Geosensing Engineering and Mapping (GEM)
Civil and Coastal Engineering Department
University of Florida

Normalizing Lidar Intensities

GEM Center Report No. Rep_2006-12-001

(This report supersedes GEM Center Report No. Rep_2004-07-001)

M. Starek, B. Luzum, R. Kumar, K.C. Slatton

December 22, 2006

Point of Contact:
Prof. K. C. Slatton
University of Florida; PO Box 116130; Gainesville, FL 32611
Tel: 352.392.0634, Fax: 352.392.0044, E-mail: slatton@ece.ufl.edu

Normalizing ALSM Intensities

1.0 Introduction

We refer to airborne lidar designed to image the terrestrial surface as Airborne Laser Swath Mapping (ALSM). The intensities of reflected laser returns detected by an ALSM system are affected by the following factors: variations in path length, surface roughness and orientation, beam divergence, object composition, object density, saturation from background reflections, attenuation of the signal through the atmosphere, and ALSM system characteristics [1].

Surface roughness and orientation, composition, and density are fundamental components of the spectral reflectivity properties of the observed surface/object. Furthermore, laser measurements are usually not affected by background reflections such as reflections due to ambient sunlight. The Optech ALTM 1233 system utilized by the GEM Center scans laser pulses within a preferred range of angles and is designed to operate in daylight. The system utilizes a narrow-band spectral filter to prevent light from entering the detector that does not fall within a few Angstroms of the 1064 nm laser wavelength. This prevents extraneous light from producing spurious results [1]. In addition, atmospheric conditions and thus, effects on intensity values due to atmospheric extinction, are likely to be similar throughout the entire data collection as long as the data are collected within a relatively short period of time (i.e. a few days). Therefore, intensity values sampled across a region within a small temporal range can often be considered to be influenced by the same atmospheric extinction values. Note that if intensity data with significant temporal displacement are being compared this assumption may not be valid.

Those effects on intensity that are due to intrinsic components of the surface/object are regarded for some applications as important variations to be studied, and are therefore not “corrected”. Other effects, such as variations in atmospheric attenuation, may be unwanted but information about the acquisition environment is often insufficient to allow for their correction. However, large variations in the laser path length result in unwanted weakening or strengthening of the intensities that have nothing to do with surface properties. Therefore, these effects can and should be corrected for in the ALSM intensity data. Factors that influence path length, and consequently intensity, include changes in ground topography, variations in flying height, and variations in scan angles.

As can be observed in Figure 1, the path length changes dramatically due to the drastic variations in ground topography, resulting in dimmer returns from the lower elevations on the right side of the figure and brighter returns from the higher elevations on the mountain, assuming the reflectivity properties of the surfaces are similar. The same trends occur as the path length varies due to changes in scan angles (e.g. –20 deg to +20 deg) and changes in flying heights on different flight lines. To correct for these effects, the intensity values must be normalized. If the ALSM intensity data is not normalized to correct for varying path length, comparison of intensity values and display of intensity images across and between scans and different scan regions cannot be properly conducted (see Figure 2). The left side image in Figure 2 displays an intensity image that combines three adjacent (tiled) regions created from non-scaled (non-normalized) intensity values. As can be observed in the image, the middle region’s intensities differ from the neighboring regions, which results in a non-uniform, unrealistic intensity image.

The right side image in Figure 2 demonstrates the same image with the intensities normalized for all three regions using the method described in this report and accompanying computer code.

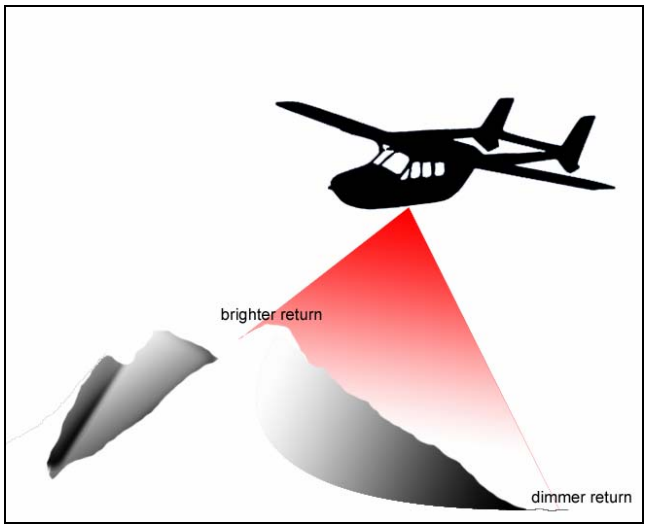


Figure 1: Changes in path length due to elevation changes.

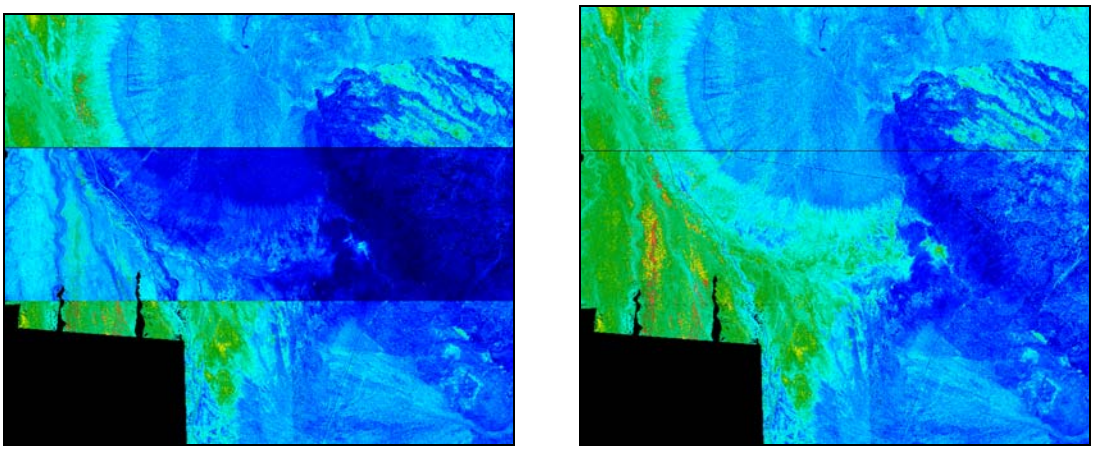


Figure 2: Mosaic of ALSM intensity image. (left) original; (right) normalized.

Scope of this report

Intensity is actually known to be a function of many variables not considered in this report. In functional terms, the lidar return intensity depends on (1) the path length, (2) the local incidence angle that the laser beam makes with the footprint surface, (3) atmospheric attenuation, (4) transmit pulse energy of the laser, and (5) the aggregate laser optics and receiver characteristics. When the same laser is used to collect data, one generally assumes variations in #5 uniformly impact the data, and we thus do not address those effect here. Without detailed information about the stability of the particular laser in use, (generally not available from the manufacturer), variations in #4 are also assumed to uniformly impact the data and thus not addressed. While variations in #2 and #3 can be addressed, we do not do so in this report. The mitigation of intensity variations due to #2 and #3 is an active research area that we and others investigate in the technical literature, for example [2, 3]. However, our purpose in posting this report is to

provide the entire lidar community with an easy-to-use 1st order correction for lidar intensity data in a state that is ready to implement (see accompanying computer code).

2.0 Normalization Routine

The intensities are normalized by an in-house routine developed in C called *norm_intensity*. The program compensates for the one over distance squared (geometric) fall-off of electromagnetic signal. Using the square of the range, the intensities are normalized to a user-defined standard range. This correction will compensate for differences in flying height, changes in ground topography (not effects on local incidence angle due to surface aspect though), and other variations in path length, such as observations taken at different scan angles. The algorithm executed by the program does not correct for effects due to beam divergence as well as the other factors that affect reflectivity as explained in section 1.0 above. The algorithm only corrects for the effects on intensity caused by variations in path length.

Algorithm Description

1. Utilizing the 9-column ALSM data file and the trajectory file for the aircraft, the GPS time tag available in both files is used to determine the two trajectory points that are closest in time to the ALSM laser shot.
2. Next, the estimated position of the aircraft at the exact time the laser shot was fired is computed by linear interpolation based on the two closest trajectory points.
3. Then, using the coordinates of the interpolated position of the aircraft and the reflected ground coordinates, the range is determined by computing the three dimensional Euclidean distance between the two.
4. Finally, the intensity is normalized by the square of the standard range as follows:
   intensity value of laser shot × (square of computed range / square of standard range)
5. The above steps are repeated for every laser shot.

Refer to corresponding *.zip folder for the C source code for the routines.

Program Execution

To run the program, the **norm_intensity** executable is called from the *MS Windows* command prompt supplied with the required inputs as follows:

```
norm_intensity trajectory_file laser_file  output_file [std range]
```

`trajectory_file` is the aircraft trajectory file in the format [time, x, y, z], `laser_file` is the ALSM data file in 9-column format [time, 2nd x, 2nd y, 2nd z, 2nd intensity, 1st x, 1st y, 1st z, 1st intensity], `output_file` is the output file name, which will be output in the same 9-column format as the ALSM input file accept that the intensities will be normalized, and `[std_range]` is an optional user-defined input for the standard range to normalize the intensities; 600m is the default and the value entered must be an integer. The trajectory file and ALSM 9-column data

file must be located in the same working directory as the executable. The output file will be output to the same directory.

Update

[22-Dec-06]

This document and associated files constitute an update to a technical report by the Geosensing Engineering and Mapping (GEM) Center at the University of Florida. That report, was entitled *Normalizing ALSM Intensities*, GEM Center Report No. Rep_2004-07-001, by B. Luzum, M. Starek, K.C. Slatton, dated July 19, 2004. That report will be superseded by this one, *Normalizing Lidar Intensities*, GEM Center Report No. Rep_2006-12-001, by B. Luzum, M. Starek, R. Kumar, K.C. Slatton, dated December 22, 2006. This update was created to accommodate the implementation of the computer code to various C-language compilers.

Most commonly used C compilers:

The most widely used C compilers are

1.) Unix and Linux based (GCC)
   a. General URL: http://www.gnu.org/
   b. Specific URL: http://www.gnu.org/software/gcc/
2.) Microsoft Visual Studio
   a. General URL: http://msdn.microsoft.com/
   b. Specific URL: http://msdn2.microsoft.com/en-us/vstudio/aa718325

A majority of C code users fall in one of the above two categories. A code successfully compiled using a GCC compiler should run on any Unix or Linux based C compiler. A third compiler that is popular is

3.) Borland
   a. General URL: http://www.codegear.com/
   b. Specific URL: http://www.codegear.com/Home/tabid/36/Default.aspx

Compilers we have used for our program:

The following three compilers are used to compile our intensity normalization program:

1.) GCC
2.) Microsoft Visual Studio 2005
3.) Borland C++ 5.5

Test conditions checked:

We tested our code in some common situations that users might encounter. These are tabulated in Table 1. Notice that in table entry #4, an error can occur if one or more rows in the trajectory file are missing. This is distinct from the case in entry #2 of a present, but empty, row.

Table 1: Commonly encountered scenarios

No.	Test case	Result
1	Correct aircraft trajectory files and laser files (ascii 9 column format) are given as input to the program. The minimum-time (smallest time tag) in the laser file should be greater than the minimum-time in trajectory file and the maximum-time of laser file should be less than the maximum-time of trajectory file.	Output file with normalized intensities was correctly generated.
2	Introduce empty rows in the trajectory file to see if an '*early end of trajectory file*' error occurs at this location.	No error occurred. An output file with normalized intensities was correctly generated.
3	Introduce empty rows in the laser file.	No error occurred. An output file with normalized intensities was correctly generated.
4	Delete some rows of the trajectory file so that a time value of the laser file does not have a reference in the trajectory file.	Error occurred. '*Early end of trajectory file! Need more trajectory data*'.
5	A new line expression between two rows is deleted so that two rows are now in a single row with double the number of columns.	No error occurred.

The error in entry #4 can be mitigated by applying the following logic. Note that no modification (including interpolating to "fill in" missing lines) of the actual trajectory file is performed.

Example: Consider two consecutive rows of data from the trajectory file. Let the time in row 1 be traj_time1 = 249566.15179 and the time in row 2 be traj_time2 = 249566.17179. We will refer to these two as 'reference time tags'. All the points in the laser input file which have a time tag greater than traj_time1 and less than traj_time2 will use these reference time tags for calculation of the normalized intensities. When the code encounters a row of data in the laser file which has a time tag greater than traj_time2, the reference time tags will be updated so that traj_time2 will now become traj_time1 and traj_time2 will be taken from the next row of the trajectory file. The new reference time tags will now be: traj_time1 = 249566.17179 and traj_time2 = 249566.19179. As this process continues, if the program doesn't encounter a next row in the trajectory file which has a time tag greater than that of the laser file, then it generates and error "*Early end of trajectory file! Need more trajectory data!*" In short, the start time of the trajectory file should be less than the start time of the laser file and the end time of the trajectory file should be greater than the start time of the laser file.

File Formats:

- The input trajectory file should contain 4 columns (time, x, y, z).
- The input laser file should either be a 9 column or a 5 column file.

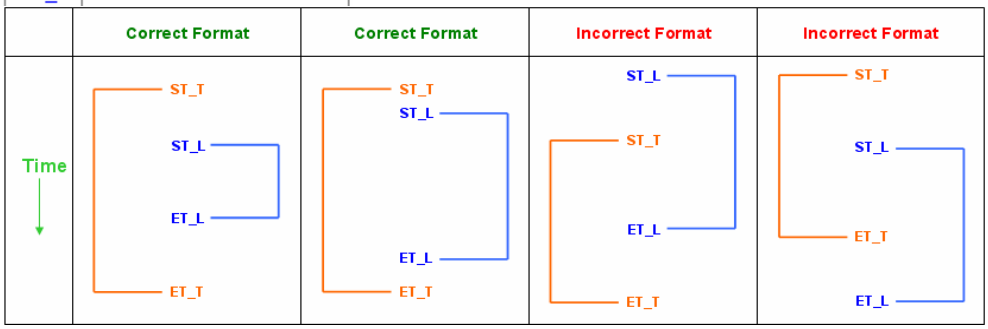
- The start time tag of the trajectory file should be less than the start time tag of the laser file.
- The end time tag of the trajectory file should be greater than the end time tag of the laser file.

A snippet of the relevant normalization code is shown below:

```
fscanf(trajfileptr, "%lf%lf%lf%lf", &trajtime, &trajx, &trajy, &trajz);
              while ((fgets(laserstring, MAXLINE, laserfileptr)) != NULL)
              {
              numcols = sscanf(laserstring,"%lf%s%lf%lf%lf%s%lf%lf%lf",
              &lasertime, x1string, &laser1y, &laser1z, &laser1i, x2string, &laser2y,
              &laser2z, &laser2i);
              while( lasertime > trajtime )
              {
                            prevtime = trajtime;
                            prevx = trajx;
                            prevy = trajy;
                            prevz = trajz;

                            if (fgets(trajstring, MAXLINE, trajfileptr) != NULL) {

                            sscanf(trajstring, "%lf%lf%lf%lf", &trajtime, &trajx,
                            &trajy, &trajz);
                            printf("\rTrajectory time: %.1lf", prevtime);
                      }
                            else
              {
                            printf("\nEarly end of trajectory file! Need more trajectory
                            data!");
                            fclose(outfileptr);
                            exit(1);
                      }
                }
              difftime = lasertime - prevtime;
              epoch = trajtime - prevtime;
              matchx = prevx + (trajx - prevx) * (difftime / epoch);
              matchy = prevy + (trajy - prevy) * (difftime / epoch);
              matchz = prevz + (trajz - prevz) * (difftime / epoch);

              length = strlen(x1string);
              offset = length - NUM_X_CHARS;
              laser1x = atof(x1string + offset);

         range1sq = pow(matchx-laser1x,2) + pow(matchy-laser1y,2) + pow(matchz-
             laser1z,2);
         norm1i = (int) floor(laser1i * (range1sq / std_range_squared) + 0.5);
>
```

Below are a few graphical examples pertaining to correct and incorrect file formats.

Legend	
ST_T	Start time of Trajectory file
ET_T	End time of Trajectory file
ST_L	Start time of Laser file
ET_L	End time of Laser file



Contents of the zip file:

These are the contents of the *.zip file associated with this report. That *.zip file is called **`Rep_2006-12-001.zip`**.

1.) **norm_intensity.exe** – This is the executable file for the program.
Usage from command prompt: *norm_intensity trajectoryfile laserfile outfile [std range]*
Note: standard range is optional. Default is 600. The column format for the aircraft trajectory file is [t x y z], which means time tag, georeferenced x-position, georeferenced y-position, georeferenced z-position. The column format for the 9-column Laser file is [t xF yF zF iF xL yL zL iL], which means time tag, georeferenced x-position of the first return, georeferenced y-position of the first return, georeferenced z-position of the first return, intensity of the first return, georeferenced x-position of the last return, georeferenced y-position of the last return, georeferenced z-position of the last return, intensity of the last return. For 5-column format, it is [t xF yF zF iF].

Note that the input laser file will nominally have either 9 columns or 5 columns, and the corresponding normalization output file will have the same number of columns, either 9 or 5. However, in a 9 column laser file, sometimes a subset of rows will have only 5 columns. This happens when the laser electronics do not record a second return. In these circumstances, the program works just fine by generating just 5 columns for such rows. The sample input and output below illustrate this case.

Laser file input into normalization code:
[Notice that the 1st and 3rd rows have 9 columns but not the second row.]

```
…
…
249533.480915  370044.81  3281574.65   42.02   114  370044.81  3281574.65  42.01  114
249533.480945  370044.87  3281574.01   41.59    74
249533.480975  370044.93  3281573.47   41.27   123  370044.93  3281573.47  41.26  122
…
…
```

Output (normalized) laser file:

```
…
…
249533.480915   370044.81  3281574.65   42.02   111   370044.81  3281574.65  42.01  111
249533.480945   370044.87  3281574.01   41.59    72
249533.480975   370044.93  3281573.47   41.27   119   370044.93  3281573.47  41.26  118
…
…
```

2.) **Folder - Source Code for three compilers**
This folder contains three more folders. The details are given in the Table 2.

Table 2: Three versions of C code supplied

Folder	Contains	How is it different from the original version?
Borland	norm_intensity.cpp	No change in the code. Only difference is that the file is saved as a .cpp file because the compiler is the *Borland C++ 5.5 compiler*.
Microsoft Visual Studio	norm_intensity.c	Some functions used in the original code are deprecated in Visual Studio 2005. So to remove the warnings, those functions are replaced by their newer counterparts. The functions updated are: • strcpy to strcpy_s • fopen to fopen_s • sscanf to sscanf_s
Unix or linux based compiler	norm_intensity.c	No change.

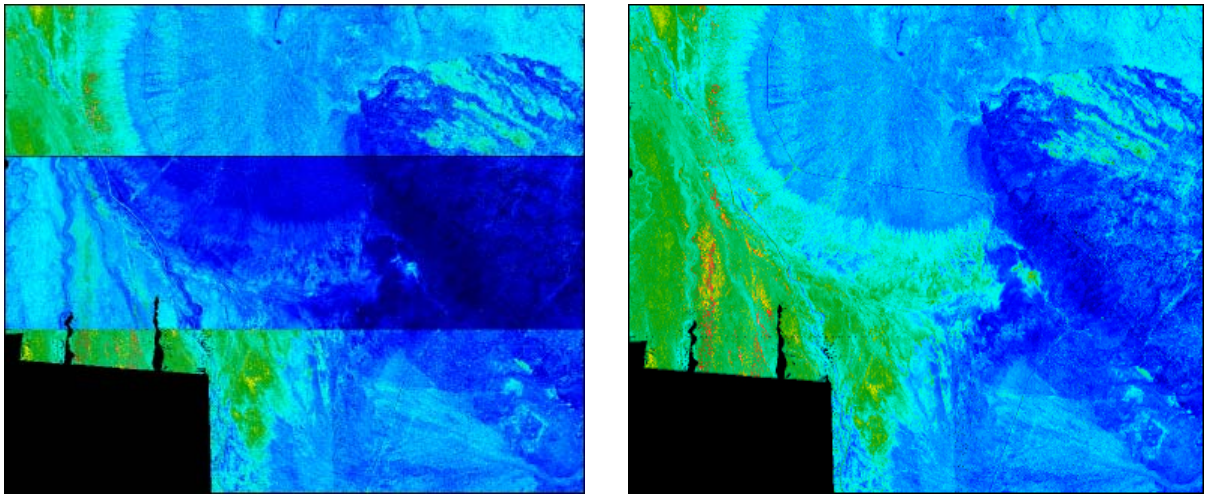
3.) **Folder - Data files**
This folder contains the sample data files used. The details are given in Table 3.

Table 3: List of data files supplied

File	Description
laser_file.all	This is the 9 column input laser file.
traj_file.txyz	This is the input trajectory file.
laser_file_normalized.all	This is the output file with the normalized intensities generated by the program.

Below are two images depicting a typical intensity variation. An image mosaic is formed by processing data from multiple flight lines over a large area in a process known as "tiling". The image on the left is composed of three tiles oriented in the horizontal direction. As can be observed in the image on the left, the middle tile's intensities differ from the neighboring tiles, which results in an erroneous intensity variation. The image on the right shows the same mosaic with the intensities normalized for each flight line.

Before Normalization **After Normalization**



Caveat for the older version (2004) of this code

The older version of this code is given in GEM Center Report No. Rep_2004-07-001. One user of that code reported getting the '*Early end of trajectory file! Need more trajectory data*' error. Now this can happen if the trajectory file is simply missing one or more rows (as indicated in Table 1). But it may also occur if the if-else condition where the '*Early end of trajectory file! Need more trajectory data*' message appears in the code uses the '**sscanf**' function, which is deprecated in Microsoft Visual Studio 2005. The replacement function for that is '**sscanf_s**'. Although the replacement functions were primarily introduced for security enhancement in the C

runtime, they also improve the handling of memory buffer overflow. Thus, in the relevant code supplied with this report (Source Code for three compilers /Microsoft Visual Studio/ norm_intensity.c) the newer function syntax is used to avoid this error.

References

[1] Optech, "About Laser Radar," *http://www.optech.on.ca*, accessed July 2004.

[2] Luzum, B.J., Slatton, K.C., Shrestha, R.L., 2005. Analysis of spatial and temporal stability of airborne laser swath mapping data in feature space. *IEEE Transactions on Geoscience and Remote Sensing* 43 (6), June 2005, 1403-1420.

[3] Jutzi, B., Stilla, U., 2006. Range determination with waveform recording laser systems using a Wiener Filter. *ISPRS Journal of Photogrammetry and Remote Sensing*, in press, doi:10.1016/j.isprsjprs.2006.09.001.