Processing UAV and lidar point clouds in GRASS GIS

XXIII ISPRS Congress, Prague 2016

Vaclav Petras (Vashek)

Anna Petrasova, Justyna Jeziorska, Helena Mitasova

Center for Geospatial Analytics

NC STATE UNIVERSITY

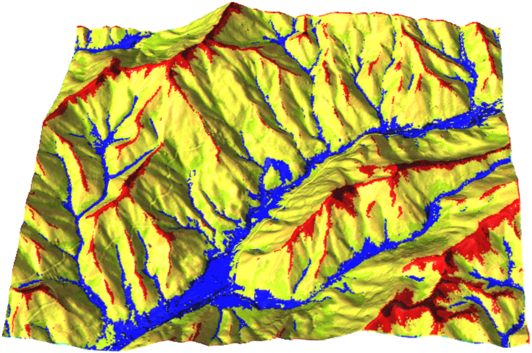
July, 2016

Providing algorithms to the community

- new landform recognition approach – geomorphons
- by Jasiewicz and Stepinski from AMU, Poland and University of Cincinnati, USA
- not just a paper

  Geomorphology, 2013
- not just public code
- *r.geomorphon*

  module in GRASS GIS addons repository



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

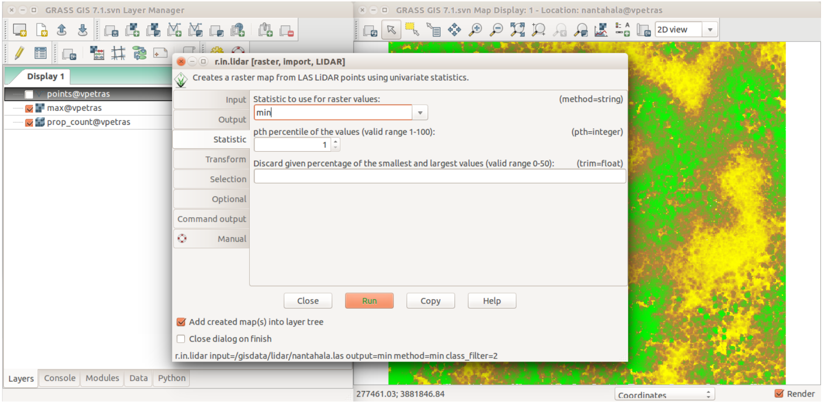
GRASS GIS

- all in one
  - hydrology modeling, image segmentation, point clustering, ...
- from small laptops to supercomputers
  - Raspberry Pi, Windows, Mac, GNU/Linux, FreeBSD, IBM AIX
- learn now, use forever
  - over 30 years of development and interface refinement
- used by
  - US Oak Ridge National Laboratory, Edmund Mach Foundation, JRC, ...

latest release 7.0.4

Sunday, May 1, 2016

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 and command line interfaces

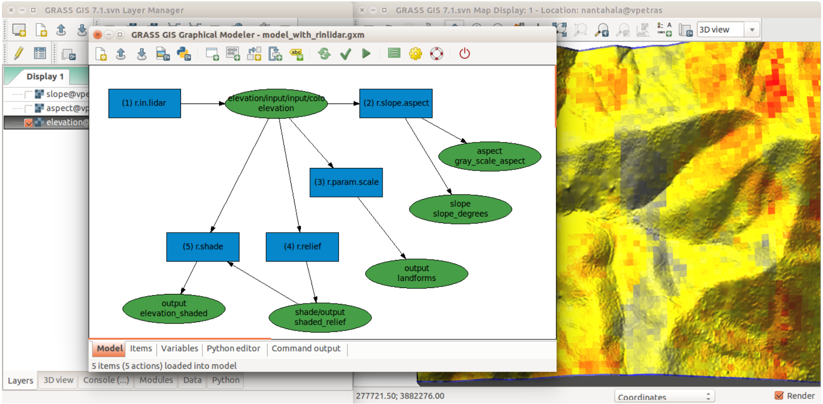
Documentation, Command Line (Shell, Bash, cmd.exe):

```
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')
```

Graphical Modeler

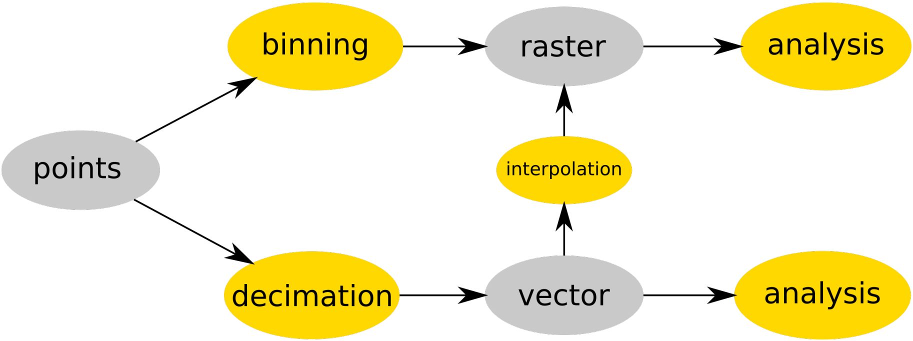


Points

- collected by lidar
- generated by Structure from Motion (SfM) from UAV imagery

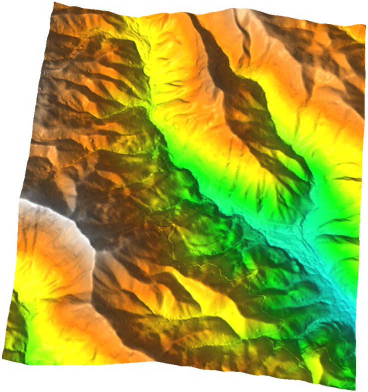
surface interpolated from points and visualized in GRASS GIS

Workflow overview



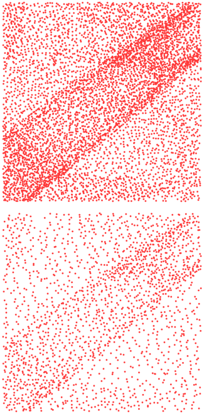
Surface interpolation

- *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



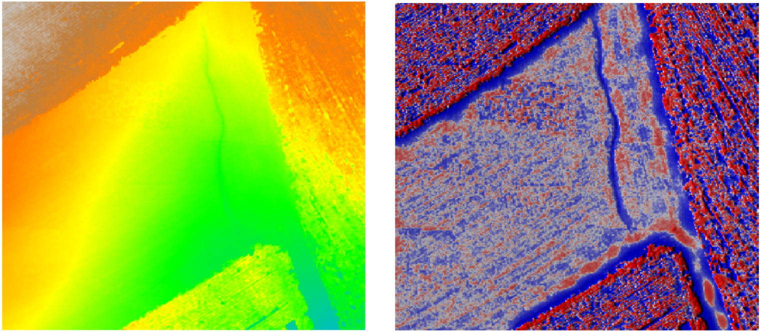
Import and decimation

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



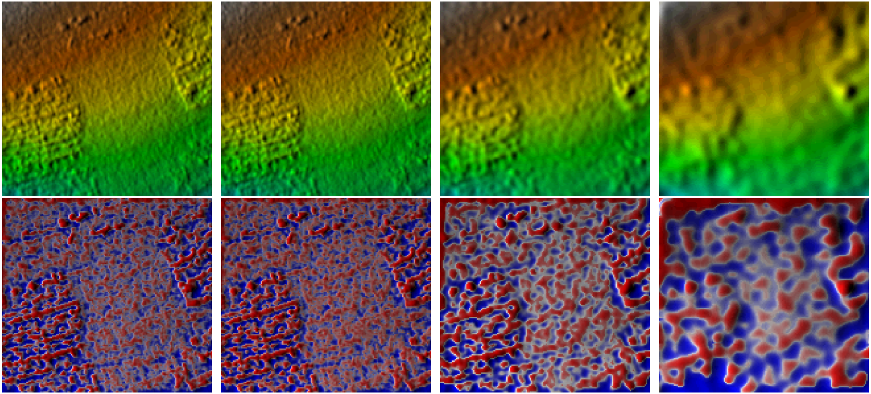
Local relief model (LRM)

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



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

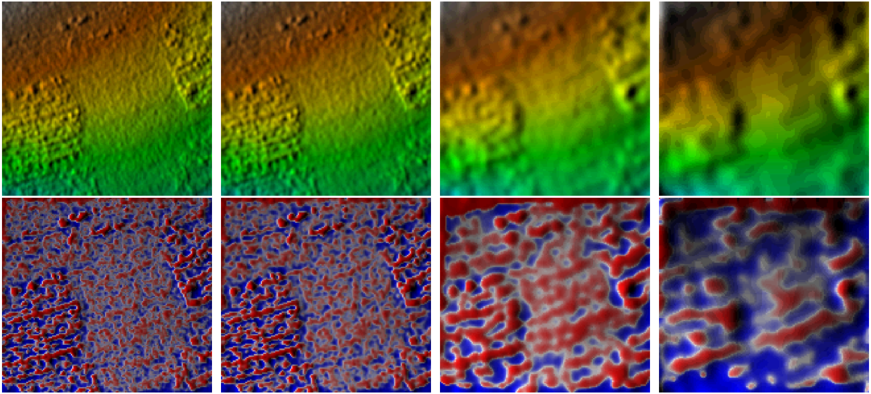
Count-based decimation influence on interpolated elevation



keep all	skip every 5	keep every 20	keep every 100
0 %	20 %	90 %	99 %

test data (not used in the study)

Influence of grid-based decimation resolution



grid size 0.1 m
0 %

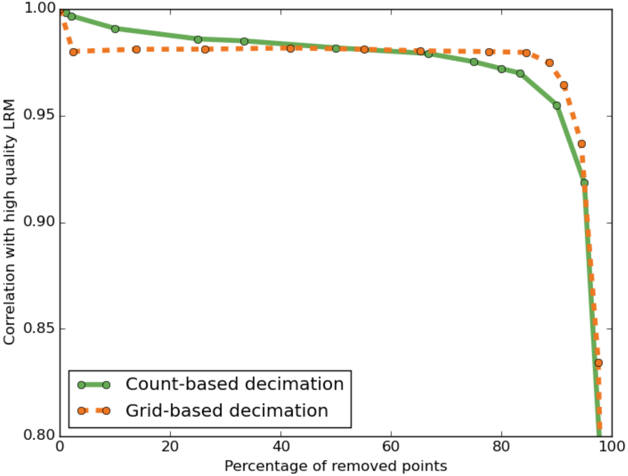
grid size 0.3 m
81 %

grid size 0.9 m
98 %

grid size 1.5 m
99 %

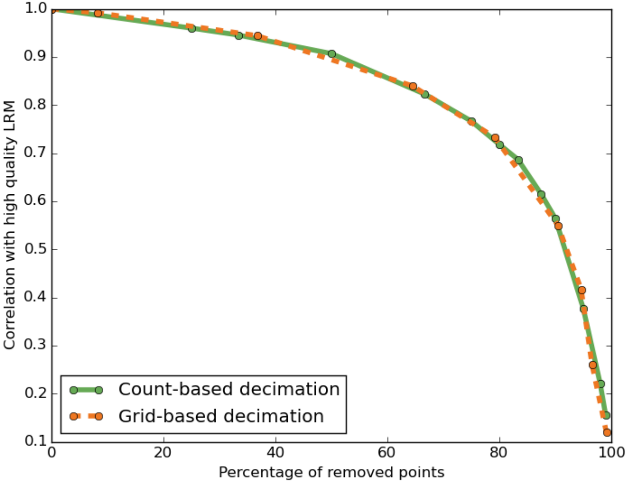
test data (not used in the study)

Decimating UAV/SfM point cloud



interpolations and local relief model computations at resolution 0.5 m 72 point per 1 m^2

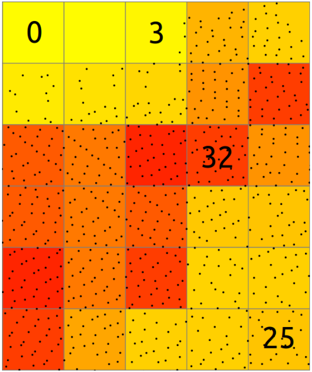
Decimating lidar point cloud



interpolations and local relief model computations at resolution 0.5 m 1 point per 1 m^2

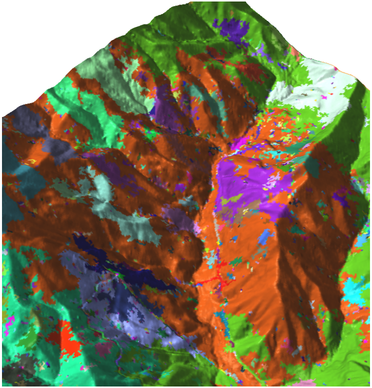
Binning points to raster

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



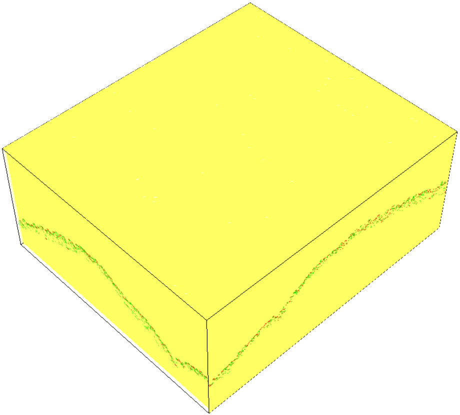
Raster processing

- many algorithms are raster-based
  - 163 raster modules
  - 45 imagery modules
  - 20 spatio-temporal raster modules
- example:
  1. count of ground points
  2. count of non-ground points
  3. used as image bands
  4. segmentation using *i.segment*



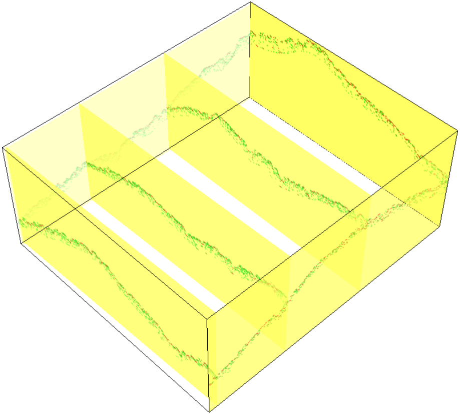
3D raster

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



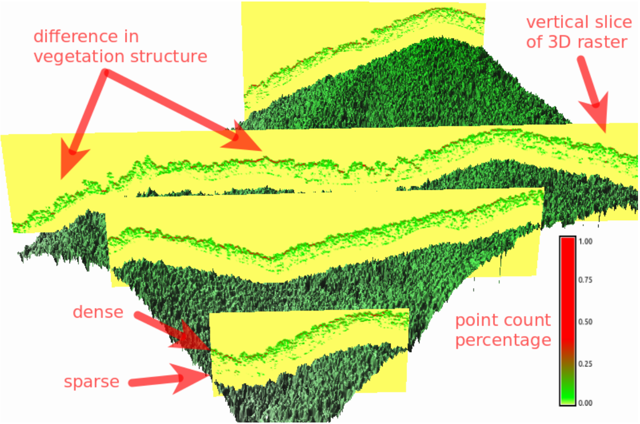
3D raster

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

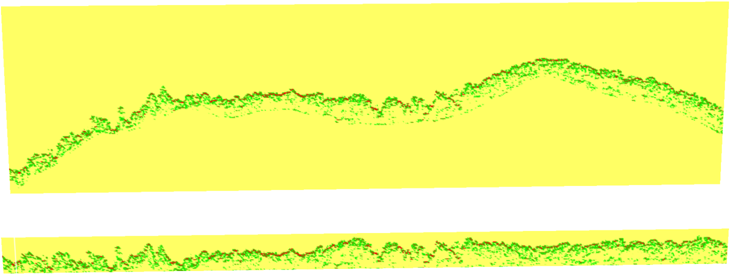


Binning points to 3D raster

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



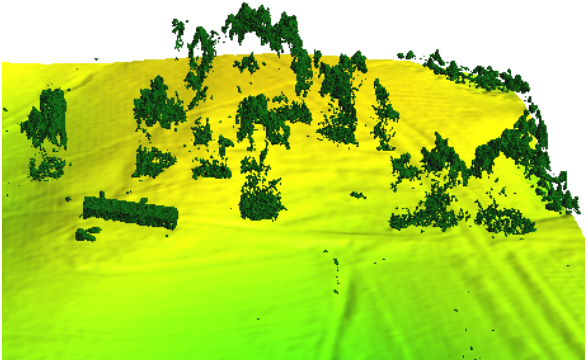
Point heights reduced to surface



- *r3.in.lidar*, option *base_raster*
- height reduced by 2D raster values
- similarly also for 2D binning: height above a surface, top of the canopy, ...

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
  - currently in v.in.pdal
  - progressive morphological filter by Zhang
  - provided by PCL



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

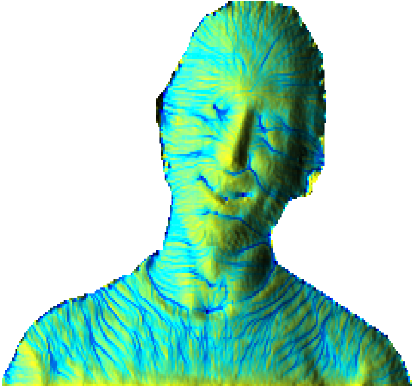
Using other open source projects

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
- GRASS modules can work with large data
  - sometimes a special flag is needed
  - if not, report a bug
- 3D rasters, PDAL integration

Get GRASS GIS 7.3 development version at
`grass.osgeo.org/download`

GRASS user mailing list
`lists.osgeo.org/listinfo/grass-user`

Paper and slides available at
`wenzeslaus.github.io/grass-lidar-talks`

Acknowledgements

Software

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

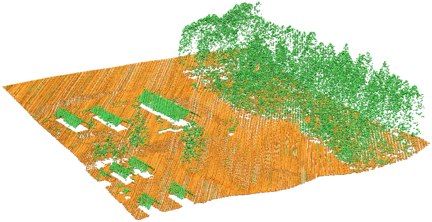
Thanks to users for feedback and testing, especially to Doug Newcomb, Markus Neteler, Laura Belica, and William Hargrove.

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*.