VoxGen

A C++ Program to Voxelize LAS Files

Megan Kress kressmeg@msu.edu

Andrew Finley finleya@msu.edu

April 27, 2015

1 Introduction to LAS Files

LAS files[6] contain data from LiDAR point clouds. Each point has a set of properties described by the LAS file:

- Classification: Points may or may not be classified. Classifications include unclassified, ground, low vegetation, medium vegetation, high vegetation, building, and water.
- Scan Angle
- Return Number: For each pulse there are 1-5 returns, so a point may be described by its return number for the pulse in which it was scanned.

2 Libraries

2.1 Boost

Boost[1] is a library that in VoxGen allows processing of an entire directory at once using its filesystem[11] class. A user can choose whether to run VoxGen on a directory or on a single file.

2.2 libLAS

libLAS[9] is a C/C++ library that can read and write LAS files.

2.3 kdtree

kdtree[5] is a simple C library that creates and iterates through kd-trees. Kd-trees are k-dimensional binary search trees. In this case, our kd-tree operates in three dimensions corresponding to the x, y, and z values for each point in the LAS file.

The kdtree library has a function to add points to a kd-tree object along with a function to determine the points within a given range of some point.

2.4 GDAL

GDAL[3] is " a translator library for raster and vector geospatial data formats." In this program, GDAL is used to read the .tif files providing metrics for the LiDAR data. The GDAL API[2] provides functions to open files and get data from the raster for an individual pixel.

3 Custom Classes

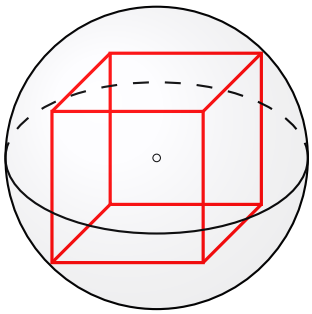
3.1 Point

The simple Point class contains the x, y, and z coordinates for a point.

3.2 Voxel

The Voxel class contains the x, y, and z coordinates for the center of the voxel along with the number of points in the voxel and a vector of the points.

This class also contains two functions: inVox() and trimVox(). inVox() takes in as arguments a point or 3D coordinates and returns whether or not that point is within the voxel. trimVox() is called on a voxel in VoxGen because the kdtree library's function to determine the points within a given range of a point returns points in a sphere around the voxel, and the Voxel class only wants points within the rectangular prism that is the voxel.



3.3 VoxCol

The VoxCol class simply contains the x and y center coordinates for a given voxel column and a vector of voxels (Voxel objects) that are within that column.

3.4 VoxData

VoxData is a class that summarizes the voxel information for a given LAS file. It includes a vector of all the voxel columns (VoxCol objects), the number of points in all the voxels, the number of voxel columns, and the number of voxels in each voxel column.

VoxData also includes methods to read and write files.

3.5 vgpar

vgpar requires a text document parameter file of relevant information for the VoxGen program (see below).

The vgpar class constructor takes in a string of the text file location, and the class includes functions getString() and getNum(), which utilize a map to obtain a value referenced in the paragraph.

For example, getString("filter") returns the string "n," which is then used as user input in VoxGen to determine that the user does not wish to filter points from the LAS file.

Example VoxGenPar Text File

```
##A paragraph of relevant information for VoxGen Program

fileORdir file

inFile /home/megan/Data/Duke/Duke_AM_24Oct2013_c0r0.las
inDir /home/megan/Data/Duke
```

```
#Filtering y or n?
filter n

#If filtering on, select parameters
#Classifications A: All, U: Unclassified, G: Ground,
#L: Low Vegetation, M: Medium Vegetation,
#H: High Vegetation, B: Building, W: Water
classes A
returns A
angle 20

#Voxel Dimensions
base 10
height 10

#Initial Coordinates(all 0s for min/max from file)
x1 0
x2 0

y1 0
y2 0

#Look for line intersections y or n?
```

```
lines n



#R output y or n?
routput y
outName testR
outDirectory /home/megan/Downloads

#VoxData File y or n? The file index number will be
#appended to the title in the given directory.
voxdata y
vdataTitle voxData
vdataDir /home/megan/Downloads
```

4 Functions

- **Voxelize**: VoxGen takes in user input in the form of a text file. Based on the LAS file and filters specified, VoxGen divides the points from the LAS file into voxels of specified dimensions.

- **Output R File**: If the user chooses to create an R file, VoxGen will create an R source file, which will input various matrices into the R

workspace that contain voxel information.

- **VoxData File**: VoxGen reads and writes text files to create or save VoxData object information.

- **Voxel Column Metrics Flat File**: VoxGen writes a text file in which each row represents data corresponding to a voxel column's metrics.

- **Voxel Column Histogram Flat File**: VoxGen writes a text file in which each row has the number of points in each voxel column's voxels.

See appendix for descriptions of the two flat files.

5 VoxGenR

VoxGenR is an R package that can be used for basic analysis of voxel columns created in VoxGen. Note: VoxGenPar must have the R out file enabled in order to create an R file that can be imported by VoxGenR.

The source package for VoxGenR is located in the VoxGen/VoxGenR directory. It can be loaded into R by running R and entering the following commands:

```
> install.packages("/path/to/VoxGen/VoxGenR/VoxGenR_1.0.tar.gz",
    repos = NULL, type="source")
> library(VoxGenR)
> importData("/path/to/rOutFile.R")
```

R's environment now contains a list of matrices called VoxList, in which each index represents a four column voxel column matrix, wherein each row represents a voxel. Column 1 is the x center of the voxel, 2 the y center, 3 the z center, and 4 the number of points in the voxel. Here is an example of accessing a list index in VoxList:

```
 > VoxList[[83]]
         [,1]    [,2] [,3] [,4]
[1,] 363515.5 4306537 10.2   52
[2,] 363515.5 4306537 15.2    0
[3,] 363515.5 4306537 20.2    0
[4,] 363515.5 4306537 25.2    0
[5,] 363515.5 4306537 30.2    0
[6,] 363515.5 4306537 35.2    0
```

This shows that the first voxel in voxel column 83 contains 52 points.
The VoxGenR functions selectPoints() and heatMapAll() may now be run:

selectPoints(): This function displays the centers of all the voxel columns

and allows the user to select certain points. The columns that are selected will be displayed as histograms individually along with a heat map of the voxel point densities for all columns selected.

heatMapAll(): This function displays a heat map of all the voxel columns next to each other to gain a basic understanding of the points' height distributions for the LAS file used in VoxGen.

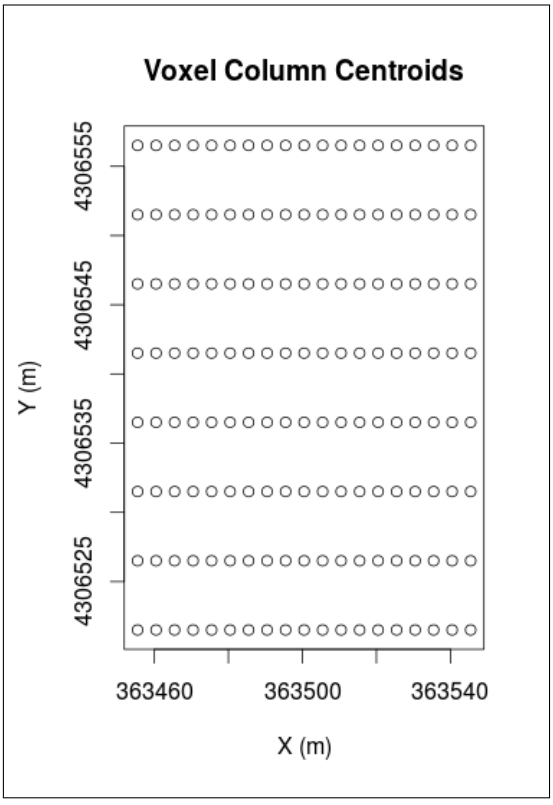


Figure 1: Voxel Column Centers Displayed after importData()

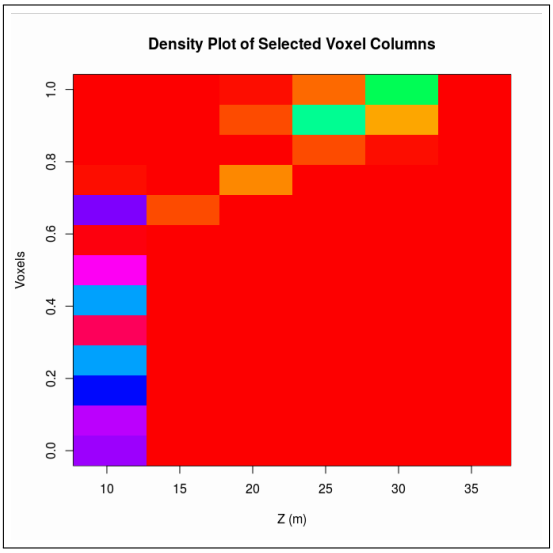


Figure 2: Rainbow Heat Map Displayed after selectPoints() - Red is Least Points

6 Quality Assurance

QGIS was used to check the accuracy of the output lidar metric flat files. The flat file was loaded into a spreadsheet program, and four metric raster layers were loaded into QGIS. A random number generated selected which rows from the flat file to examine. "ZoomToCoordinates" found the point in QGIS, and the "Identify Features" tool displayed the metric values at that point.

Acquisition: 20140712_brendon_1a

Type: mosaic.tif files

Metrics: tree_d6, ground_slope, shrub_mean, all_p50

Centroids: 60

Acquisition: 20140714_tanana_flats
Type: individual .tif files per segment
Metrics: pulse_density, tree_mean, all_d9
Centroids: 40

All of the 100 centroids had data in the flat file that matched the observed data in QGIS.

An R script (see appendix, VoxGenTest.R) was utilized to check the accuracy of the output voxel histogram flat files. The script loads the .txt file conversion of a given split LAS file along with the output histogram flat file. Next, based on the centers of each voxel in the flat file, the script determines how many points in the LAS file lie within the voxel. The script then counts the number of mismatches between numbers of points.

Running this script gave an accuracy of **100%** for 1.48 million voxels.

7 Additional Programs

The VoxGen directory contains three additional programs to aid in the processing of LiDAR data.

7.1 MinMaxZ

The purpose of MinMaxZ is to obtain the minimum and maximum z values for all LAS files in a certain acqusition. This allows one to run VoxGen with consistent voxel heights for all segments. MinMaxZ is executed on the directory containing all LAS files for a given acquisition, and it outputs a text file "range.txt" that contains the bounds for all LAS files in the directory.

7.2 VGProj

VGProj simply takes in a LAS file as an argument and prints its Spatial Reference. The user may then document the projection for a given acquisition, which is essential for reprojecting the lidar metric flat files when processing them with a shapefile or raster with a different spatial reference.

7.3 SplitLas

SplitLas tiles LAS files into manageable smaller LAS files so that VoxGen may be run more efficiently.

8 Conclusion

One can use VoxGen for two types of analysis: centroid metrics and voxel examination. At this point, only the data output from the lidar metric files has been explored, but significant patterns have already emerged from examing these flat files concurrently with fire history shapefiles.

For example, the following plot shows a positive relationship between the years since burned("YSB") and the mean Fraction of first returns intercepted by trees ("tree_fcover"). VoxGen had output the tree_fcover metric for every centroid in the LAS files, and by accessing fire history data for the centroids' coordinates, we were able to see a relationship between when an area burned and how dense the foliage is in that area.

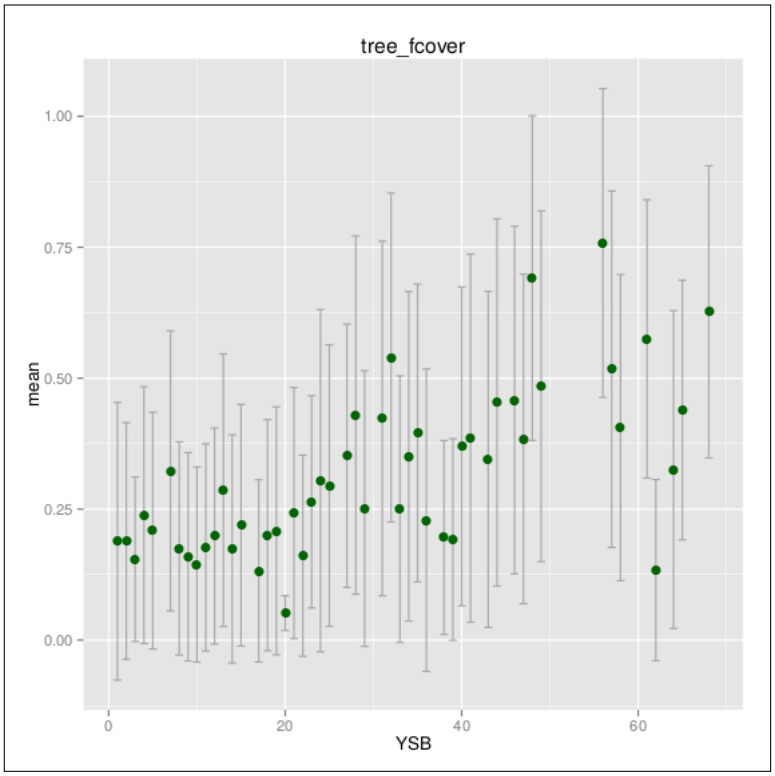


Figure 3: Analysis of tree_fcover based on lidar metric flat files output by VoxGen.

Further analysis may be done on the histogram output files. By examining a row of the histogram, one can compare densities of returns at a given height, allowing some idea of the vertical composition of the forest. In addition, one of the issues with the LiDAR data is the varied angles of the laser, which may affect the returned data. We are currently discussing on a 3D ray tracing algorithm that would be able to return the voxels through which a given line would pass.

VoxGen itself may be improved by exploring different data structures to

store points and increasing the efficiency of voxel creation. An improvement in the program, for example, would be a library for a data structure with an efficient delete/remove method so that once a point is categorized within a voxel, it is no longer included in the nearest neighbor search for the remaining voxels.

9 Acknowledgements

NASA[4] funded the research to create this program.

10 Appendix

10.1 VoxGenTest.R

```
require(data.table)

# Load the txt file of the LAS file as pts
pts <- "/home/megan/Downloads/Hist/c0r0_24.txt"

# Load the histogram flat file as vox
vox <- "/home/megan/Downloads/Hist/histff_20140728_mack
    ↪ _dome_c0r0-24"

# Convert data sets into data frames
```

```
10 pts <- fread(pts)
11 setnames(pts,c("x","y","z"))
12 pts <- as.data.frame(pts)
13
14 pts <- cbind("index" = c(1:nrow(pts)), pts)
15
16 vox <- fread(vox)
17
18 voxels <- vox[1, ]
19 voxels <- as.data.frame(voxels)
20
21 voxels <- as.numeric(voxels[,5:ncol(voxels)])
22
23 vox <- vox[2:nrow(vox),]
24 vox <- as.data.frame(vox)
25
26 # Set the min and max z values for the voxels
27 minZ = 280.66
28 maxZ = 476.88
29
30 mistakes <- 0
31 mistakesVect <- NULL
32
```

```
33 h = 0.3
34 hh = 0.15
35 w = 13
36 hw = 6.5
37
38 voxct <- ncol(vox) - 4
39 rows <- nrow(vox)
40
41
42 listPts <- NULL
43
44 remaining <- pts
45
46 pts_las <- 0
47 pts_vox <- 0
48
49 difference <- NULL
50
51 tpts <- 0
52
53
54 for(index in 1:rows)
55 {
```

```
56
57 #    cat(index,"out of", rows,"\n")
58
59    x <- as.double(vox$x[index])
60    y <- as.double(vox$y[index])
61
62    TF <- pts$x > x - hw & pts$x <= x + hw
63    pts1 <- subset(pts, TF)
64    TF <- pts1$y > y - hw & pts1$y <= y + hw
65    pts1 <- subset(pts1, TF)
66
67    pointsInCol <- nrow(pts1)
68    colPoints <- 0
69
70    tpts <- tpts + pointsInCol
71
72
73      # Test the point count for each voxel
74      # against the number of points from
75      # LAS file in that range
76
77      j <- 1
78
```

```
79      z <- voxels[j]
80
81      for(i in 1:voxct)
82      {
83  #       print(z)
84
85        TF <- pts1$z > z - hh & pts1$z <= z + hh
86        z <- z + h
87
88        pts1.2 <- subset(pts1, TF)
89
90  #       if(nrow(pts1.2) > 0) remaining <- remaining[
    ↪ remaining$index != pts1.2$index, ]
91
92        points <- nrow(pts1.2)
93
94        if(points != vox[index,i + 4])
95        {
96          cat("Mismatch\n", points, " ", vox[index,i+5],"
              ↪ \n")
97          mistakes <- mistakes + 1
98          mistakesVect <- c(mistakesVect, points - vox[
              ↪ index,i+4])
```

```
99  #           listPts <- c(listPts, pts1.2$index)
100         }
101         else
102         {
103 #           if(nrow(pts1.2) > 0) cat("MATCH\n")
104 #           listPts <- c(listPts, pts1.2$index)
105         }
106 
107         pts_las <- pts_las + points
108         pts_vox <- pts_vox + vox[index,i+4]
109 
110         colPoints <- points + colPoints
111 
112       }
113 
114 #       if(pointsInCol > 0) cat(pointsInCol, colPoints,"\
    ↪ n")
115       difference <- c(difference, pointsInCol - colPoints
        ↪ )
116 }
117 
118 print(points)
119 print(mistakes)
```

10.2 Flat File Format Descriptions

```
*****************************************

VoxGen Flat File Formats

*****************************************

Lidar Metric File Format

VoxGen outputs a lidar metric flat file in the
        following format by row per voxel column
        (if there is no file containing the metric,
        a default value of -99 is used):

x coordinate vox column center, y coordinate vox
        column center, acquisition id, segment id,
        pulse_density, pulse_scan_angle,
        returns_per_pulse, mean, qmean, stdev,
        skew, kurt, p10, p20, p30, p40, p50, p60,
        p70, p80, p90, p100, d0, d1, d2, d3, d4,
```

```
        d5, d6, d7, d8, d9, refl_max, shrub_mean,
        shrub_stdev, shrub_refl_max, tree_mean,
        tree_qmean, tree_stdev, tree_rugosity,
        tree_skew, tree_kurt, tree_fcover,
        tree_fract_al, tree_p10, tree_p20, tree_p30,
        tree_p40, tree_p50,tree_p60, tree_p70, tree_p80
           ↪ ,
        tree_p90, tree_p100, tree_d0, tree_d1, tree_d2,
        tree_d3, tree_d4, tree_d5, tree_d6, tree_d7,
        tree_d8, tree_d9, tree_iqr, tree_vdr, tree_mad,
        tree_aad, tree_crr, tree_refl_max,
        ground_elev_mean, ground_slope, ground_aspect,
        ground_refl_max



**********************************************

Histogram File Format

VoxGen outputs a histogram flat file in the following
        format by row per voxel column:

x coordinate vox column center, y coordinate vox column
```

```
        center, acquisition id, segment id,
        bin1 number of points, ..., binN number of
           ↪ points
```

References

[1] *Boost Background Information*. boost. http://www.boost.org/users/. 2005.

[2] *GDAL API Tutorial*. GDAL. http://www.gdal.org/gdal_tutorial.html. 20 November 2013.

[3] *GDAL - Geospatial Data Abstraction Library* . GDAL. http://www.gdal.org/.

[4] *G-LiHT: Goddard's LiDAR, Heyperspectral & Thermal Imager*. National Aeronautics & Space Administration Goddard Space Filight Center. NASA. http://gliht.gsfc.nasa.gov/. 2013.

[5] *kdtree: A simple C library for working with KD-Trees*. Google Code. https://code.google.com/p/kdtree/. Nov. 2011.

[6] *LAS Specification*. asprs. The American Society for Photogrammetry & Remote Sensing. http://www.asprs.org/a/society/committees/standards/LAS_1_4_r13.pdf. 15 July 2013.

[7] *libLAS API Reference Documentation*. libLAS. http://www.liblas.org/doxygen/index.html.

[8] Loskot, Mateusz & Butler, Howard. *C++ Tutorial*. libLAS. http://www.liblas.org/tutorial/cpp.html. 22 February 2011.

[9] Loskot, Mateusz & Butler, Howard. *libLAS*. libLAS. http://www.liblas.org/index.html. 22 July 2014.

[10] R Core Team. *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/. 2014.

[11] *Simple ls Program*. boost. http://www.boost.org/doc/libs/1_31_0/libs/filesystem/example/. 2002.