Using free and/or open source tools to build workflows to manipulate and process LiDAR data:

GDAL

Christopher Crosby

April 30th, 2008

Exercises in this series:
1. LAStools – LAS binary points to ascii points
2. GEON Points2Grid (see Points2Grid instructions or integrated help) – DEM generation from ascii point data
3. **GDAL – Raster file format conversion and projection**

The goal of this suite of exercises is demonstrate how you can use a set of free and/or open source tools to build workflows to manipulate LiDAR data from point cloud formats to meaningful grid products. Many of the tools illustrated here require that they be executed from the command line and are initially less user friendly since they lack a graphical user interface (GUI). However, once comfortable with these tools you will hopefully see how powerful they can be. In order to manage large dataset or batch process data, these tools could be scripted to semi-automate processing.

Potentially helpful basic MS-DOS navigation:

Start > Run > `cmd` to get command line window

- **`cd directory`** - go into directory
- **`cd ..`** - up a directory
- **`cd ../..`** – up two directories
- **`dir`** to list contents of a directory
- **`help command`** - lists flags and usage
- **`exit`** - …
- **`tab`** – auto completes

GDAL – Geospatial Data Abstraction Library

GDAL Docs: http://www.gdal.org/gdal_utilities.html
Very nice tutorial: http://trac.osgeo.org/gdal/wiki/UserDocs/RasterProcTutorial

Launch FWTools Shell to get command line interface.

Use MS-DOS navigation tricks above to get to directory containing raster data of interest. In the case of this tutorial, we will assume that you have created DEMs using GEON Points2Grid and that those grids are in the same directory that initially contained the LAS point cloud data discussed earlier in this exercise series.

First we will use the utility GDALInfo to get some information about the ascii grids we created with Points2Grid. Type:

Gdalinfo *filename*
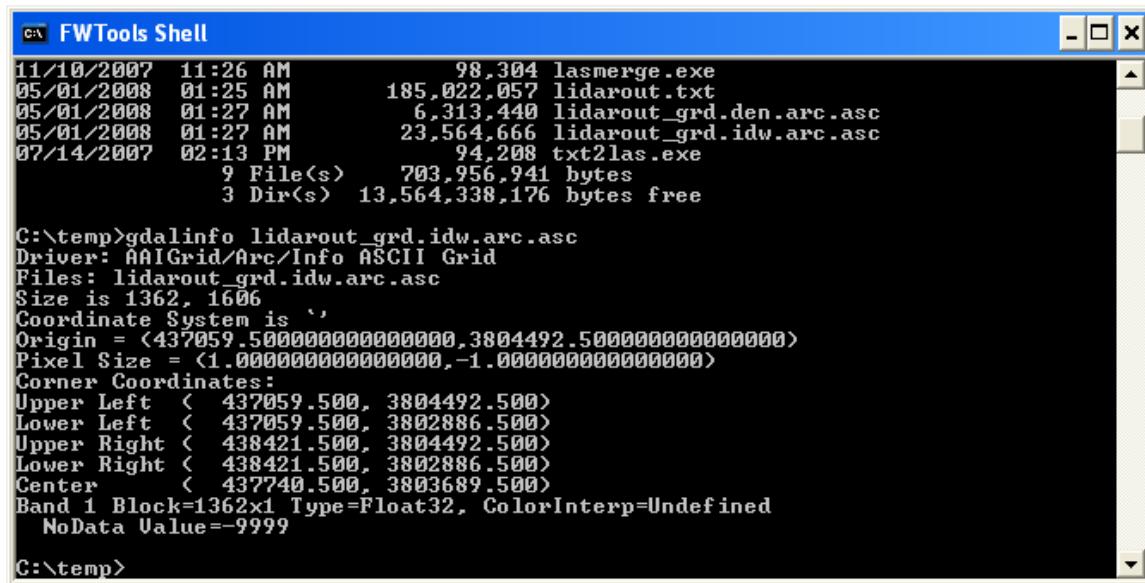
```
FWTools Shell                                                    _ □ ×
11/10/2007  11:26 AM              98,304 lasmerge.exe
05/01/2008  01:25 AM         185,022,057 lidarout.txt
05/01/2008  01:27 AM           6,313,440 lidarout_grd.den.arc.asc
05/01/2008  01:27 AM          23,564,666 lidarout_grd.idw.arc.asc
07/14/2007  02:13 PM              94,208 txt2las.exe
               9 File(s)        703,956,941 bytes
               3 Dir(s)   13,564,338,176 bytes free

C:\temp>gdalinfo lidarout_grd.idw.arc.asc
Driver: AAIGrid/Arc/Info ASCII Grid
Files: lidarout_grd.idw.arc.asc
Size is 1362, 1606
Coordinate System is `'
Origin = (437059.500000000000000,3804492.500000000000000)
Pixel Size = (1.000000000000000,-1.000000000000000)
Corner Coordinates:
Upper Left  (  437059.500, 3804492.500)
Lower Left  (  437059.500, 3802886.500)
Upper Right (  438421.500, 3804492.500)
Lower Right (  438421.500, 3802886.500)
Center      (  437740.500, 3803689.500)
Band 1 Block=1362x1 Type=Float32, ColorInterp=Undefined
  NoData Value=-9999

C:\temp>
```

Above shows the result of running gdal info on the ascii grid file. Note that information about grid extent, pixel size, grid type (floating point), NoData Value are all given. Because P2G does not assign a projection to the grids, there is no projection given by gdalinfo

Typing:
Gdalinfo –mm *filename*

Forces a bit more information including file type (see first
line – AAIGrid/Arc/Info ASCII Grid):
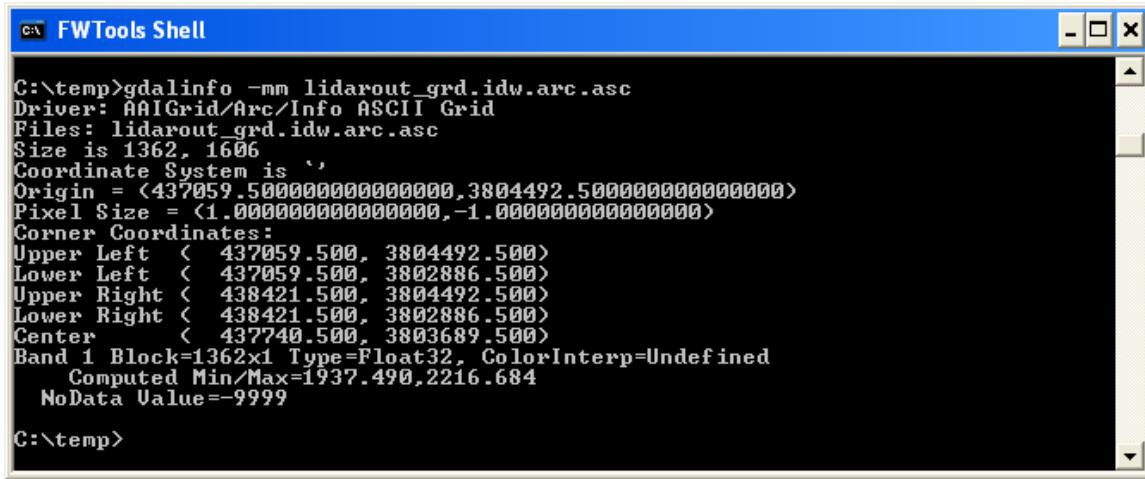
```
C:\temp>gdalinfo -mm lidarout_grd.idw.arc.asc
Driver: AAIGrid/Arc/Info ASCII Grid
Files: lidarout_grd.idw.arc.asc
Size is 1362, 1606
Coordinate System is `'
Origin = (437059.500000000000000,3804492.500000000000000)
Pixel Size = (1.000000000000000,-1.000000000000000)
Corner Coordinates:
Upper Left  (   437059.500, 3804492.500)
Lower Left  (   437059.500, 3802886.500)
Upper Right (   438421.500, 3804492.500)
Lower Right (   438421.500, 3802886.500)
Center      (   437740.500, 3803689.500)
Band 1 Block=1362x1 Type=Float32, ColorInterp=Undefined
    Computed Min/Max=1937.490,2216.684
  NoData Value=-9999

C:\temp>
```

Now, open this ascii file into ArcGIS save out an ESRI binary grid file and then generate a hillshade from that file (you may already have a binary grid file and hillshade available. If so, continue).

Now, use gdalinfo again on the binary grid to see what additional information is provided:

Gdalinfo –mm *filename*

```
C:\temp>gdalinfo -mm wallace_clip4
Driver: AIG/Arc/Info Binary Grid
Files: wallace_clip4
       wallace_clip4\dblbnd.adf
       wallace_clip4\hdr.adf
       wallace_clip4\metadata.xml
       wallace_clip4\prj.adf
       wallace_clip4\sta.adf
       wallace_clip4\w001001.adf
       wallace_clip4\w001001x.adf
Size is 4298, 1422
Coordinate System is:
PROJCS["UTM Zone 11, Northern Hemisphere",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            TOWGS84[0,0,0,0,0,0,0],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0,
            AUTHORITY["EPSG","8901"]],
        UNIT["degree",0.0174532925199433,
            AUTHORITY["EPSG","9108"]],
        AXIS["Lat",NORTH],
        AXIS["Long",EAST],
        AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",-117],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["METERS",1]]
Origin = (243785.221500000010000,3904365.713299999900000)
Pixel Size = (0.500000000000000,-0.500000000000000)
Corner Coordinates:
Upper Left  (  243785.222, 3904365.713) (119d48'57.53"W, 35d14'58.78"N)
Lower Left  (  243785.222, 3903654.713) (119d48'56.74"W, 35d14'35.73"N)
Upper Right (  245934.222, 3904365.713) (119d47'32.60"W, 35d15'0.75"N)
Lower Right (  245934.222, 3903654.713) (119d47'31.81"W, 35d14'37.70"N)
Center      (  244859.722, 3904010.213) (119d48'14.67"W, 35d14'48.24"N)
Band 1 Block=512x4 Type=Float32, ColorInterp=Undefined
  Min=605.140 Max=667.252   Computed Min/Max=605.140,667.252
  NoData Value=-3.4028234663852886e+038

C:\temp>
```

Notice that gdalinfo can provide much more information about the binary grid than it can about the ascii grid, including projection and files that compose the grid.

To convert file formats, we can use the gdal_translate command. The default output format is GeoTIFF. Type:

gdal_translate *filename newfile.tif*

NOTE: To generate decent looking tiff, jpeg or other images you'll want to use a hillshade as input, NOT a DEM – these graphics formats don't easily floating point elevation data..

Open the GeoTIFF in a graphics program to view.

To override the default and write another format, use the –of flag to define output file format:

gdal_translate -of JPEG *filename newfile.jpg*

Again, open jpeg to view results.

Finally, the gdalwarp command will allow you to project raster datasets. Projection is best accommodated in gdal with EPSG codes – these are numeric codes that correspond to coordinate systems of geospatial data. To find the appropriate EPSG code for your data, visit: http://spatialreference.org/

In the following example, we reproject convert file formats simultaneously. The "-s_srs" flag identifies input coordinate system, the "-t_srs" flag identifies output coordinate system, and while we are at it, we also convert the file to a geotiff using a cubic resampling method:

```
gdalwarp -s_srs EPSG:32611 -t_srs EPSG:4326 -r cubic
filename filename.tif
```