often use by vendors to store an ID for a scanning swath

The frequencies are sorted by class value and the list can be discontinuous as class values with 0 frequencies are ignored.

LAS_Subset_by_Attribute.exe

```
Syntax: LAS_Subset_By_Attribute.exe <Input LAS> <Output LAS>
                                    <Attribute> <Min> <Max>
```

Attribute values:

1 X
2 Y
3 Z
4 Intensity
5 Return Number
6 Number of Returns
7 Scan Direction Flag
8 Edge of Flight Line
9 Classification
10 Scan Angle Rank
11 User Data or File Marker
12 Point Source ID or User Bit Field
13 GPS Time

The input LAS file is subset using the specified attribute and range of values for that attribute and the result is sent to the output LAS file. If no returns comply with the set criteria no output is generated. This script can be used to isolate first returns, returns classified as ground or above-ground, belonging to a certain Point Source ID or User Data class, etc. If the desired range of values for a particular attribute is discontinuous, say Intensity less than 10 or more than 200, the script has to be run two or more times, with the resulting output LAS files joined later into one file using LAS_merge.exe. In the previous example the range in the first run should be [0, 9] and in the second [201,255]. The script can be also useful in splitting the file into parts to avoid memory issues with other scripts. For example, you can run LAS_stats.exe first and get the range of values for X coordinate (East-West) direction, or the Y coordinate (North-South) direction. Then you can split the range in two or more parts and use the subranges with LAS_Subset_by_Attribute.exe to have the original LAS file split into two or more parts.

LAS_Subset_by_Raster.exe

```
Syntax: LAS_subset_by_Raster.exe
            <Raster file (.flt or .asc)>
            <Input LAS>
            <Output LAS>
            <Minimum elevation value relative to raster cell value>
            <Maximum elevation value relative to raster cell value>
```

The script uses a binary (FLOATGRID) or ASCII (GRIDASCII) raster file to subset the input LAS file using a user-specified range of values relative to the raster. The output LAS file is deleted if the set criteria yield no retained returns. Only returns located on value raster cells are included in the output LAS; a return located, in two dimensions, to the area occupied in a NODATA raster cell is ignored. Reading raster data in ASCII format is slow. Either use binary raster format or clip the raster to the extent of the LAS file.

LAS data are read one record at a time to a buffer. X, Y, and Z attributes are extracted from the buffer and overlaid with the raster. The combination of raster data (value cell, NODATA cell, or outside the raster's bounding box) and Z value determines if the record is written to the output file. The scripts reports the number of returns in the output LAS, on NODATA raster cells, outside the raster's bounding box, or rejected by the selection criteria.

The raster usually represents a DEM. The script can be used to isolate above or below ground returns; or returns within a specific elevation range. It can also be used with other rasters to perform spatial clipping. Assume you have a shapefile of a forest stand and you need to clip a LIDAR point cloud to that stand. You first need to convert the shapefile to a raster whose cells have value, say, 1 and NODATA anywhere else. This raster can be used to clip the LAS file by specifying a very large range of elevation values, say -100,000 to 100,000. The output LAS file will contain only returns on raster cells with value 1 regardless of the Z attribute for each return. A small resolution for the raster file will ensure decent correspondence between the shapefile boundary and its raster representation.

LASZ2H.exe

```
Syntax: LASZ2H.exe <Input LAS> <Output LAS> DEM[*.asc or *.flt]
                   Options:
                      -tr (Truncate negative heights to zero)
```

As in the previous script, this one uses a DEM file in raster form, either text or binary, to convert return elevation to height above ground. The script reports the number of returns in the output LAS, on NODATA DEM cells, or outside the DEM's bounding box. Most LAS file will contain a number of returns that are below the DEM generated from the same laser data. The great majority of these returns will get a small negative Z value in the output LAS. If this is undesirable, the user can use the –tr option which will convert all negative Z values to 0.

LAS2Shape.exe

```
Syntax: LAS2shape.exe <LAS file>
```

Converts an LAS file to point shapefile. The output shapefile comprises three files with extensions .shp, shx, and dbf. The latter contains the return attributes.

The script assumes that ScanAngleRank (a return attribute) values in the input LAS are, after converted from unsigned to signed format, no smaller than -99. Theoretically valid scan angles range between -90 to 90 degrees, but in practice they do not exceed 30 degrees. Signed ScanAngleRank values must be represented by a maximum of 3 hexadecimal characters in the output DBF file. LAS specs define ScanAngleRank as unsigned char, hence the range is 0 to 255. Angles at the left of the airborne platform have values close to 255 and you have to subtract 256 from them to get them as negative. Hence angles with value in .las 127 to 154 will result in signed values smaller than -99 and will generate a runtime error. The script does not check for such values because they are invalid. I have yet to see this error in practice.

The November 2010 version of this script was using Frank Warmerdam's code to create the .shp, shx, and .dbf components of a shapefile. Because of substantial overhead in the API runtime was long, especially for large .las files. The present version uses native code and it is nearly an order of magnitude faster.

LAS_Tile.exe

```
Syntax: LAS_Tile.exe <LAS_List (.asc)> <LAS_Tile_Prefix> <TileSize>
                     -r Raster_Resolution [Optional]
                     -b Buffer_Size [Optional]
                     -s [Optional; Saves tile info to a .dbf file]
```

Rearranges existing LAS tiles or tiles LAS files representing scanning swaths or custom areal configurations. The user specifies a text file (with .asc extension) containing a list of LAS files.

All LAS in the list must have the same scale and offsets. The next expected argument is a prefix that all output LAS will have. Tile size is the last mandatory argument. If the size is too small, the data in a single input file will be distributed among a large number of tiles. To avoid having an excessive number of output files open for writing simultaneously, the maximum number of tiles per input file is set to 100. If the size of the tile is too large, the number of output LAS will remain small but the individual tile LAS can exceed the 2.14GB (32-bit) limitation which can be an issue with other applications.

Optional argument -r (raster resolution) controls the location of the origin of the tiling grid. A proper choice of tile size and resolution combination would guarantee alignment of output tiles with other raster data. Optional argument -b (buffer size) adds a buffer around each tiled LAS. When a buffer is used, a single return can be included in up to 4 tiles. Thus a large buffer will generate substantial data duplication. The size of the buffer cannot be larger than haft the size of the tile. A buffer ensures that laser cloud derivatives, whose computation involves an area of reference, near the unbuffered edge of the tile won't contain edge artifacts. The size of the buffer should exceed the size of the areal reference mentioned above. Using both -r and -b options provides flexibility with raster LAS derivatives. Optional argument -s saves the tiling information to a shapefile and the tile vertices to an additional .dbf file. If -b options has been used, two shapefiles are generated; one containing the unbuffered and the other the buffered tiles. The shapefiles can be used to clip derivatives of the buffered raster to their unbuffered equivalents. The shapefile tables report tile name, row, column, and number of returns in the tile. If a buffer has been specified, the number of returns reported for a tile in the two shapefiles generated includes the returns located at the buffer.

LAS2Raster.exe

```
Syntax: LAS2Raster.exe <LAS File> <Raster Resolution> <Output Prefix>
                       <Output File Type (ASCII or FLOAT)>
                       Options:
                           [-DEM FileName (.asc, .txt, or .flt)]
                           [Output Metrics]
Metrics:
   -CNT(return count)
   -Hmin -Hmedian -Hmean -Hmax -Hstdev (return height)
   -H10 -H20 -H30 -H40 -H50 -H60 -H70 -H80 -H90 (height percentiles)
   -Imin -Imedian -Imean -Imax -Istdev (intensity)
   -I10 -I20 -I30 -I40 -I50 -I60 -I70 -I80 -I90 (intensity percentiles)
   -RNmean -RNmedian -RNstdev (return number)
   -UDmean -UDmedian -UDstdev (user data)

If no metrics are specified, all will be computed
Metrics should be placed after the Output File Type
```

Generates selected laser cloud metrics as organizes them in raster format using a user-specified resolution. If no metrics are specified, all will be computed. Metrics should be placed after the Output File Type.

The -DEM argument is no metric. If only -DEM DEMFileName is specified, all metrics will be computed. If -DEM is specified, returns present in the input LAS file but either on DEM NODATA cells or outside the DEM's bounding box will be ignored. The –DEM values will be used to convert the return elevation values to height (above ground) values. Specifying -DEM will not affect the other metrics, except height, unless there are return on DEM NODATA cells or outside the DEM's bounding box. In that case, the computed rasters for, say, intensity will be different if -DEM is specified from when -DEM is not. If -DEM is not specified and the Z values in the input LAS represent elevation, requested height metrics will represent elevation, not height.

The DEMFileName can have extension either 'asc' if in GRIDASCII format or 'flt' if FLOATGRID format. In the latter case a corresponding file with 'hdr' should also be present. GRIDASCII and FLOATGRID formats follow the ArcInfo standards. The raster output can be FLOAT or ASCII. In the first case binary output is generated in FLOATGRID format. In the second case the format is GRIDASCII.
The origin of the rasters is located on a multiple of the resolution specified. Components of the script are executed in parallel.

LAS_Incidence_Angle.exe

```
Syntax: LAS_Incidence_Angle.exe <Input LAS> <Reference data file>
```

Reference data contain the 3D location of the laser instrument with time (X, Y, Z, and time values for each location in that order) stored in a space-delimited ASCII file without missing values. The range of values in each reference data file field is checked.

The script computes the incidence angle for each return in the input LAS file. The incidence angle is defined as the angle formed by the trajectory of the pulse that generated the return and a line perpendicular to a horizontal plane at the locus of the return. Its theoretical range is [0, 90) degrees. It only depends on the location of return and the location of the instrument that emitted the return’s pulse. Should not be confused with the scan angle rank attributes embedded in the LAS files. The scan angle is measured relative to the attitude of the airborne platform that carries the laser instrument and only to the side of the platforms trajectory. In laser acquisitions with volatility in platform attitude, scan angles can be very different from incidence angles. A 14 degree cutoff in incidence angle is often advocate as a necessary for avoiding return artifacts such as those cause by path backscattering and other phenomena. Laser vendors that implement this cutoff actually use the scan angle, not the incidence angle. This script can be used to determine the number, or percentage of returns per incidence angle class.

Computed incidence angles are saved into a text file with two digits of precision with the same order as the corresponding laser returns.

Only LAS files containing GPS time information can be used. The entire airborne platform location with time (or reference) data and LAS data are loaded into memory. The script checks if the LAS GPS time range is completely within the reference GPS time range.

To compare scanning and incidence angles I use LAS2ASC.exe to get the X, Y, and Z coordinates from the LAS file. Then I run LAS2ASC.exe again to export the ScanAngleRank into a text file. I then combine the X, Y, Z, scanning angle, and incidence angle into a dataframe in R. Will all data in R, I can map each of the two angles in 2D space, compute angle correlation, etc.

LAS_prj.exe

```
Syntax: LAS_prj.exe  <LAS | ASCII_List> <FromProjectionFile>
                     <ToProjectionFile> <ZunitConversionCode>
                     Codes:
                        1: nochange
                        2: meters to feet
                        3: feet to meters
```

Projects XYZ coordinates of LAS file(s) from a source to target coordinate system. Source and target projection information is provided by the user in .wkt files. Argument ZunitConversionCode control conversion of Z coordinates, if needed. Accepted values are 1 (no change), 2 (change Z unit from meters to feet), and 3 (change Z unit from feet to meters). The user can specify one LAS file or many LAS files listed in a text file. The output LAS file(s) contain a _prj suffix. It is assumed that the entire LAS file processed and intermediate x, y, and z vectors can be stored into memory. If this is not the case the offending LAS files should be split into smaller files.

It is very important to ensure that the correct WKT files are selected for describing the source and target coordinate systems. They can be obtained from many places, including the www.spatialreference.org. Note that the differences between WKT files can be minute. An offset can be a little different or a meridian. It is advisable to use the script with a single file and verity that the output LAS, perhaps converted to a raster using LAS2Rraster.exe, is correctly georeferenced. An overlay with other georeferenced data over the same area of interest in the target projection usually suffices in determining if the projection has been successful. Also note that occasionally the source and target projection systems do not share a direct transformation. In that case perhaps an intermediate step can be used.

Projecting entire laser data sets is not advisable. If you must overlay, fuse, combine, or compare LAS data sets at the return level over the same area but in different projection systems you can use this script.

The script uses the GDAL API. Unless you have GDAL DLLs installed on your computer and in a directory referenced to the path used by your command line, you much add in the working directory, on add their directory to the path, the dependent DLLs. There are two versions, one for 64 bit (available in LAS_prj 64-bit dependencies.zip) the other for 32-bit (available in LAS_prj 32-bit dependencies.zip). Note that these DLLs are only used by LAS_prj.exe.

LAS_RotTrans.exe

```
Syntax:
LAS_rottran.exe  <INPUT.las> <OUTPUT.las>
                 <X translation> <Y translation> <Z translation>
                 <Rotation angle> <Rotation center X> <Rotation center Y>
```

Performs translation (shifting) and rotation of LAS coordinates by user-specified distances and angles. Distances are measured in native LAS data units, and angles in degrees. The script first rotates the point cloud and then translates it. Because the number of coordinate decimals retained by the LAS file is small, typically 2, precision is lost during processing. In the event that the rotated/translated coordinates are inverted back by using the reverse script arguments, you should expect that the twice rotated/translated coordinates would differ from the original ones by small amounts, but no more than 0.04 spatial units.

This script has been developed to assist in the assessment of the effects return registration precision has on derivatives computed by processing point clouds. Minor but continuous changes in airborne platform attitude, errors in triangulated GPS coordinates, and distance precision loss emerging from the fact that time elapsed between pulse photon transmission and retrieval can be measured only in fine, but not infinitesimally small increments, lead to registration discrepancies of returns within and between scanning swaths. Swath registration typically relies on the overlapping area of DEMs extracted from the adjacent swaths. DEM registration ranges between 0.5 to 2.0 or more meters. The mean registration precision of returns in adjacent swaths registered by following the procedure described above cannot be finer that half the DEM resolution. Registration errors within the same scanning swath are not address at all. LAS_RotTrans.exe allows rotating and translating returns within a small area and belonging to one scanning swath to be repositioned and then combined with the returns in the same small area acquired in a different scanning swath. The combined set is then evaluated by independent registration criteria. The process is repeated for a large number of translation intervals along each of the three axes and angular increments clock – and counterclockwise. The translation/rotation combination that yields the best registration for the area is thus computed.

LAS_Merge.exe

```
Syntax: LAS_Merge.exe <ASCII List> <LAS Output>
```

Combines two or more overlapping or adjacent LAS files listed in a text file (ASCII List) into a single output LAS. The script verifies format compatibility of the input files and gives a warning if the size of the output file exceeds the 32-bit limit (~2.147GB). The script stops if the number of returns in the requested output file is expected to exceed the 32-bit limit.

This script can generate a huge file. A file in LAS format 1.2 type 3 has 34 bytes for every return and will be more than 146GB before the return number limitation is activated. Having the data from an entire LiDAR acquisition in one or very few files may be initially appealing but will likely become a nightmare when the time comes to process the data!

---

trj2ASCII.exe

```
Syntax:
trj2ASCII.exe <trajectory file name [*.trj] | list of files[.asc or txt]>
              <ASCII output file name [.asc or .txt]>
              <Output Type [1: X Y Z T, 2: X Y Z T ID]>
```

Converts one or more (listed in text file with .asc or .txt extension) binary airborne platform trajectory files to a single space-delimited text file, with four (X, Y, and Z coordinates, and GPS Time readings) or five (plus a scanning line or flight ID descriptor) data columns. The binary trajectory file must have a .trj extension. They are used to determine distance and angular offsets between a laser return and the corresponding laser scanner position. The computation of these offsets is a prerequisite for computing return incidence angles (LAS_incidence_angle.exe), for normalizing return intensities (LAS_IntRopt.exe), and other applications. With the trajectory information in text form it is easier to determine whether GPS Time data is unique or not. Details on why this is of importance are available in the information for LAS_IntRopt.

LAS_IntROpt.exe

```
Syntax:
LAS_IntRopt.exe
    <ASCII file w/ list of LAS files>
    <Laser trajectory data in X, Y, Z, T space-delimited ASCII>
    <Max distance, in 3D, of returns forming a pair>
    <Radius, in 2D, when searching for pairs of returns>
    <Min time difference between scanning swaths>
    <Max time difference between scanning swaths>
    Options:
     -tr {truncate normalized intensities to the [1,255] range}
     -n  {no normalization; only compute normalization parameters}
```

This script computes the model parameters and performs intensity normalization of laser point clouds acquired with fixed-gain instruments. It uses returns that are very close to each other (in three dimensions) but generated from different scanning swaths. The great majority of laser acquisitions target some degree of overlap between adjacent scanning swaths to ensure that there are not areas missed during the flights. The script utilizes this overlap to identify return pairs. It computes the mean range (distance to the laser instrument) of all paired returns and iteratively calculates the optimal exponent of actual to mean range. Optimal is the exponent that minimizes the sum of the squared intensity differences within the pairs. It has a theoretical range 0 to 4. Details on the procedure can be found in Gatziolis (2011)[1]. The full paper is available at http://pnwfia.info/dgatziolis/PDFs/Gatziolis_PERS_2011.pdf.

[1] Gatziolis, D. Dynamic range-based intensity normalization for airborne, discrete return LiDAR data for forest canopies. 2011. *Photogrammetric Engineering and Remote Sensing*, 77(3):251-259.

Observed intensities in fixed-gain LiDAR data sets are inversely related to the distance between laser instrument and illuminated targets and proportionally to the cosine of the pulse incidence angles. In a fixed-gain instrument, sensor sensitivity is set *a priori* and remains unchanged for the entire flight. Due to attenuation of pulse energy (i.e. reduction in photon density), objects closer to the laser instrument yield higher intensities, as pulse energy backscattered by them returns to the instrument with fewer energy loses. The script should not be used with data acquired with dynamic gain settings because for those data the effects of distance and incidence angle on intensity have, at least partially, been accounted. Running the script on such data will yield model parameters values near their bounds of or outside the theoretical range and will introduce additional noise in the observed intensities. Fixed-gain acquisitions have become increasingly rare after 2007-2008, since modern LiDAR instruments have since been equipped with dynamic-gain sensors. There are, however, many fixed-gain acquisitions archived. Their information content can be improved by performing intensity normalization.

The file with the airborne platform trajectory data should be space-delimited ASCII with fields X, Y, Z, and T (GPS Time) in that order and there should be no missing values. The script uses the GPS time recorded for each return to determine the location of the aircraft at that time. If GPS time records in the laser trajectory data file and each of the LAS files used as input are unique, a return is safely matched to a trajectory record and the return's range and incidence angle are computed correctly. Because returns are generated with a much higher frequency (50kHz +) than records of airborne platform location (1-200Hz), the exact platform location corresponding to a return is interpolated from the previous and next recorded location in the platform trajectory file. In acquisitions over large areas that last for more than a week, the same range of GPS time can appear more than once, because GPS Time is reset every Sunday at 12:00 am. This is a complication because GPS Time alone is not sufficient for linking a return to a unique airborne platform location. The situation can become even more complicated if portions of the acquisition area are scanned exactly a week (or more weeks) apart, since they will exhibit similar GPS time and similar X, Y, and Z return coordinates. In such conditions accurate computation of return-to-instrument range and incidence angle is very challenging and computationally time consuming. Where GPS time ranges are expected to be, even partially, duplicated during an acquisition, it is advisable that the LAS files are furnished with a unique flight identifier that is also embedded in the trajectory files. The presence of a flight ID greatly simplifies processing. Hence, if you have used LAS_IntRopt.exe and got unexpected results or runtime errors, use trj2ASCII.exe to get a text version of the trajectory file, and then import the text file in R, Excel, a database, or other utility and verify that there are no GPS time overlaps. If they are, some additional effort will be required before the script can be run. Let me know of such issues; I may be able to provide assistance.

The script generates a new LAS file for every input LAS file. As evident above, the input LASs must contain GPS time information. Processing is memory intensive. Calculating laser instrument to return distance (range) and incidence angle, both algorithmic requirements, requires that the entire platform trajectory file and the X, Y, Z, Intensity, and GPS time component of the currently processed LAS file are stored in memory, plus a number of GPS time and return ID vectors. To avoid memory problems I have limited the maximum size of an LAS that can be processed by LAS_IntROpt.exe to approximately 1.6GB, which is equal to 75% of

the 32-bit limit. If you have a file larger than this limit, you will need to split it by using the LAS_subset_by_attribute.exe.

There are two optional arguments: -tr truncates normalized intensities to the [1,255] range. Without the -tr option the potential range of observed intensities is [0, 65535] because the LAS format specification defines intensity as an unsigned short integer. If argument -n is used, the script stops after computing the mean range and optimal range ration exponent mentioned above and presenting a report. The -n option is useful for an initial evaluation of the number of pairs that are identified for a given maximum distance between return pairs, and mean range in each LAS file.

The number of identified pairs increases exponentially as the maximum distance that defines pair becomes larger. An unreasonably distance could lead to a very large number of pairs identified and prolonged processing time. To assess approximately how many are identified and thus determine if the distance specified is appropriate, the user can run the script with the –n option and only a couple of LAS files. Suggested initial distance value is the $1/10^{th}$ of the mean, between-pulse footprint distance. Most vendor reports include it. For an acquisition with mean return density 7 to 8 returns per square meter, a 0.05 m distance could be appropriate. Also 100,000 return pairs across the entire acquisition are more than enough for a robust estimation of the range ratio exponent, provided that they are uniformly distributed across the total area of interest.