Reading Full-Waveform (FWF) data - some simple approaches with examples

B. Bookhagen and A. Rheinwalt

04-10-2019

Converting LAS 1.4 / WDP Files

Starting with LAS version 1.3, full waveform data can be stored in LAS files in accompanying WDP (Waveform Data Packet, see LINK). However, reading and accessing these files can be tricky and depends on installed software. To avoid issues with version and incompatibilities, we will use the open-source LASlib package (github to read LAS/WDP and write to an ASCII file.

A detailed description of the LAS formats LAS 1.4 and LAS 1.3.

Loading data

In order to turn a LAS+WDP file to a readable ASCII files. This assume two similarily named files exist with different extensions. Use `lsainfo` to obtain information about the LAS dataset:

```
lasinfo Haus29_ID05_FWF.las
```

and visualize the data with:

```
displaz Haus29_ID04_FWF_V13.laz Haus29_ID05_FWF_V13.laz  \
    Haus29_ID06_FWF_V13.laz
```

Convert to ASCII:

```
las2txt -i Haus29_ID05_FWF.las -o  \
    Haus29_ID04_FWF_V14_xyzinrtWV.asc -parse xyzinrtWV
#or for a LAS13 file:
las2txt -i Haus29_ID04_FWF_V13.laz -o  \
    Haus29_ID04_FWF_V13_xyzinrtWV.asc -parse xyzinrtWV
las2txt -i Haus29_ID05_FWF_V13.laz -o  \
    Haus29_ID05_FWF_V13_xyzinrtWV.asc -parse xyzinrtWV
```

```
las2txt -i Haus29_ID06_FWF_V13.laz -o  \
    Haus29_ID06_FWF_V13_xyzinrtWV.asc -parse xyzinrtWV
```

These ASCII files are available in compressed format: ***Haus29_ID04_FWF_V13_xyzinrtWV.zip****,* ***Haus29_ID05_FWF_V13_xyzinrtWV.zip****, and* ***Haus29_ID04_FWF_V13_xyzinrtWV.zip****.*

The `-parse` flag specifies how to format each each line of the ASCII file. Here, we use:

- x, z, UTM coordinates
- i - intensity
- n - number of returns for given pulse
- r - number of this return
- W - for the wavepacket information (LAS 1.3 only)
- V - for the waVeform from the *.wdp file (LAS 1.3 only)

The first line is:

```
33362192.189 5808385.047 34.648 1211 1 1 39367.146669 1 218  \
    112 13378 1.46273e-06 3.32017e-05 0.000146124 16 56 157  \
    156 158 161 166 165 162 176 226 313 415 51 0 590 626 590  \
    500 391 295 229 192 176 167 161 155 152 157 163 164 164  \
    160 165 162 161 158 156 156 158 159 157 156 162 167 163  \
    160 158 156 157 158 162 163 161 159 161 162 159 153
```

These values refer to the following items:

- **33362192.189 5808385.047 34.648 1211 1 1** - UTM-X, Y, Z, intensity, number of returns for a given pulse and number of this return (1/1)
- **39367.146669 1** - GPS Time, # of wdp descriptor,
- **218 112 13378** - byte offset to waveform data, number of samples in bytes (56*2=112), point (UTM-X, Y, Z) is 13378 picoseconds after the start of waypoint sampling
- **1.46273e-06 3.32017e-05 0.000146124** - direction vector pointing to the aircraft (dx=1.46273e-06, dy=3.32017e-05,dz=0.000146124)
- **16** - 16bit sampling
- **56** - nr. of samples
- **157 156 158 161 166 165 162 176 226 313 415 51 0 590 626 590 500 391 295 229 192 176 167 161 155 152 157 163 164 164 160 165 162 161 158 156 156 158 159 157 156 162 167 163 160 158 156 157 158 162 163 161 159 161 162 159 153** FWF intensity data

Visualizing FWF data with displaz

We are using a combination of Python codes and displaz to visualize the data. This is in an interactive environment that will allow you to click on a point in displaz and see the corresponding wavepoint in displaz (visualized by using the sampling frequency and intensity) and a plot showing elevation asl and intensity. The code fwf_selector_annotate.py contains this setup.

All relevant data are in the subfolder examples/fwf-lidar-displaz-examples. Please make sure to run the fwf_selector_annotate.py from this folder, as it needs to read the FWF and LAS data.

Extracting profiles and performing spline fitting

In this step, we use a simple universal spline fitting approach to smooth the point clouds and find peaks using a peak finding function. The code fwf_peaks.py shows the details of the necessary steps.

Using identified peak to generate a denser PC

Using the above (fwf_peaks.py) described approach, we can identify all peaks in FWF data and generate points from them. See fwf_densify.py for details.

In order to avoid cluttering, we apply an intensity filter of 50 (only points with intensities above 50 are then converted into a point).

The relevant options are on line 44. Here you may consider removing the `prominence` option.

```
peaks, properties = find_peaks(usp(s), prominence = 5)
```

to

```
peaks, properties = find_peaks(usp(s))
```

and setting the intensity threshold at lines 62/63 (using a lower value here will generate significantly more points):

```
ptsi = ptsi[ampi > 50]
ampi = ampi[ampi > 50]
```

Fitting Gaussian functions to FWF data

Using lmfit, because it is easy to use and very flexible.

```
conda install -c conda-forge lmfit
pip install progressbar
```

An example of fitting Gaussian functions to a FWF signal is in fwf_peaks_with_Gaussian_fitting.py.

Note: Depending on your setup, you may have to use LAS (and not LAZ) files. Use `laszip` *to convert:*

```
laszip -i Haus29_ID04_FWF_V13.laz -olas -o  \
    Haus29_ID04_FWF_V13.las
laszip -i Haus29_ID05_FWF_V13.laz -olas -o  \
    Haus29_ID05_FWF_V13.las
laszip -i Haus29_ID05_FWF_V13.laz -olas -o  \
    Haus29_ID05_FWF_V13.las
```

The code fwf_gaussian_fit.py performs a simple Gaussian Function fit for all FWF data contained in the example files. You can run this on all three flight lines.

Converting the examples (will take some time - fitting not paralleled yet). Run this where the data are stored (change *.LAZ to* .LAS if necessary).

```
python fwf_gaussian_fit.py Haus29_ID04_FWF_V13_xyzinrtWV.asc  \
    Haus29_ID04_FWF_V13.laz
python fwf_gaussian_fit.py Haus29_ID05_FWF_V13_xyzinrtWV.asc  \
    Haus29_ID05_FWF_V13.laz
python fwf_gaussian_fit.py Haus29_ID06_FWF_V13_xyzinrtWV.asc  \
    Haus29_ID06_FWF_V13.laz
```

In a last step, you could combine output files from different flight lines via `lasmerge` or `pdal`:

```
lasmerge -i *_amp.las -olaz -o Haus29_ID040506_FWF_V13_amp.laz
lasmerge -i *_width.las -olaz -o  \
    Haus29_ID040506_FWF_V13_width.laz
lasmerge -i Haus29_ID*_FWF_V13.las -o  \
    Haus29_ID040506_FWF_V13_org.laz
```

PulseWaves

Additional information can be found in a Google Groups and github. In a conda environment, you can install `pulsewaves` libraries and additional tools from github with:

```
conda install -c conda-forge pulsewaves
git clone https://github.com/PulseWaves/PulseWaves.git
```

The WaveTools binaries (for Windows only) are in PulseWaves/PulseTools and can be called directly. In Linux/Mac OSX you could use `wine` with:

```
wine ~/PulseWaves/PulseTools/pulseinfo.exe <input_file.plz>
```

An example to access NEON files (from Linux):

```
wine ~/PulseWaves/PulseTools/pulseinfo.exe  \
    NEON_D17_SJER_DP1_L001-1_2019032516_translate.plz
```

Example for the LAS/WDP to PulseWave conversion:

First ensure that you have LAS V1.3 (not 1.4)

```
#FlightLine 04
wine /opt/LAStools/bin/las2las.exe -set_version 1.3 -i  \
    Haus29_ID04_FWF.las -o Haus29_ID04_FWF_V13.laz
cp Haus29_ID04_FWF.wdp Haus29_ID04_FWF_V13.wdp

#FlightLine 05
wine /opt/LAStools/bin/las2las.exe -set_version 1.3 -i  \
    Haus29_ID05_FWF.las -o Haus29_ID05_FWF_V13.laz
cp Haus29_ID05_FWF.wdp Haus29_ID05_FWF_V13.wdp

#FlightLine 06
wine /opt/LAStools/bin/las2las.exe -set_version 1.3 -i  \
    Haus29_ID06_FWF.las -o Haus29_ID06_FWF_V13.laz
cp Haus29_ID06_FWF.wdp Haus29_ID06_FWF_V13.wdp
cp -rv Haus29_ID04_FWF_V13.laz Haus29_ID04_FWF_V13.wdp  \
    Haus29_ID05_FWF_V13.laz Haus29_ID05_FWF_V13.wdp  \
    Haus29_ID06_FWF_V13.laz Haus29_ID06_FWF_V13.wdp V13
```

Convert to PulseWaves with:

```
wine ~/PulseWaves/PulseTools/pulse2pulse.exe -i  \
    Haus29_ID04_FWF_V13.laz -o Haus29_ID04_FWF_V13.pls
```

Generate an ASCII file from PulseWave files:

```
wine ~/PulseWaves/PulseTools/pulse2pulse.exe -i  \
    Haus29_ID04_FWF_V13.pls -o Haus29_ID04_FWF_V13.asc
```

Convert a FWF file to a LAZ file with new thresholds:

```
wine ~/PulseWaves/PulseTools/pulseextract.exe -i  \
    Haus29_ID04_FWF_V13.pls -o  \
    Haus29_ID04_FWF_V13_th200_d10.laz -threshold 200 -decay 10
```