New lidar processing functionality in GRASS GIS 7.1

webinar for USFWS Remote Sensing group

Vaclav Petras (Vashek)

North Carolina State University, Center for Geospatial Analytics

NC STATE UNIVERSITY

February 22, 2016

Questions

- How many points are really necessary to create a detailed DEM?
- Which method of point decimation preserve more information?

Implementation

- Open source implementation for further review and improvement.
- Methods implemented in GRASS GIS so that they can be used by a broad audience.

Scripting

Python

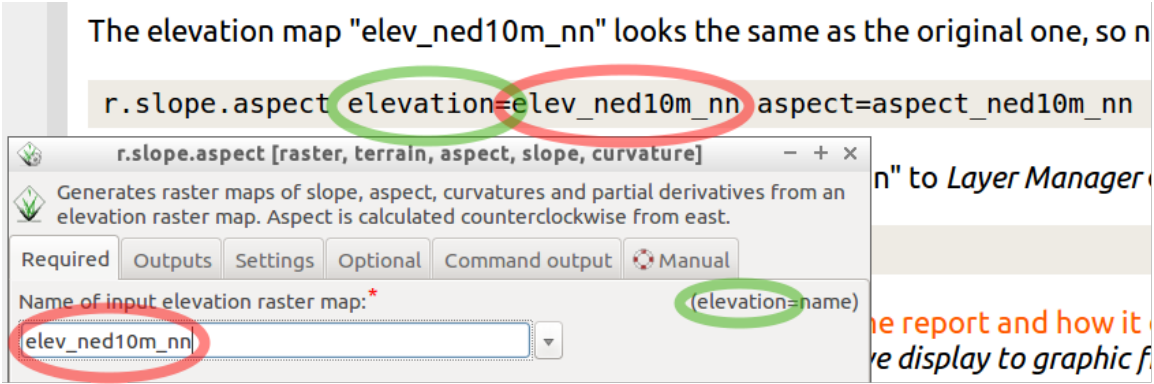
- simple scripting but also advanced programming
- `run_command('r.in.lidar', file=files.txt, output='surface', method='max', class_filter=[1, 2], flags='e')`

Bash (shell)

- simple syntax, easy to work with files, for simple task
- native to Linux, possible to use on MS Windows (e.g. MSYS)
- `r.in.lidar -e file=files.txt output=max method=max class_filter=1,2`

also R: `execGRASS("r.in.lidar", file = "files.txt", output="max", method="max", ...)`

GUI and scripting interface convergence

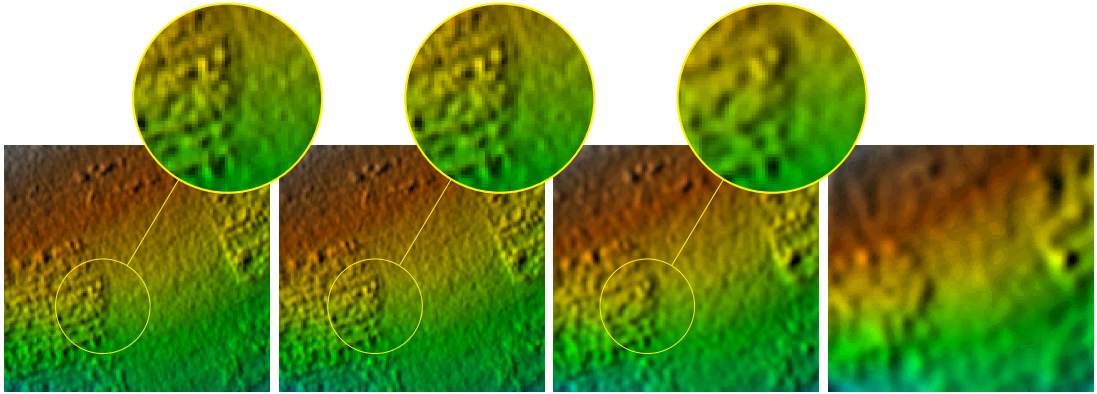


Typical lidar workflow

r.in.lidar

- depend on the side of output and type of analysis
- can be reduced by `percent` option
- on Linux available memory for process is RAM + SWAP partition

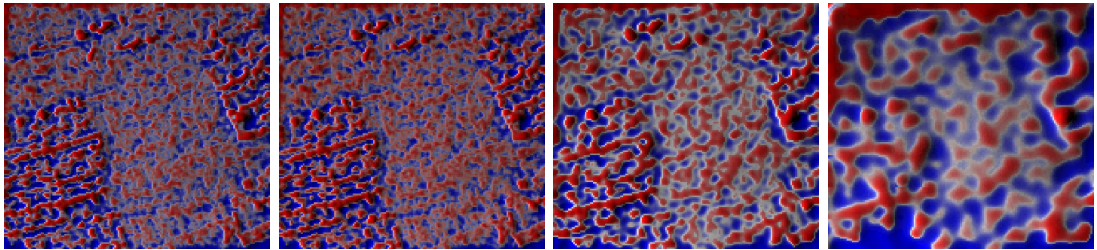
Count-based decimation influence on interpolated elevation



all	skip=5	preserve=20	preserve=100
0 %	20 %	90 %	99 %

```
g.region nsres=0.3 ewres=0.3 rows=149 cols=161 (cells=23989)
v.surf.rst ...  npmin=120 tension=20 smooth=2 segmax=40
```

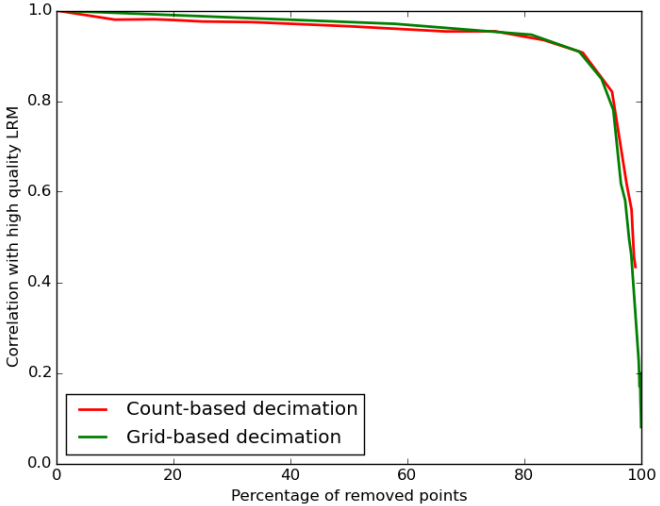
Count-based decimation influence on local relief model



all	skip=5	preserve=20	preserve=100
0 %	20 %	90 %	99 %

```
r.local.relief input=... output=... shaded_output=... neighborhood=11
```

Comparison of count-based and grid-based decimation



Merge point clouds as vector maps

v.patch: flags to work without topology and with z
v.lidar.mcc: do not build topology in 7.1 This is enabled by the change in v.patch. This makes it little bit faster.

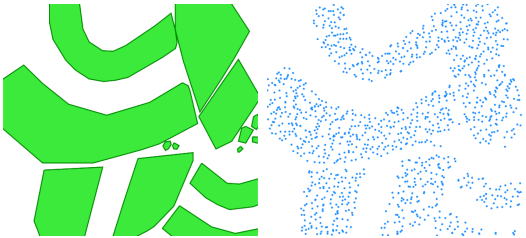
Crop the point cloud by polygon

v.in.lidar – limit the import to selected areas (2D)

areas

Crop the point cloud by polygon

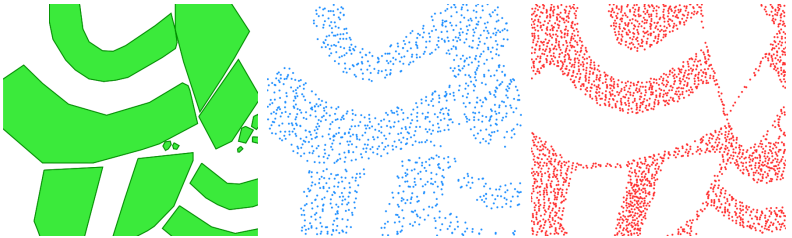
v.in.lidar – limit the import to selected areas (2D)



areas v.in.lidar mask=

Crop the point cloud by polygon

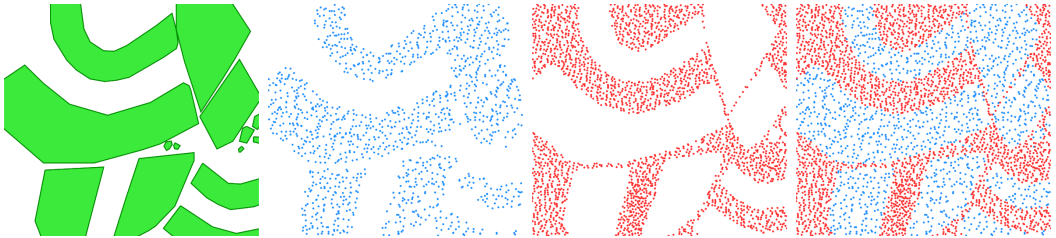
v.in.lidar – limit the import to selected areas (2D)



`areas` `v.in.lidar mask=` `v.in.lidar -i mask=`

Crop the point cloud by polygon

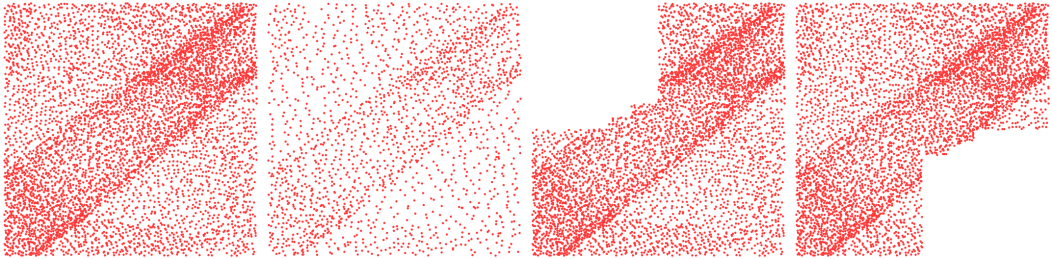
v.in.lidar – limit the import to selected areas (2D)



`areas` `v.in.lidar mask=` `v.in.lidar -i mask=` `v.patch -nz`

Count-based decimation

v.in.lidar – count-based decimation during import



v.decimate – point cloud decimation of vector maps (also supports grid-based decimation with preserving point properties)

Speed optimization

r.in.lidar

- choose computation region extent and resolution ahead
- have enough memory to avoid using `percent` option

v.in.lidar

- `-r` limit import to computation region extent
- `-t` do not create attribute table
- `-b` do not build topology (applicable to other modules as well)
- `-c` store only coordinates, no categories or IDs

Memory requirements

r.in.lidar

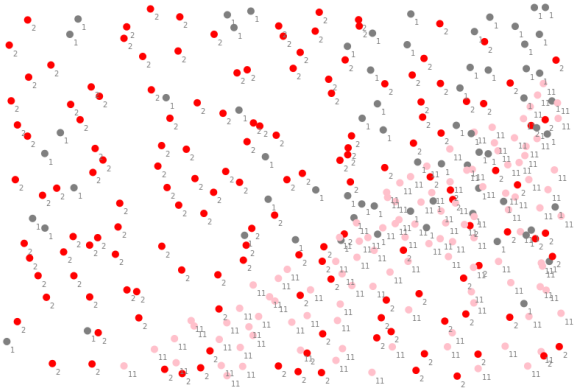
- depend on the side of output and type of analysis
- can be reduced by `percent` option
- on Linux available memory for process is RAM + SWAP partition

v.in.lidar

-

Store return and class information as category

v.in.lidar can store return or class information as category
using layers and categories for something else than ID and class



Also: read coordinates only – speed improvement (-c flag)

Binning of points from multiple LAS files

r.in.lidar – read multiple LAS files in one run

The original workflow

```
r.in.lidar input=tile_01.las output=tile_01
r.in.lidar input=tile_02.las output=tile_02
...
r.patch input=tile_01,tile_02,...  output=elevation
```

is replaced by

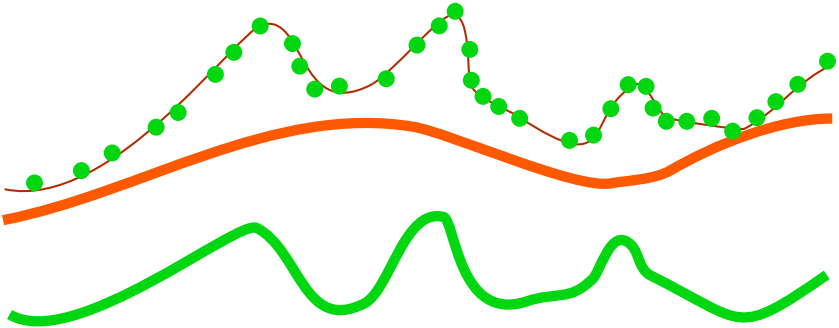
```
r.in.lidar file=tile_list.txt output=elevation
```

where `tile_list.txt` is

```
tile_01.las
tile_02.las
...
```

Compute height above a given raster during binning

r.in.lidar – derive height above ground of features

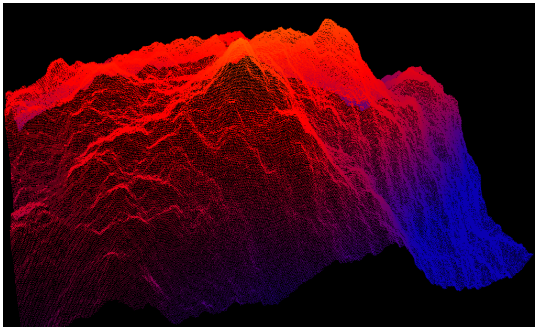


The resolutions of binning and ground raster can differ, so different statistics can be computed during binning.

Export vector points from GRASS GIS as LAS

v.out.lidar – export points in a vector map as lidar points

- visualization (plas.io, CloudCompare)
- further processing (PDAL, libLAS, CloudCompare, . . .)
- testing workflows with generated data



r.surf.fractal output in plas.io

Integration with PDAL

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

GRASS Lidar tools roadmap

- now: basic tools available in GRASS GIS 7.0
  - 7.0.3 released this January with 64bit support for MS Windows
- now: presented functionality available for testing in development version of GRASS GIS
  - daily build for MS Windows and Ubuntu
  - self-compiled version (simple for Fedora, CentOS, ... possible on Mac OS)
- summer: 3D raster, 2D display, smooth reprojections finished
- fall/winter: backport of stable functionality to 7.0 or release of 7.1

Summary

- count-based and grid-based decimation perform the same on a *given* point cloud
- analysis needed for every dataset → need for tool to create a report
- improvements needed for the project integrated into GRASS GIS

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