GRASS GIS loves lidar

FOSS4G NA 2016

Vaclav Petras (Vashek)

Anna Petrasova, Helena Mitasova

Center for Geospatial Analytics

NC STATE UNIVERSITY

May 5, 2016

GRASS GIS

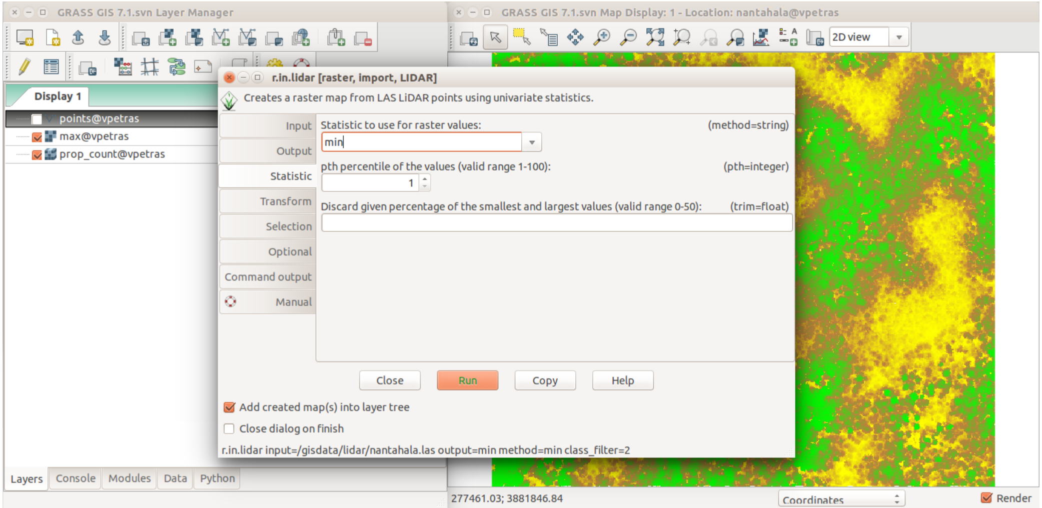
- all in one
  - hydrology modeling, image segmentation, point clustering, ...
- from small laptops to supercomputers
  - Raspberry Pi, Windows, Mac, GNU/Linux, GNU/Hurd, FreeBSD, IBM AIX
- learn now, use forever
  - over 30 years of development and interface refinement
- probably used more than you think
  - similarly to C/C++ is often not mentioned but is somewhere in there

latest release 7.0.4 Sunday

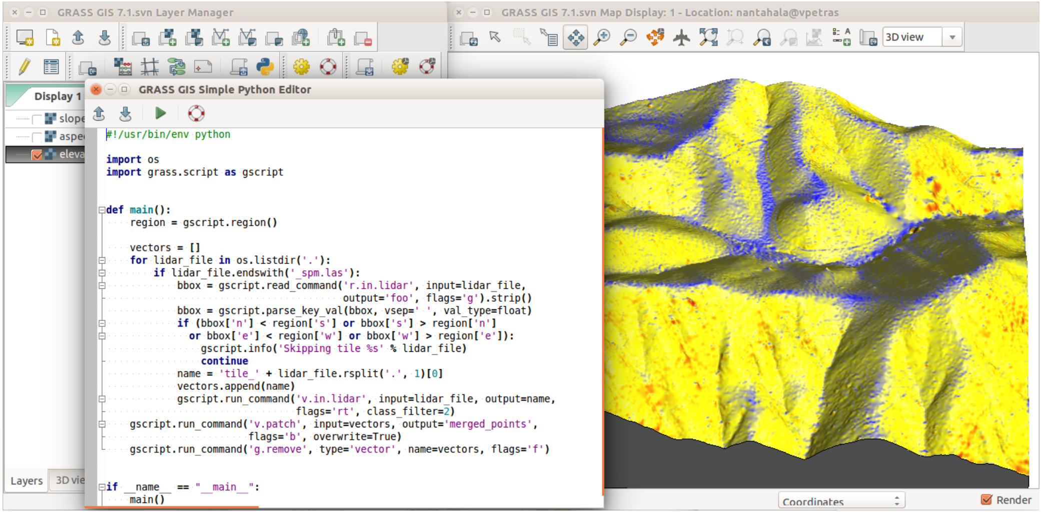
CLI

```
Welcome to GRASS GIS 7.1.svn (r68305M)
GRASS GIS homepage:                         http://grass.osgeo.org
This version running through:               Bash Shell (/bin/bash)
Help is available with the command:         g.manual -i
See the licence terms with:                 g.version -c
Start the GUI with:                         g.gui wxpython
When ready to quit enter:                   exit

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

GRASS 7.1.svn (nantahala):~/dev/grass/gcc_trunk > g.region vector=points
GRASS 7.1.svn (nantahala):~/dev/grass/gcc_trunk > g.region res=5
GRASS 7.1.svn (nantahala):~/dev/grass/gcc_trunk > r.in.lidar input=/gisdata/lidar/points.las output=mean
```

GUI


GRASS GIS 7.1.svn Layer Manager
GRASS GIS 7.1.svn Map Display: 1 - Location: nantahala@vpetras
2D view
Display 1
points@vpetras
max@vpetras
prop_count@vpetras
r.in.lidar [raster, import, LIDAR]
Creates a raster map from LAS LiDAR points using univariate statistics.
Input
Output
Statistic
Transform
Selection
Optional
Command output
Manual
Statistic to use for raster values:
(method=string)
min
pth percentile of the values (valid range 1-100):
(pth=integer)
1
Discard given percentage of the smallest and largest values (valid range 0-50):
(trim=float)
Close
Run
Copy
Help
Add created map(s) into layer tree
Close dialog on finish
r.in.lidar input=/gisdata/lidar/nantahala.las output=min method=min class_filter=2
Layers
Console
Modules
Data
Python
277461.03; 3881846.84
Coordinates
Render

Python


GRASS GIS 7.1.svn Layer Manager
GRASS GIS 7.1.svn Map Display: 1 - Location: nantahala@vpetras
3D view
Display 1
slope
aspe
eleva
Layers
3D vie
GRASS GIS Simple Python Editor
#!/usr/bin/env python
import os
import grass.script as gscript
def main():
region = gscript.region()
vectors = []
for lidar_file in os.listdir('.'):
if lidar_file.endswith('_spm.las'):
bbox = gscript.read_command('r.in.lidar', input=lidar_file,
output='foo', flags='g').strip()
bbox = gscript.parse_key_val(bbox, vsep=' ', val_type=float)
if (bbox['n'] < region['s'] or bbox['s'] > region['n']
or bbox['e'] < region['w'] or bbox['w'] > region['e']):
gscript.info('Skipping tile %s' % lidar_file)
continue
name = 'tile_' + lidar_file.rsplit('.', 1)[0]
vectors.append(name)
gscript.run_command('v.in.lidar', input=lidar_file, output=name,
flags='rt', class_filter=2)
gscript.run_command('v.patch', input=vectors, output='merged_points',
flags='b', overwrite=True)
gscript.run_command('g.remove', type='vector', name=vectors, flags='f')
if __name__ == "__main__":
main()
Coordinates
Render

Python versus CLI

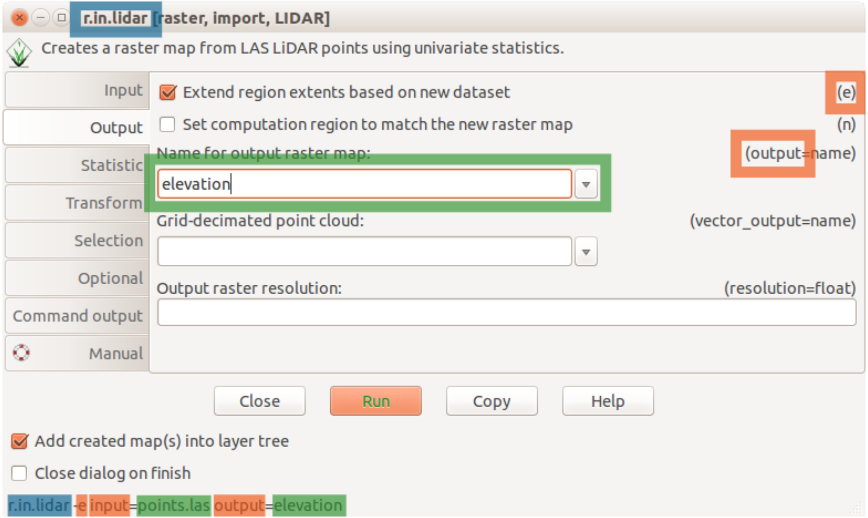
Documentation, Command Line (Shell, Bash, cmd.exe):

```
r.in.lidar input=points.las \
           output=elevation -e
```

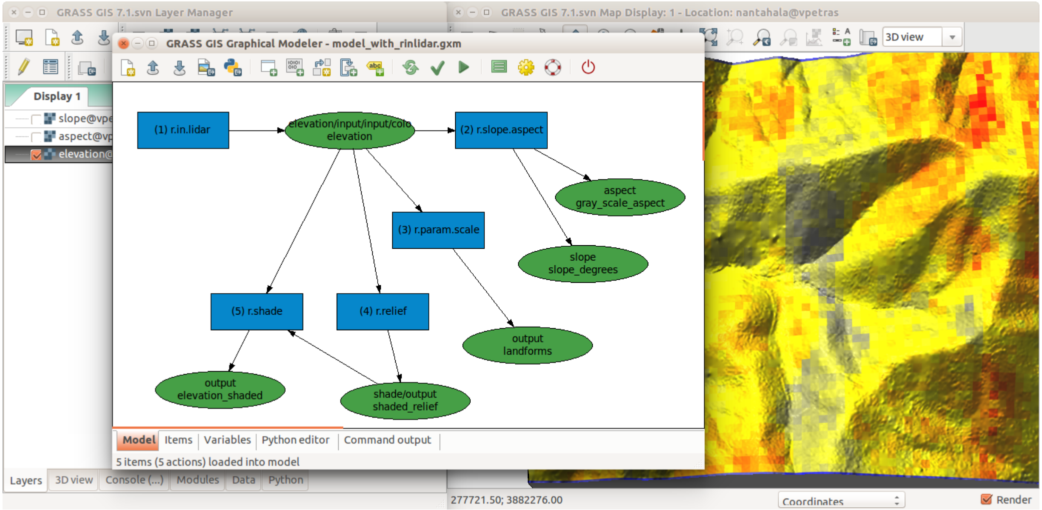
Python:

```
from grass.script import run_command
run_command('r.in.lidar',
            input="points.las",
            output="elevation",
            flags='e')
```

Module GUI


r.in.lidar [raster, import, LIDAR]
Creates a raster map from LAS LiDAR points using univariate statistics.
Input
Output
Statistic
Transform
Selection
Optional
Command output
Manual
Extend region extents based on new dataset
(e)
Set computation region to match the new raster map
(n)
Name for output raster map:
(output=name)
elevation
Grid-decimated point cloud:
(vector_output=name)
Output raster resolution:
(resolution=float)
Close
Run
Copy
Help
Add created map(s) into layer tree
Close dialog on finish
r.in.lidar -e input=points.las output=elevation

Graphical Modeler

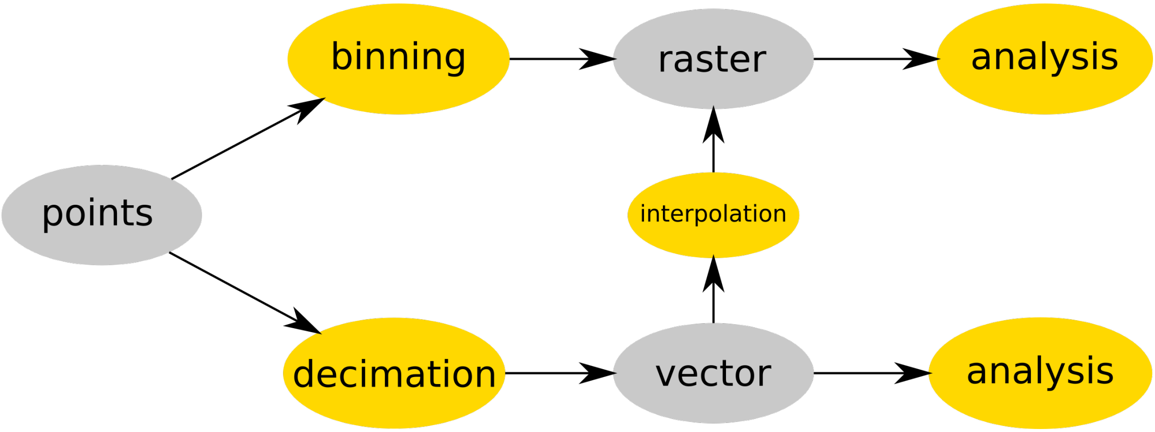


Points

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

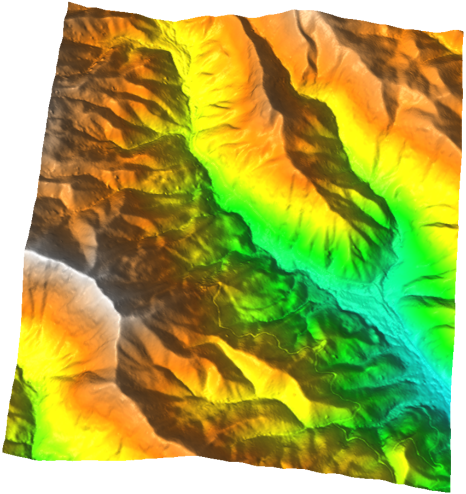
surface interpolated from points and visualized in GRASS GIS

Workflow overview



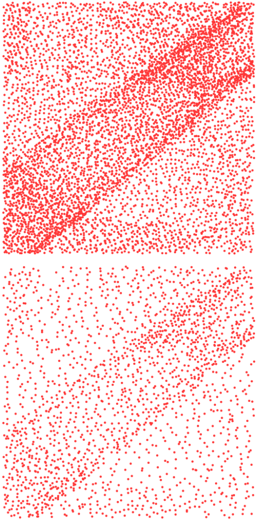
Surface interpolation

- *v.surf.idw*
  - Inverse Distance squared Weighting
- *v.surf.bspline*
  - Bicubic or bilinear Spline interpolation with Tykhonov regularization
- *v.surf.rst*
  - Regularized Spline with Tension
  - *v.surf.rst.mp* (experimental)
    - 2 millions of points in 11 minutes



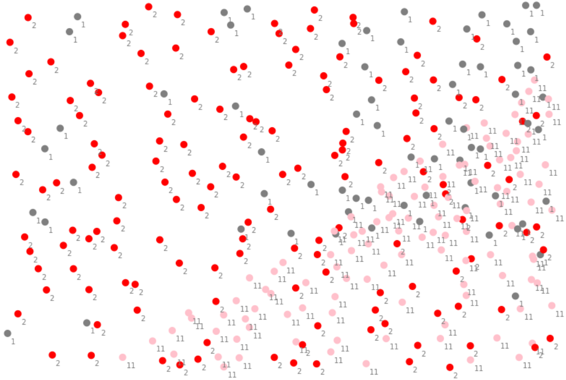
Import and decimation

- *v.in.lidar*
  - libLAS
  - LAS/LAZ to GRASS GIS native vector
  - data stored in GRASS GIS database
- decimation ≈ thinning ≈ sampling
  - count-based decimation (skips points)
  - grid-based experimental, others needed?
  - fast count-based as good as more advanced decimations



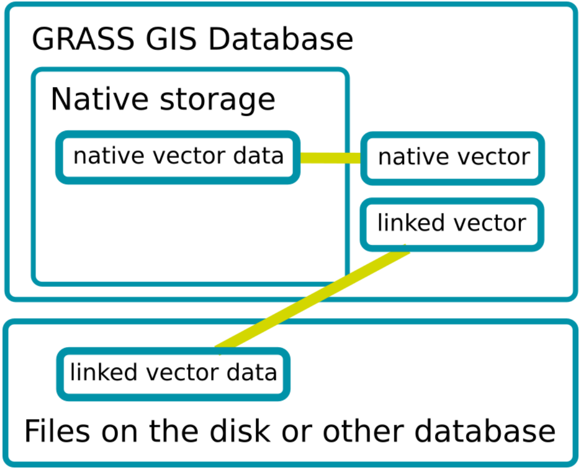
GRASS vector model and format

- topology and index
  - can be disabled (-*b* flag)
- attributes in a database
  - SQLite, PostgreSQL, ...
  - can be disabled (-*t* flag)
- each feature can have any number of categories/classes
  - without attribute table



Linked external data

- *r.external*
  - raster data (GDAL)
  - *r.external.out* for newly created data
- *v.external*
  - vector data
    - GDAL/OGR
    - PostGIS including topology
  - *v.external.out* for newly created data
  - alternative: @OGR
    `v.info map=.../directory@OGR layer=file`



- missing: libLAS/PDAL backend
  - intermediate C API needed in PDAL or GRASS GIS

Current state of integration with PDAL

PDAL

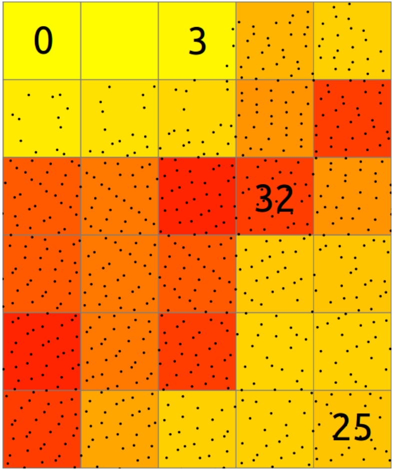
- Point Data Abstraction Library
- format conversions
- processing, filtering

Experimental integration

- *v.in.pdal*
  - next: *r.in.pdal*, *r3.in.pdal*
- runs PDAL filters during import
  - filters are followed by GRASS processing

Binning points to raster

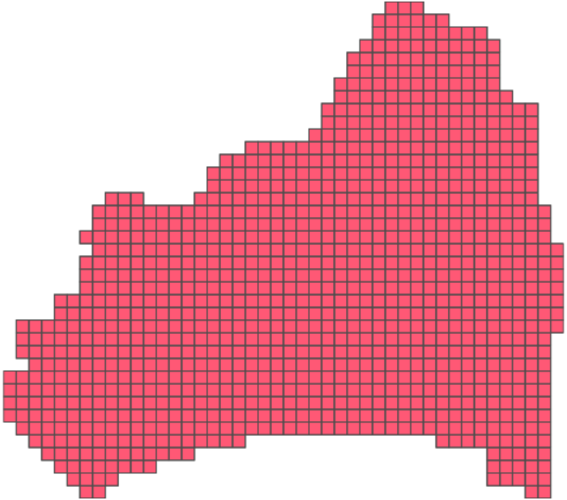
- *r.in.lidar*
- import and analysis
- statistics of point counts, height and intensity
  - n, min, max, sum
  - mean, range, skewness, ...



Read multiple tiles as one

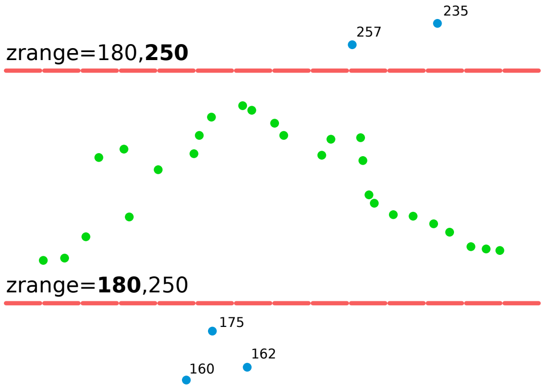
- *r.in.lidar*, option *file*
  - read multiple tiles as one
  - no merging

0.5 billion points in 90 files in minutes



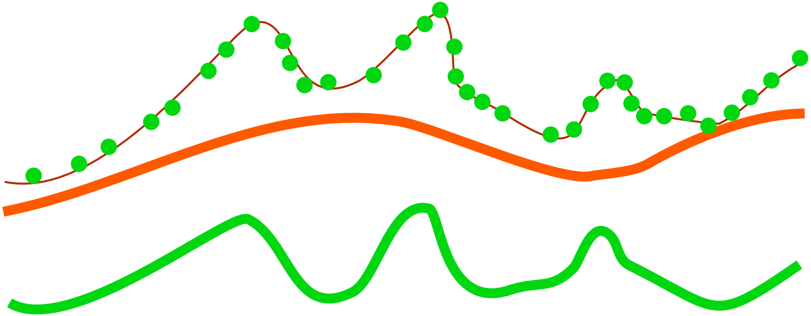
Filtering points

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



Height above a surface

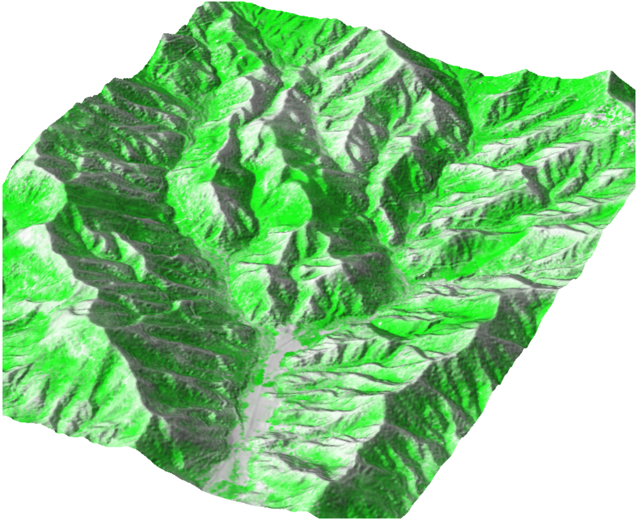
- *r.in.lidar*, option *base_raster*
- given surface + points cloud $\longrightarrow$ height of features



- low additional memory requirements

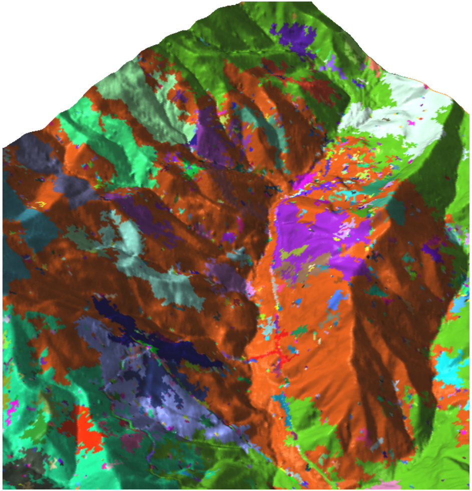
Height above a surface

- different resolutions
  - 1m ground surface
  - 30m height above ground
- different statistics
- different combinations
  - surface can be e.g. top of the canopy
  - combine with *zrange*
  - combine with intensity



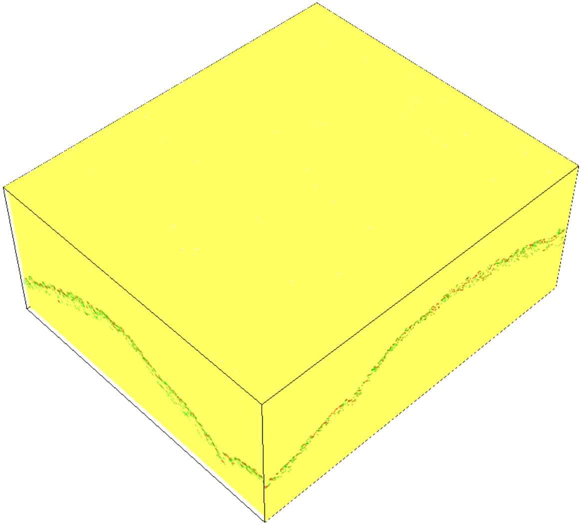
Rastersize early

- many algorithms are raster-based
  - a lot of data with continuous nature
  - natural spatial index
- example:
  1. count of ground points
  2. count of non-ground points
  3. used as image bands
  4. segmentation using *i.segment*



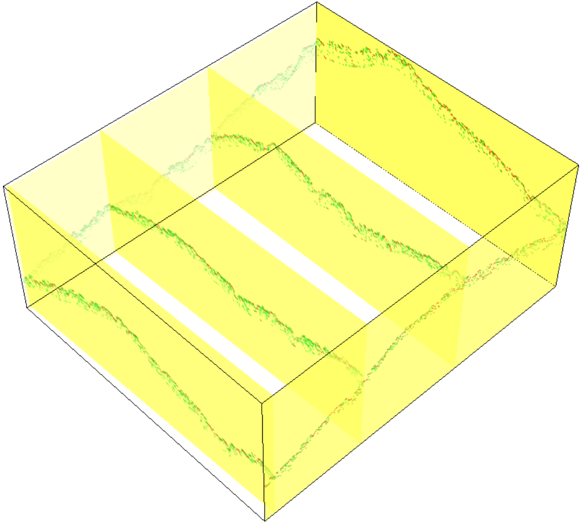
3D raster

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



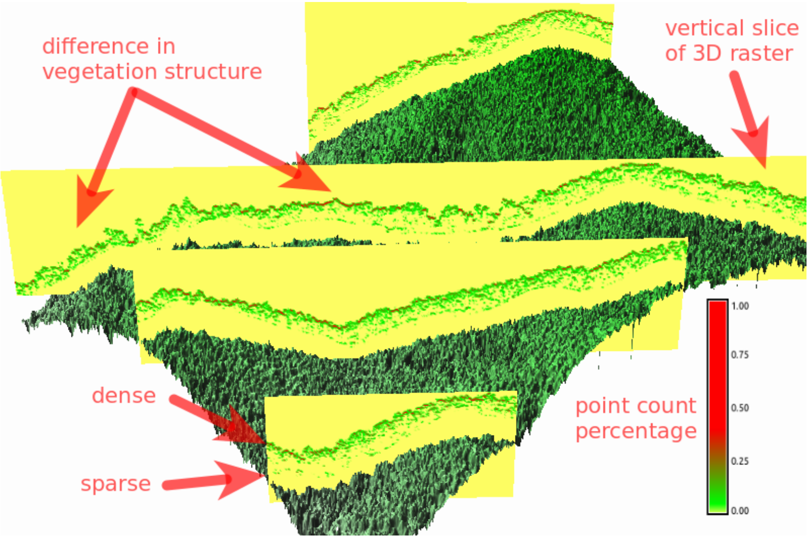
3D raster

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

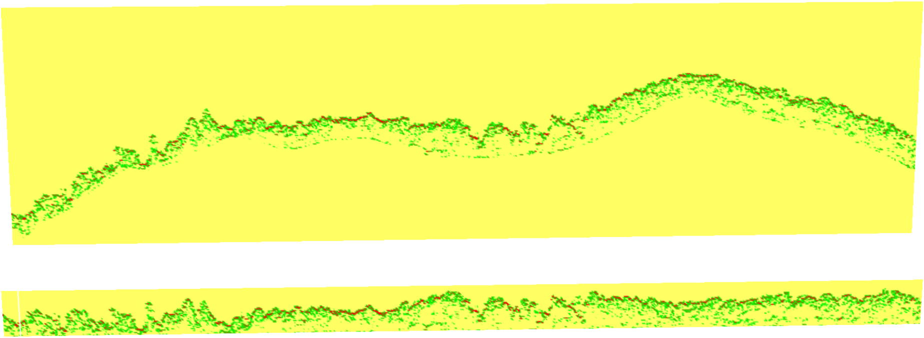


Binning points to 3D raster

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



Point heights reduced to surface



- *r3.in.lidar*, option *base_raster*
- height reduced by raster values

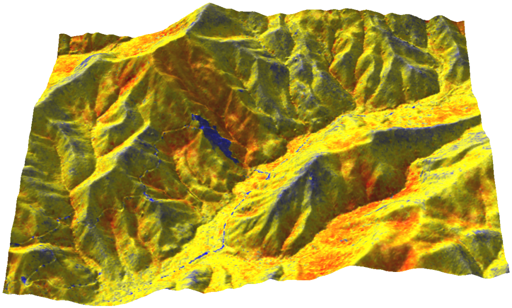
Trade-offs

Raster processing

- hight memory (RAM) usage – fast
- low memory usage (high I/O) – slow

Vector processing

- slower than raster
  - e.g., interpolation much slowed than binning
- hard to make general statements



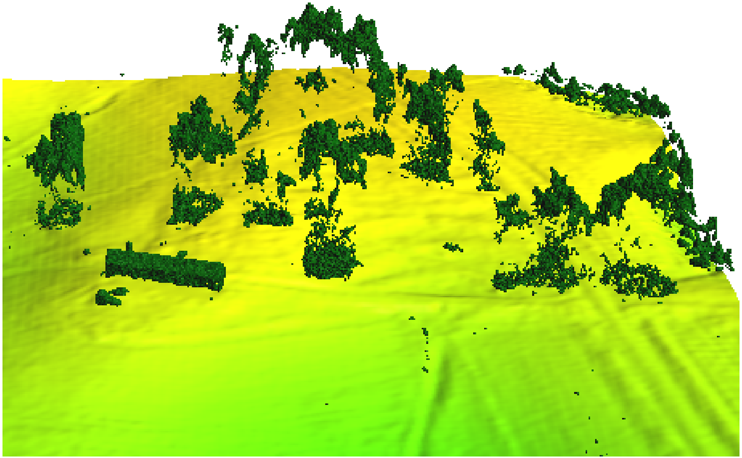
visualization: range from binning on interpolated surface

example: binning with base elevation subtraction: ≈1000 files, > 9 billion points

≈3 hours, ≈10GB of memory (in-memory mode)

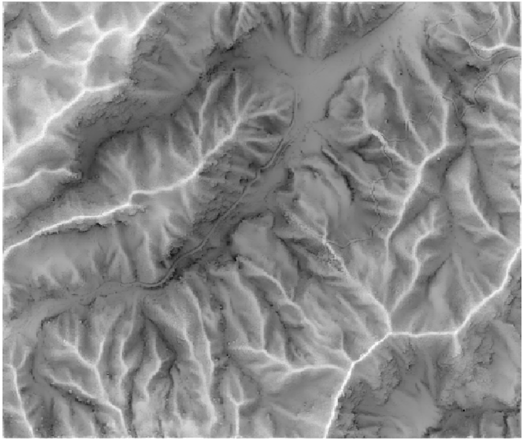
Ground detection

- *v.lidar.edgedetection, v.lidar.growing, v.lidar.correction*
  - by Brovelli, Cannata, Antolin & Moreno
- *v.lidar.mcc*
  - multiscale curvature based classification algorithm
  - by Blumentrath, according to Evans & Hudak
- PDAL filters.ground
  - now in v.in.pdal
  - progressive morphological filter by Zhang
  - provided by PCL



Sky-view factor

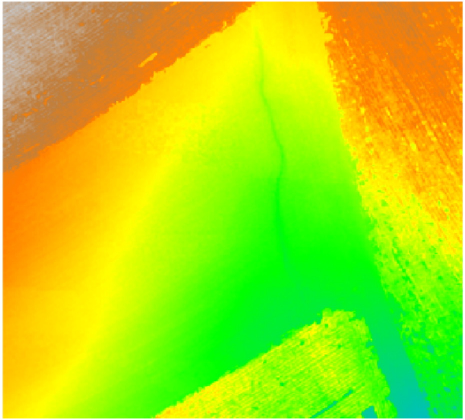
- *r.skyview* (percentage of visible sky)

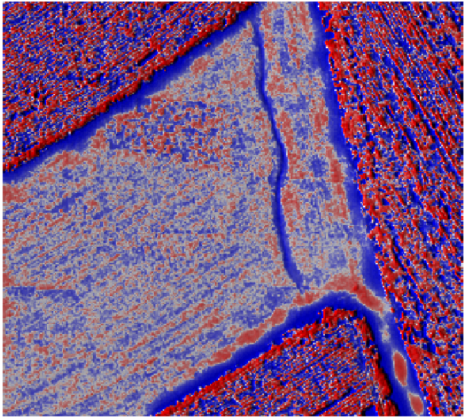


comparison of shaded relief and sky-view factor

Local relief model (LRM)

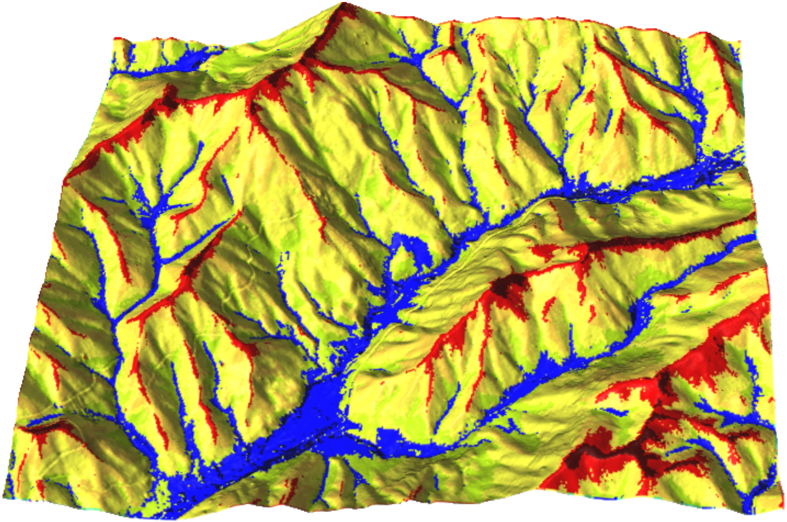
- *r.local.relief* (micro-topography, features other than trend)





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

- *r.geomorphon*
  - new landform classification approach
  - by Jasiewicz & Stepinski



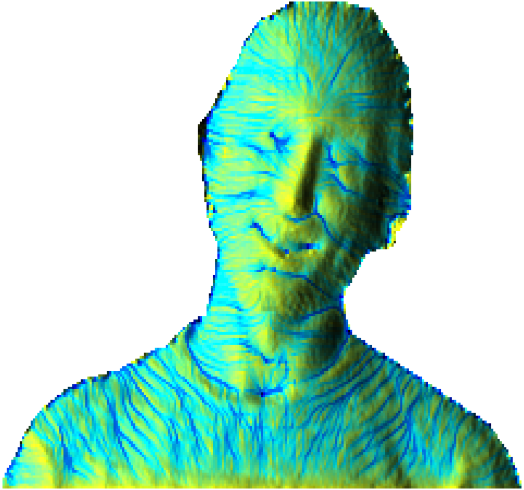
libfreenect2 + PCL + GRASS GIS = *r.in.kinect*

r.in.kinect

- scans using Kinect
- OpenKinect libfreenect2
- Point Cloud Library (PCL)
- GRASS GIS libraries
  - C API
  - raster processing
  - regularized spline with tension interpolation

used in Tangible Landscape

TANGIBLE LANDSCAPE



Summary

- rasterize early
- make use of existing methods for raster and vector processing
- 3D rasters, PDAL integration

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

Slides and paper available at
`wenzeslaus.github.io/grass-lidar-talks`
GRASS user mailing list
`lists.osgeo.org/listinfo/grass-user`

Acknowledgements

Software

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

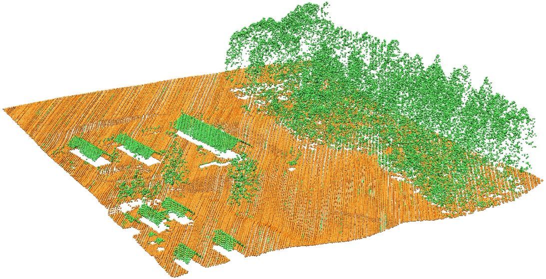
Thanks to users for feedback and testing, especially to Doug Newcomb, Helena Mitasova, Markus Neteler, Laura Belica, and William Hargrove.

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



Acknowledgements

Presentation software

Slides were created in LaTeX using the BEAMER *class*.