Efficient processing of dense point clouds in GRASS GIS

at US-IALE 2016 Annual Meeting

Vaclav Petras (Vashek)

Douglas Newcomb, Helena Mitasova

Center for Geospatial Analytics

NC STATE UNIVERSITY

March 18, 2016

available at

`wenzeslaus.github.io/grass-lidar-talks`

Points

- collected by lidar
- generated by Structure from Motion (SfM) from UAV imagery
- a lot of points

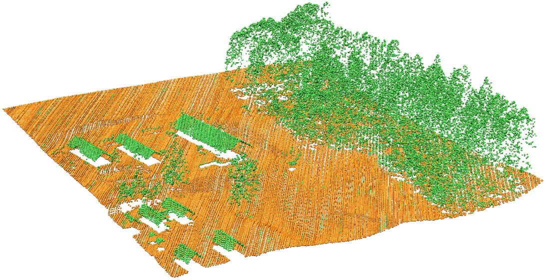
surface interpolated from points from SfM

Acknowledgements

Datasets

Lidar and UAV Structure from Motion (SfM) data for GIS595/MEA792: UAV/lidar Data Analytics course

Nantahala NF, NC: Forest Leaf Structure, Terrain and Hydrophysiology. Obtained from OpenTopography. `http://dx.doi.org/10.5069/G9HT2M76`



Free, libre and open source

Scripts and code I'm writing

- review
- re-usable
  - by other people
  - by future myself

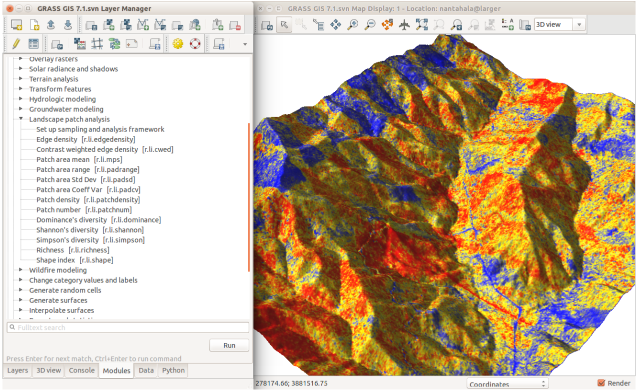
Software I'm using

- driven by needs of users
- longevity
  - learn now, use forever
  - GRASS GIS: over 30 years of development

Open Science Logo, Greg Emmerich, CC BY-SA 2.0

GRASS GIS

- universal scientific and processing platform
  - GUI, CLI, Python API
  - from small laptops to supercomputers
- lidar processing included
- data size and type challenges



Acknowledgements

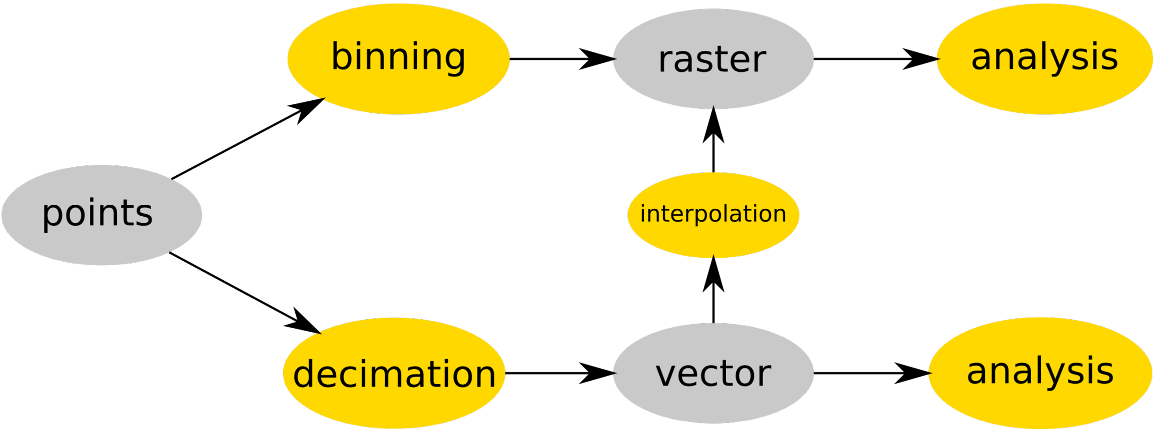
Software

Presented functionality is work done by Vaclav Petras, Markus Metz, and the GRASS development team.

Thanks to users for feedback and testing, especially to Douglas Newcomb, Helena Mitasova, Markus Neteler, Laura Belica, and William Hargrove.

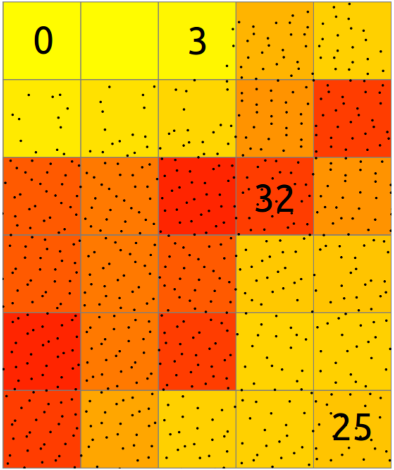
GRASS GIS

Workflow overview



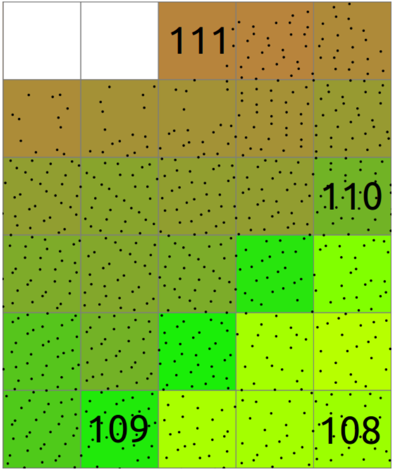
Binning as 2D histogram

- counts number of points in cell



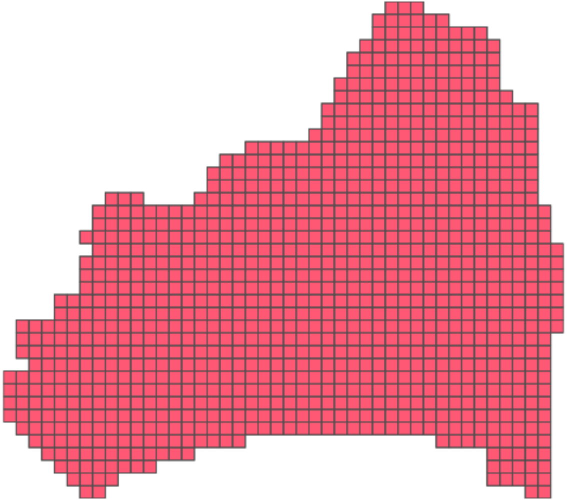
Binning points to raster

- *r.in.lidar* (import and analysis)
- statistics of point counts, height and intensity
  - n, min, max, sum
  - mean, range, skewness, . . .



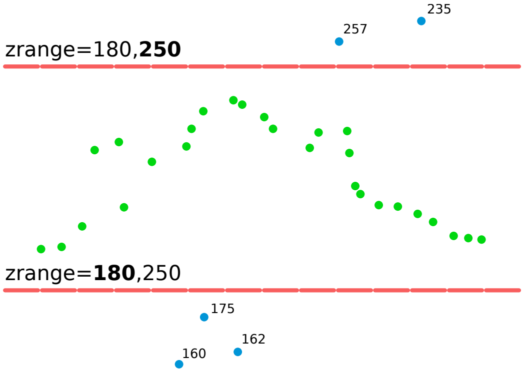
Practical functions

- analytical and practical functions in *r.in.lidar*
- read multiple tiles as one
  - no merging
  - 0.5 billion points in 90 files in minutes



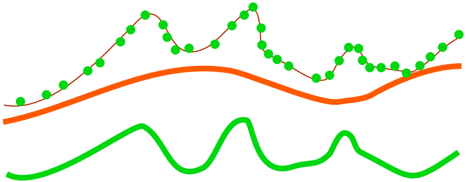
Filtering points

- filter points by
  - range of Z
  - return
  - class
  - ...
- at the time of binning with *r.in.lidar*
  - minimal additional cost

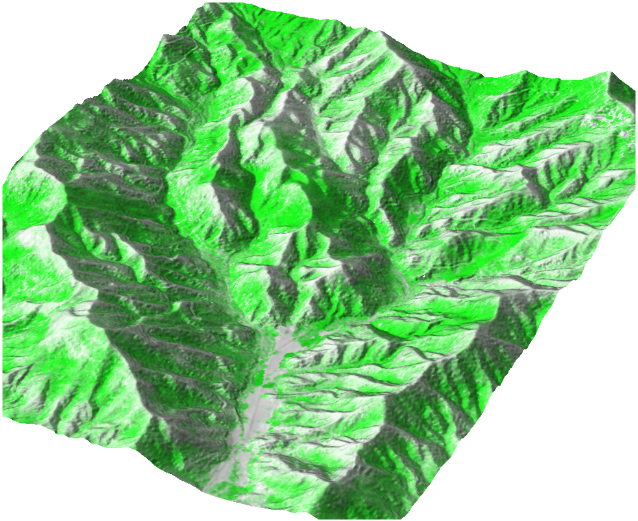


Height above a surface

- new base raster feature in *r.in.lidar*
- given surface + points cloud $\longrightarrow$ height of features

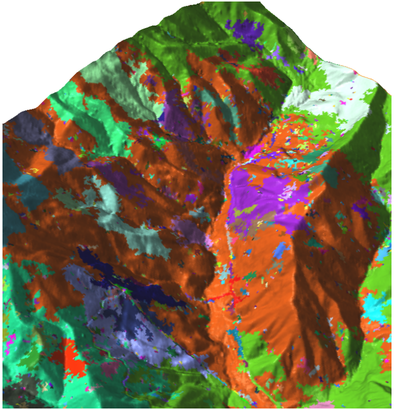


- not limited by memory



Rasterize early

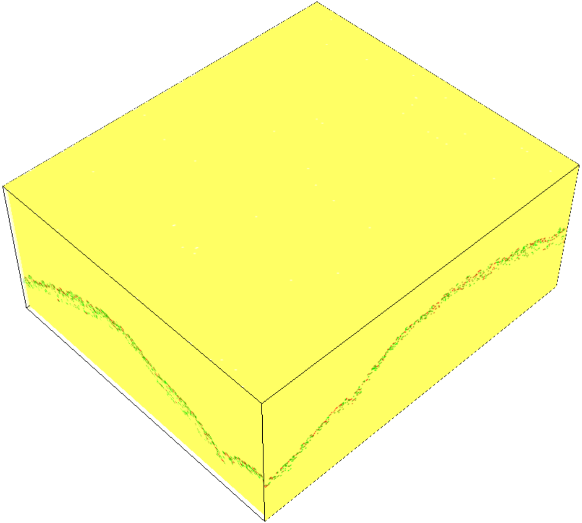
- less cells then points
  - 578 mil points (ground 30 mil)
  - 15 mil cells in 8km $\times$ 7km at resolution 2m
    - faster to loop through
    - less disk space
- raster
  - natural spatial index
  - that's what the algorithms use



i.segment on different point counts

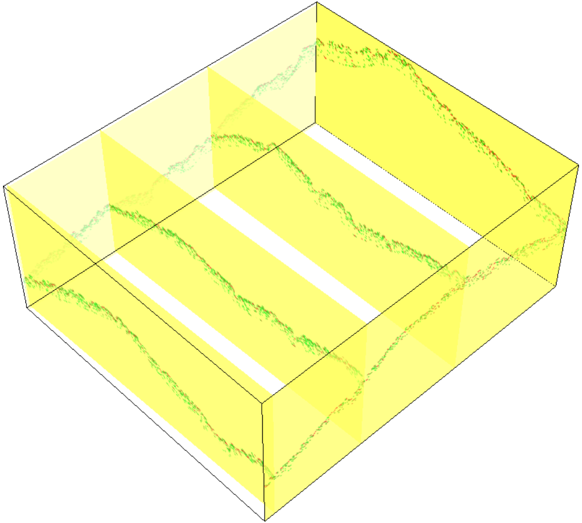
3D raster

- stacked 2D rasters
- challenging to visualize
- same principles as in 2D
  - e.g. 3D raster map algebra



3D raster

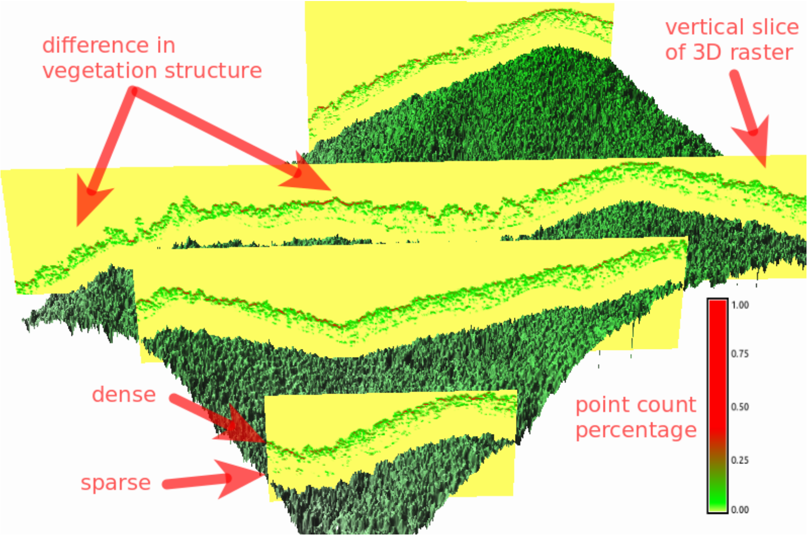
- stacked 2D rasters
- challenging to visualize
- same principles as in 2D
    - e.g. 3D raster map algebra



Binning points to 3D raster

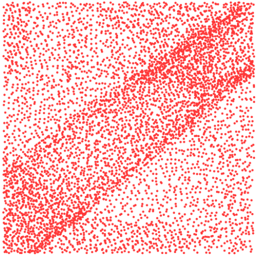
- *r3.in.lidar*
- proportional count
  - count per 3D cell relative to the count per vertical column
- intensity can be used instead of count

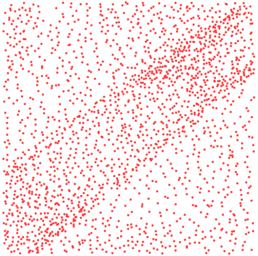
height reduction by base raster under development (analysis and space efficient)



Decimation

- *v.in.lidar*
  - filtering same as in *r.in.lidar*
- often more points than we need (research shows, Singh et al. 2015, Petras et al. 2016)
- interpolation, clustering, ... are costly
- decimation ≈ sampling
  - fast count-based as effective as more advanced decimation





Large point clouds

Rasters (binning of points)

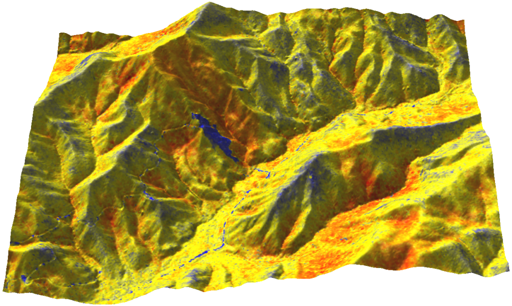
- trade-off: memory (RAM) or slow
- 64bit version
  - your operating system may limit max memory

Brunswick county: binning, ≈1050 files, > 9 billion points
Hyde county: binning, ≈950 files, > 4 billion points, base elevation 5ft raster, 60ft height raster

≈0.5-3 hours, 1-13GB of memory (in-memory mode)

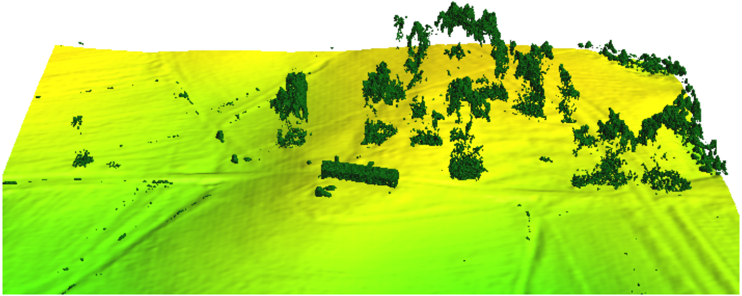
Vectors (points as points)

- point cloud specific optimizations
  - no IDs stored
  - no attribute table
  - no topology created



Ground detection

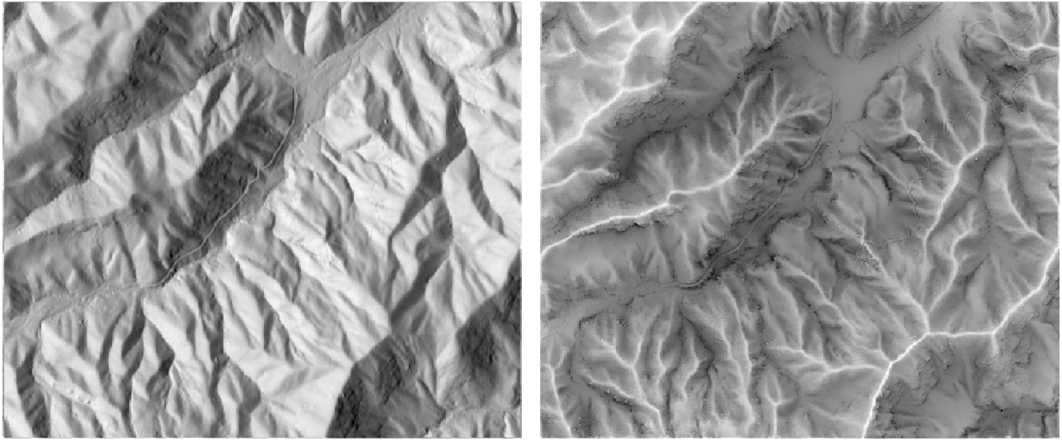
- *v.lidar.edgedetection*, *v.lidar.growing*, *v.lidar.correction*
  - uses returns
- *v.lidar.mcc*
  - multiscale curvature based classification algorithm[1]



[1] Evans, J. S. & Hudak, A. T. 2007: A Multiscale Curvature Algorithm for Classifying Discrete Return LiDAR in Forested Environments.

Sky-view factor

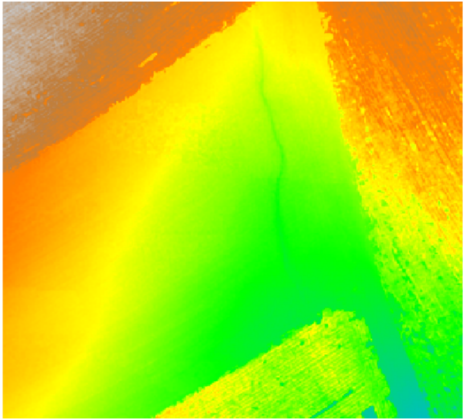
- *r.skyview* (percentage of visible sky)

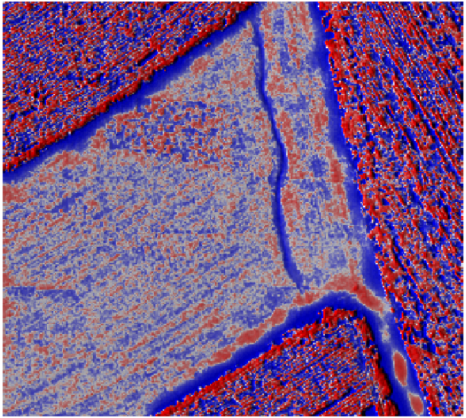


comparison of shaded relief and sky-view factor

Local relief model (LRM)

- *r.local.relief* (micro-topography, features other than trend)



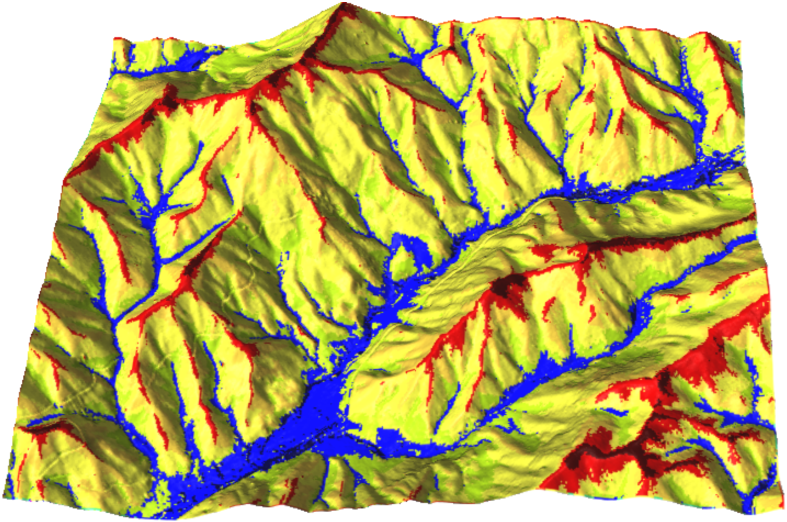


30-60cm wide, 30cm deep, 60m long gully (resolution 30cm)

Landforms

- *r.geomorphon*
- geomorphons - a new approach to classification of landform[1]

[1] Jasiewicz, J., Stepinski, T., 2013, Geomorphons - a pattern recognition approach to classification and mapping of landforms, Geomorphology



Integration with PDAL

PDAL

- Point Data Abstraction Library
- formats besides LAS/LAZ
- algorithms, filters, decimations

Experimental integration

- *v.in.pdal*
- reprojection during import
- ground filter
- compute height as a difference from ground

Summary

- rasterize early
- make use of existing methods for raster and vector processing
- 3D rasters, PDAL integration
- the plan for next 30 years driven by users – grass-user mailing list

Get GRASS GIS 7.1 development version at
`grass.osgeo.org/download`

Slides available at
`wenzeslaus.github.io/grass-lidar-talks`

Paper in preparation: *Processing UAV and lidar point clouds in GRASS GIS*