Informatics 590 – Final Project

Kevin Russell

**From LAS to DEM: Large-scale lidar terrain processing**

Define problem: The state of Indiana produced a statewide lidar dataset from surveys performed in 2011, 2012, and 2013. Lidar datasets are huge collections of three-dimensional point clouds that can be processed in a variety of ways to produce terrain models of ground surfaces, tree canopies, surface reflectivity, and building heights for a few examples. Lidar data is often consumed with desktop GIS applications and basic computing hardware. I propose to use big data methods to process and analyze the raw lidar data to generate a series of digital terrain surfaces as raster grids.

Data properties: 1 county (Benton County) is contained in over 500 .las files that total 38.8 GB in size. The when one includes the other counties of Indiana the total size of the .las dataset increases to 3.24 TB.

One las file can contain on average between 1,000,000 and 5,500,000 points in 3D space. The Benton County contains over 1.5 billion points.

Demonstrate goals and objectives: My goal is to process the raw lidar data into a series of digital elevation models to visualize the hill shade, slope, and aspect of the lidar terrain. To so I will need to read the raw .las data and convert it to a delimited text file. I accomplish this within a TORQUE batch script to Karst. The data from the converted text files will then be programmatically added to a numpy array from which a number of mathematical operations in a python script to derive digital terrain models of the slope surface, hill shade, and aspect.

Provide tasks to complete project deliverables:

Define Software: IU Karst with Python 2.7 and Numpy and LAStools 1.0

LAStools is a batch-scritable multicore command line software suite to process and analyze lidar data (Isenburg 2015). I used two packages from the lastools library: las2txt and mergelas.

Provide list of techniques used: Before taking on this project I normally worked on python from within ESRI ArcGIS console. This project was the first time I’ve worked heavily in the UNIX shell and with python outside of ArcGIS. My coding process was very iterative, beginning first with test scripts in Canopy’s console, followed by interactive processing on Kasrt, and finally completed with batch scripts on Karst. At the onset, I limited my dataset to one .las file to increase processing times while troubleshooting. As I developed my workflow I increased the dataset to include the entire data for Benton County, Indiana.

Processing the Benton County data from .las to the .txt format required on average 50 minutes of processing on Karst. At the beginning of my research my scripts were terminated when I exceeded the default walltime for processing on the home node on Karst. This required me to learn how to create a TORQUE script for submission of my scripts. This ended up being quite a struggle due to some unknown windows formatting characters that corrupted my first attempts. Finally I learned how to use VI within Karst, rewrote my PBS scripts and finally had successful batch processes run.

The following outline describes my coding process.

Use Putty to connect to Karst. Use WinSCP to ftp a batch of data to a folder on my Karst account. Begin by loading LAS tools to the Karst shell.

Module load lastools/1.0

Use LAS tools to parse .las files to text and parse x, y, z, coordinate fields.

las2txt –i \*.las –parse xyz –sep semicolon –header pound

A review of a sample file reveals the header contains 19 lines of comments. To load the data into a numpy array I need to skip the commented lines so that only the data is loaded into the array.

["# file signature: 'LASF'\n", '# file source ID: 0\n', '# reserved (global encoding):1\n', "# project ID GUID data 1-4: -2101939227 13949 18470 '\xa4\xd0,\xefbOu\xd1'\n", '# version major.minor: 1.2\n', "# system\_identifier: 'NIIRS10'\n", "# generating\_software: 'LIDAR1 tiled'\n", '# file creation day/year: 305/2013\n', '# header size 227\n', '# offset to point data 18588\n', '# number var. length records 5\n', '# point data format 1\n', '# point data record length 28\n', '# number of point records 977493\n', '# number of points by return 976831 625 37 0 0\n', '# scale factor x y z 0.01 0.01 0.01\n', '# offset x y z 0 0 0\n', '# min x y z 2828356.54 1900000.00 733.02\n', '# max x y z 2829999.98 1904999.95 795.12\n']

Figure 1. Sample of header information from lidar data.

First I had to create some variables. I used an example from Lawhead (2013) as a starting point for the python geoprocessing script and edited the code to fit my project needs.

sourceDirectory = "/N/dc2/scratch/kevrusse/output"

outputDirectory = "/N/dc2/scratch/kevrusse/output/grids/"

fileList = glob(sourceDirectory + '/\*.txt')

fileLAS = ""

delimiter = ';'

comments = '#'

Then I loop through the file list and load the text files into a numpy array. I use a numpy array instead of a python list because at this state the data is a set of billions or trillions of points and numpy arrays are more efficient in storing, reading, and writing data than python lists (Lawhead 2013).

prelimArray = [np.loadtxt(fileLAS, comments=comments, delimiter=delimiter, skiprows=19) for fileLAS in fileList]

myArray = np.concatenate(prelimArray)

To perform slope, hill shade, and aspect functions over the new numpy array I had to first, create variables for a few geoprocessing and file path parameters.

# File name of the slope grid

slopegrid = "slope.asc"

# File name of the aspect grid

aspectgrid = "aspect.asc"

# Output file name for shaded relief

shadegrid = "relief.asc"

## Shaded elevation parameters

# Sun direction

azimuth = 315.0

# Sun angle

altitude = 45.0

# Elevation exageration

z = 1.0

# Resolution

scale = 1.0

# No data value for output

NODATA = -9999

# Needed for numpy conversions

deg2rad = 3.141592653589793 / 180.0

rad2deg = 180.0 / 3.141592653589793

I then need to create a search window from which to analyze the data. The window is a 3 x 3 grid that ignores the 2 pixels at the border where no data cells typically occur.

window = []

for row in range(3):

for col in range(3):

window.append(myArray[row:(row + myArray.shape[0] -2), col:(col + myArray.shape[1] - 2)])

And then process the raster cells

x = ((z \* window[0] + z \* window[3] + z \* window[3] + z \* window[6]) - (z \* window[2] + z \* window[5] + z \* window[5] + z \* window[8])) / (8.0 \* xres \* scale);

y = (( z \* window[6] + z \* window[7] + z \* window[7] + z \* window[8]) - (z \* window[0] + z \* window1] + z \* window[ 1] + z \* window[2])) / (8.0 \* yres \* scale);

calculate slope

slope = 90.0 - np.arctan( np.sqrt( x\* x + y\* y)) \* rad2deg

calculate aspect

aspect = np.arctan2( x, y)

calculate shaded relief

shaded = np.sin( altitude \* deg2rad) \* np.sin( slope \* deg2rad) + np.cos( altitude \* deg2rad) \* np.cos( slope \* deg2rad) \* np.cos(( azimuth - 90.0) \* deg2rad - aspect);

shaded = shaded \* 255

Finally, here are examples of the batch scripts I created to run the jobs on Karst.

#!/bin/sh

#PBS -N ProcessLas2Txt

#PBS -l nodes=1:ppn=4,pmem=4gb,walltime=01:00:00

#PBS -M kevrusse@indiana.edu

#PBS -m ae

#PBS -o output.$PBS\_JOBID

module load lastools/1.0

cd /N/u/kevrusse/Karst/data/Benton

las2txt -i \*.las -parse xyz -sep semicolon -header pound -odir /N/dc2/scratch/kevrusse/output

and,

#!/bin/bash

#PBS -k o

#PBS -M kevrusse@indiana.edu

#PBS -m abe

#PBS -N ProcessText2DEM

#PBS -l nodes=1:ppn=2,pmem=2gb,walltime=04:0:00

#PBS -o output.$PBS\_JOBID

cd /N/u/kevrusse/Karst/scripts

python ProcessText2DEM.py

Provide recommendations for future improvements:

After completing this process with the methods outlined above, I have recently discovered a python library called Laspy. Instead of converting the raw las data into a text format, laspy reads, writes, or modifies las data (Brown 2015). The geoprocessing operations are basically the same but I expect laspy to perform much more efficiently. Unfortunately, this package is not currently available on the Karst system and I have not been able to install myself with pip install commands. I have a request submitted to the Karst software team to install the package, however I have no idea as to when my request may be approved.

Once I am able to process the data through laspy, I plan to increase the dataset to whole state of Indiana. Once I am able to process the entire state I intend to explore how to further process the digital elevation models with GDAL library on Karst. I hope to use this dataset to refine image caching workflows on Karst before the state-wide orthoimagery is available in 2016.

References:

Brown, G. (2015) Laspy 1.3.2, obtained from https://pypi.python.org/pypi/laspy

Isenburg, M. (2015) LAStools - efficient LiDAR processing software, (version 141017, unlicensed), obtained from <http://rapidlasso.com/LAStools>

Lawhead, J. (2013). Learning Geospatial Analysis with Python, Packt Publishing Ltd.