GRASS GIS Vector State of the Art – Gearing towards GRASS GIS 7

Markus Metz[1], Martin Landa[2], Anna Petrášová[3], Vaclav Petráš[3], Yann Chemin[4], Markus Neteler[1] and The GRASS GIS Development Team
[1] CRI, FEM, Italy, [2] CTU in Prague, Czech Republic, [3] NCSU, USA, [4] IWMI, Sri Lanka

Abstract

The upcoming GRASS GIS 7 release improves not only raster processing and general design but the vector processing in the first place. GRASS GIS, as a topological GIS, recognizes that the topology plays the key role in the vector processing and analysis.

Topology ensures that adjacent geographic components in a single vector map are related. In contrast to non-topological GIS, a border common to two areas exists only once and is shared between the two areas. Topological representation of vector data helps to produce and maintain vector maps with clean geometry as well as enables the user to perform certain analyses that can not be conducted with non-topological or spaghetti data. Non-topological vector data are automatically converted to a topological representation upon import. Further more, various cleaning tools exist to remove non-trivial topological errors.

In the upcoming GRASS GIS 7 release the vector library was particularly improved to make it faster and more efficient with an improved internal vector file format. This new topological format reduces memory and disk space requirements, leading to a generally faster processing. Opening an existing vector requires less memory providing additionally support for large files. The new spatial index performs queries faster (compared to GRASS GIS 6 more than 10 times for large vectors). As a new option the user can select a file-based version of the spatial index for large vector data. All topological cleaning tools have been optimized with regard to processing speed, robustness, and system requirements.

The vector engine comes with a new prototype for direct read/write support of OGR Simple Features API.
Additionally vector data can be directly exchanged with topological PostGIS 2 databases. This enables GRASS to read and write topological primitives beside native file-based format also to the topological PostGIS 2 databases.
Considering the wide spread usage of Esri Shapefile, a non-topological format for vector data exchange, it is particularly advantageous that GRASS GIS 7 offers advanced cleaning tools.

For power users and programmers, the new Python interface allows to directly access functions provided by the underlying C libraries; this combines the ease of writing Python scripts with the power of optimized C functionality in the library backend.

Topology support

The GRASS GIS native vector format stores objects in a topology format. The OGC Simple Features can be imported into and exported from the GRASS GIS format through topological vector conversion. For attribute management several database management system (DBMS) with SQL support are supported including SQLite (default DB backend), DBF, PostgreSQL, MySQL, ODBC.

Figure 1a: Basic and derived topological elements in GRASS GIS 7

The following **basic topological elements** can be edited directly: point, centroid, line, and boundary. A GRASS vector map can contain a combination of several different types of the elements. From these basic geometry types the following **derived topological elements** can be generated: area (closed ring of boundaries + centroid), isle (closed ring of boundaries, no centroid), and node (at both ends of lines/boundaries). Isles and Nodes are not visible to the user. Furthermore face, kernel (3D centroid) and volume (3D area) as defined in the format.

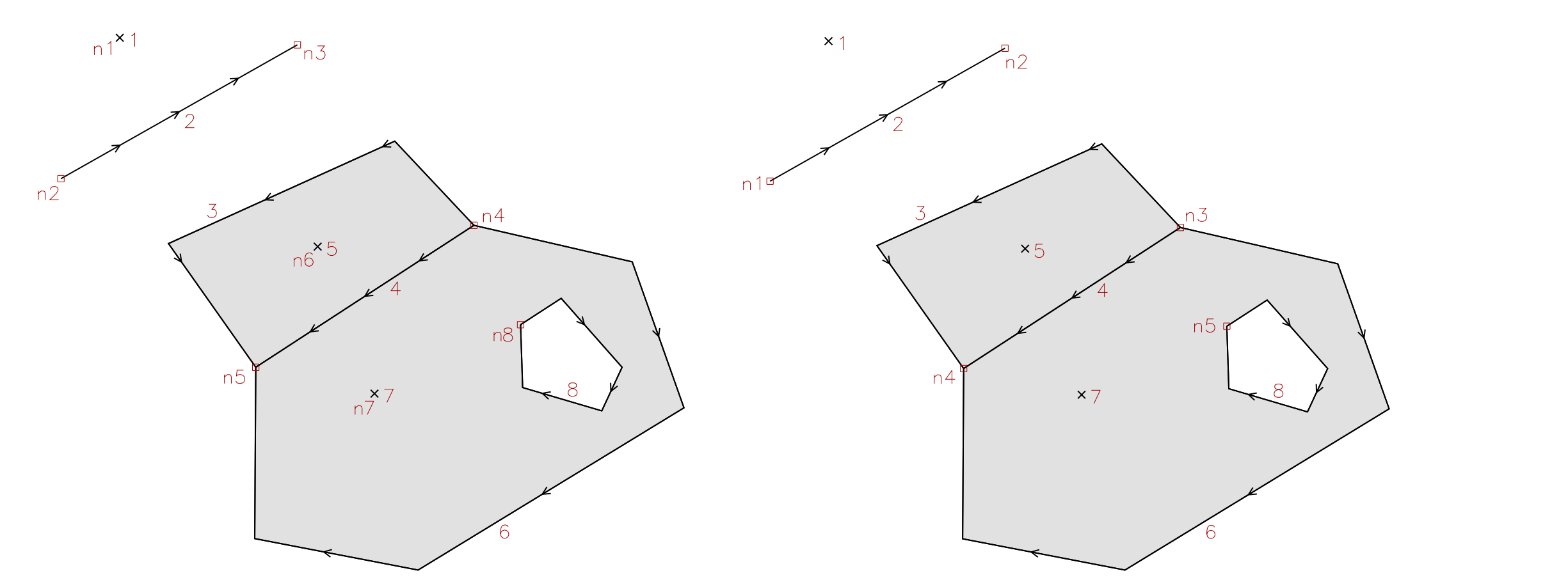


Figure 1b: Topology changes from version 6 to 7 after (points and centroids are represented by the nodes) [Landa 2013]

Data Sources

GRASS GIS 7 supports various data sources in read and write access. Beside native file-based raster and vector format GRASS allows to specify external data sources through GDAL/OGR library. GRASS GIS 7 also natively supports PostGIS database including its topological extension.

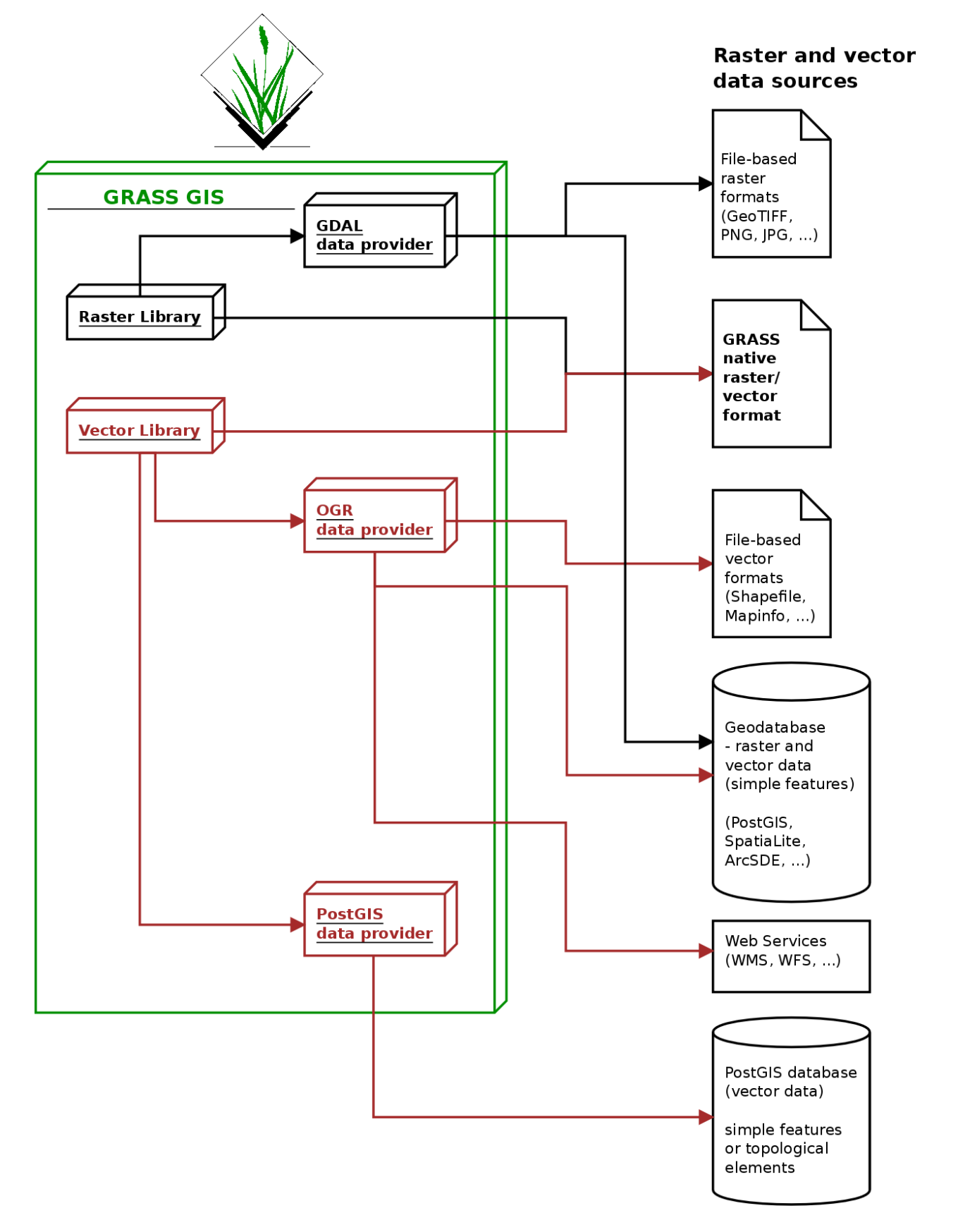

Raster and vector data sources
GRASS GIS
GDAL data provider
Raster Library
Vector Library
OGR data provider
PostGIS data provider
File-based raster formats (GeoTIFF, PNG, JPG, ...)
GRASS native raster/ vector format
File-based vector formats (Shapefile, Mapinfo, ...)
Geodatabase - raster and vector data (simple features) (PostGIS, SpatiaLite, ArcSDE, ...)
Web Services (WMS, WFS, ...)
PostGIS database (vector data) simple features or topological elements

Figure 2: Supported data sources

PyGRASS: fast Python API

PyGRASS [Zambelli 2013] is an object-oriented Python API which allows efficient manipulation with GRASS rasters and vectors. It is easy-to-use but its performance is comparable to C code since it calls GRASS C API behind the scenes.

```
# create a point
>>> point = Point()
>>> point.x = 150.65
>>> point.y = 368.50
>>> point
>>> Point(150.650000, 368.500000)
# distance to another point
>>> point.distance(Point(160.2, 372.6))
10.39290623454286
# conversion to Well-known text (WKT)
>>> point.get_wkt()
'POINT(150.650000 368.500000)'
# bounding box of a line
>>> line = Line([(5, 12), (6, 13), (4, 19)])
>>> line.bbox()
Bbox(19.0, 12.0, 6.0, 4.0)
```

Other Improvements & Additions

- **v.distance:** Calculates distances from points, lines, or areas to points, lines, or areas.
- **v.overlay:** The processing speed has been substantially improved.
- **v.net.*:** All vector network analysis tools provide now fine control over node costs.
- **v.voronoi:** New option to create Voronoi diagrams for areas.
- **v.rectify:** Vector data can now be georeferenced using various methods for 2D and 3D coordinate transformation.

GRASS GIS-PostGIS data provider: PostGIS 2 support

The native GRASS-PostGIS data provider allows the GRASS vector library to read and write PostGIS data directly without any external geospatial library. Beside simple features the provider also allows to work with topological elements through PostGIS Topology extension.

The support of PostGIS 2 Topology in GRASS GIS 7 is as follows:

- Points are stored as isolated nodes (*containing_face* is null),
- Centroids are stored as isolated nodes (*containing_face* is not null),
- Lines are stored as edges (*left_face* and *right_face* is 0),
- Boundaries are stored as edges,
- Areas are stored as faces (with id > 0),
- Isles are stored as faces (with id $<= 0$) (including universal face defined by PostGIS Topology).

Additional topological data related to nodes, lines, areas, and isles are stored in separated tables (see figure bellow).

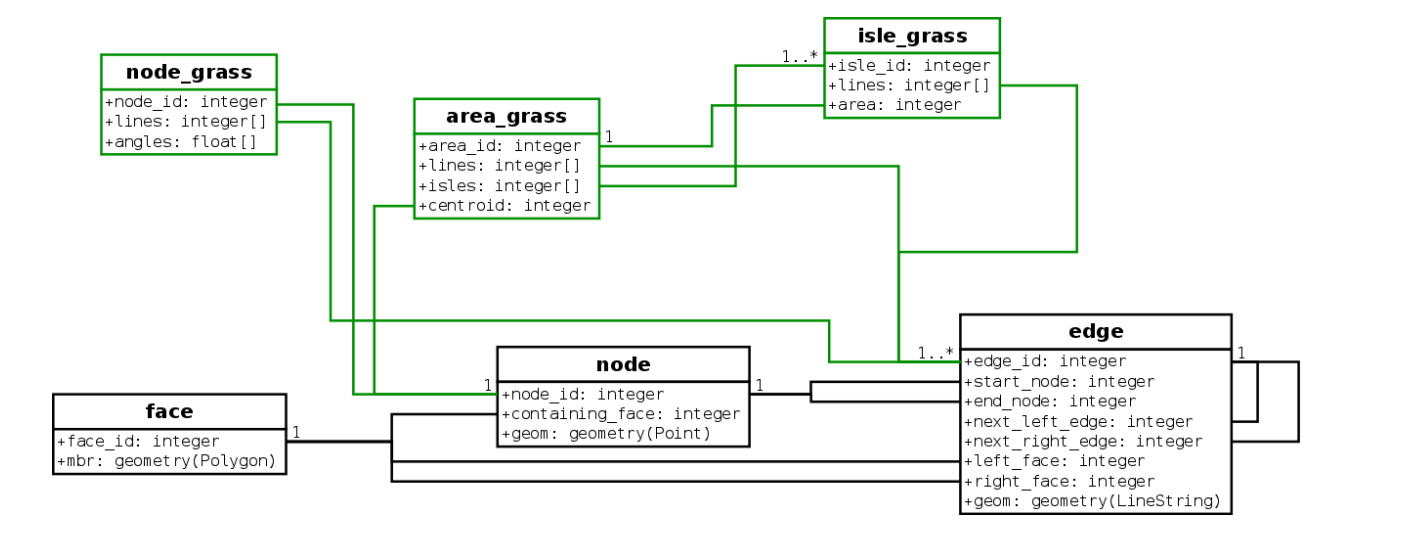


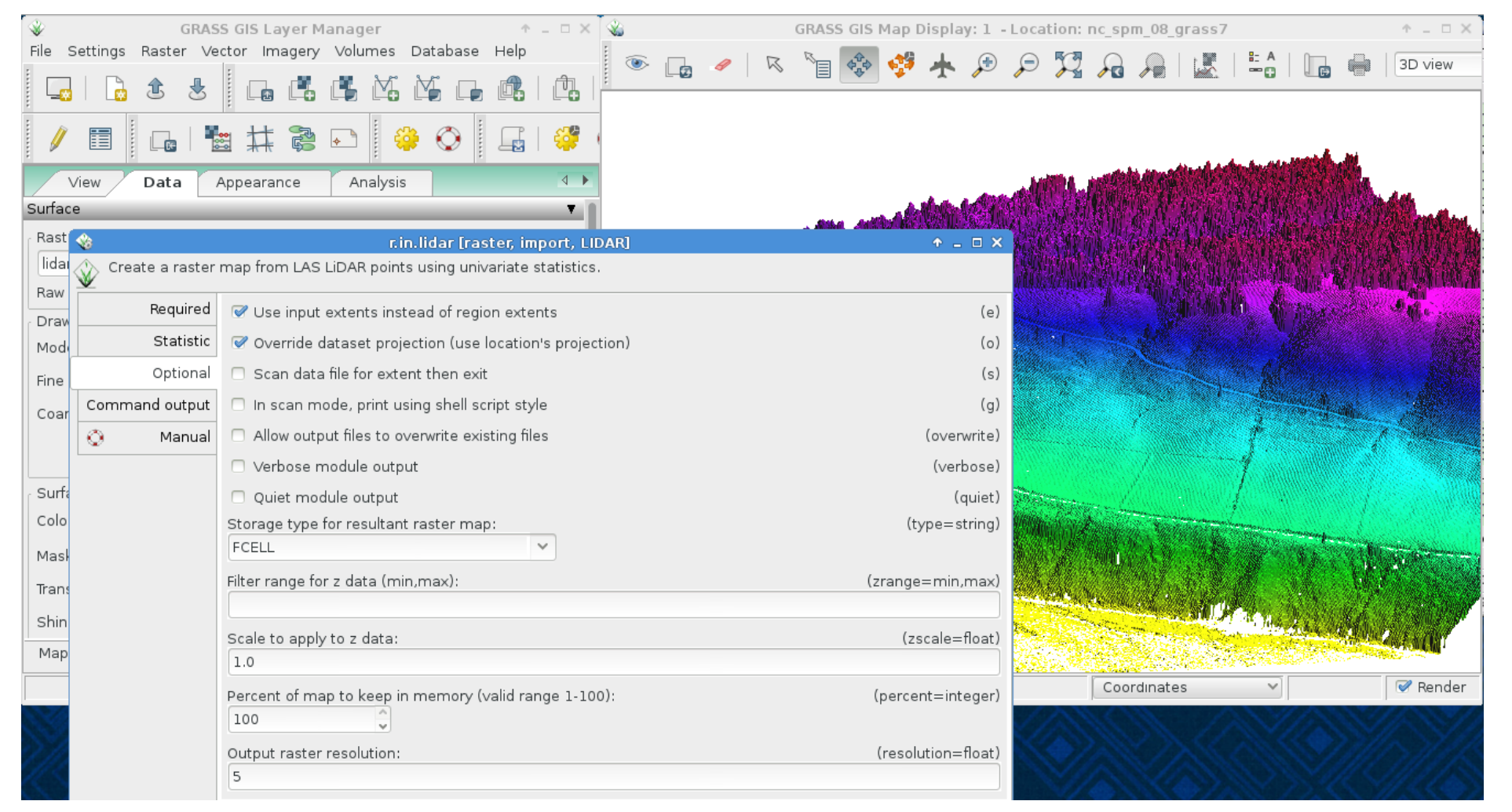
Figure 3: Extended PostGIS Topology structure

Dedicated modules

- **v.out.postgis:** Exports a vector map layer to PostGIS feature table.
- **v.external:** Creates a new pseudo-vector map as a link to a PostGIS feature table.
- **v.external.out:** Defines vector output format.

Lidar

The Lidar library (`www.liblas.org`) used by GRASS GIS permits the import of LAS Lidar data. The imported data can be in raster (**r.in.lidar** using statistics of choice) or in vector format (**v.in.lidar**).


GRASS GIS Layer Manager
File Settings Raster Vector Imagery Volumes Database Help
View Data Appearance Analysis
Surface
GRASS GIS Map Display: 1 - Location: nc_spm_08_grass7
3D view
Coordinates
Render

r.in.lidar [raster, import, LIDAR]
Create a raster map from LAS LiDAR points using univariate statistics.
Required | Statistic | Optional | Command output | Manual
[✓] Use input extents instead of region extents (e)
[✓] Override dataset projection (use location's projection) (o)
[ ] Scan data file for extent then exit (s)
[ ] In scan mode, print using shell script style (g)
[ ] Allow output files to overwrite existing files (overwrite)
[ ] Verbose module output (verbose)
[ ] Quiet module output (quiet)
Storage type for resultant raster map: (type=string) FCELL
Filter range for z data (min,max): (zrange=min,max)
Scale to apply to z data: (zscale=float) 1.0
Percent of map to keep in memory (valid range 1-100): (percent=integer) 100
Output raster resolution: (resolution=float) 5

Figure 4: Example for LAS support in GRASS GIS 7: rapid LAS data assessment through binning

On-farm water storage study with lidar data in NSW (Australia) developed a full remote sensing monitoring methodology of water availability with lidar-based bathymetric survey and multi-source remote sensing survey [Chemin 2011].

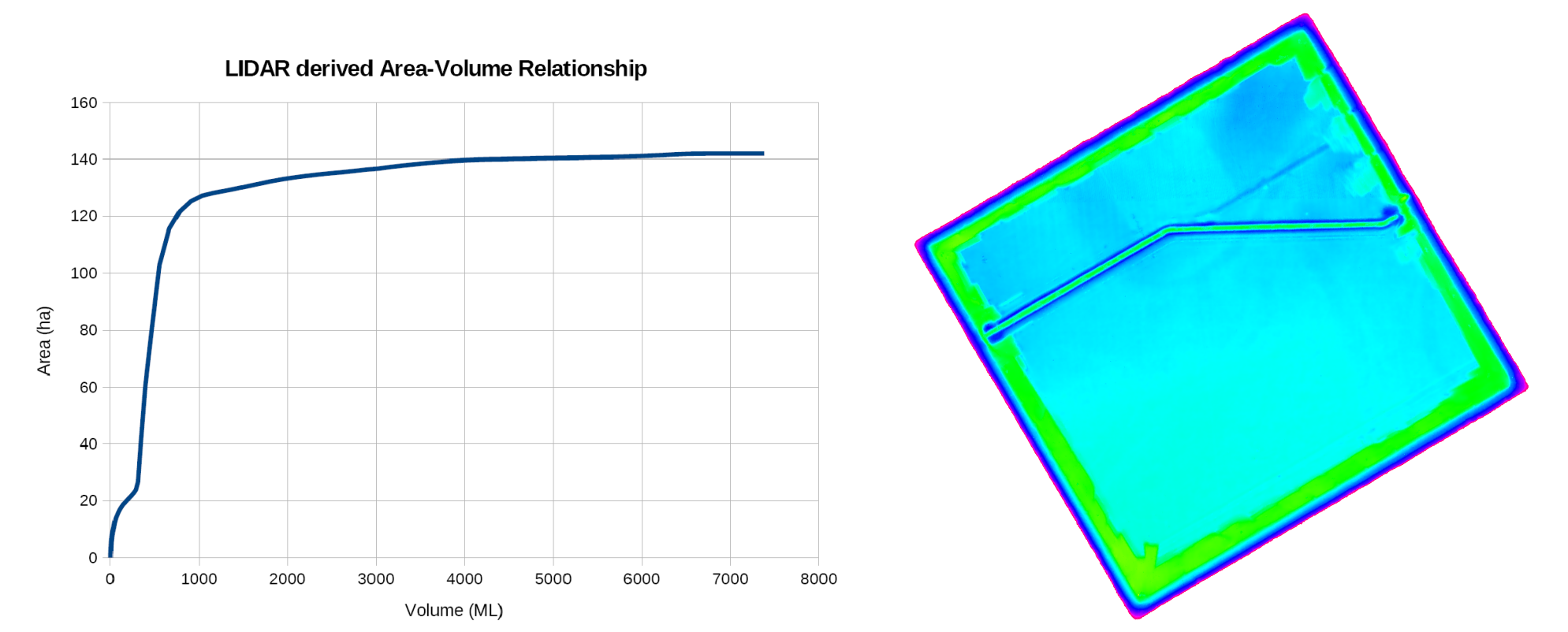


Figure 5: On-Farm-Water-Storage Lidar survey and Depth-Volume-Area surveying [Chemin 2011]

3D

GRASS 7 has 3D topological capabilities, as described in Figure 5a. An application to buildings is found in Figure 5b.

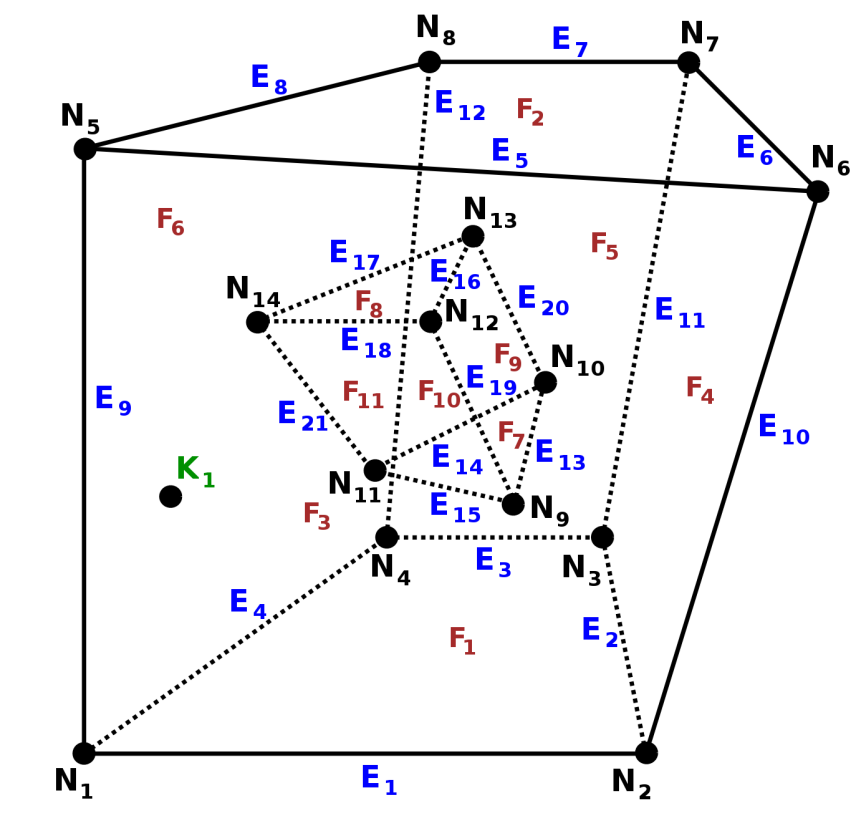

N_1 N_2 N_3 N_4 N_5 N_6 N_7 N_8 N_9 N_{10} N_{11} N_{12} N_{13} N_{14}
E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11} E_{12} E_{13} E_{14} E_{15} E_{16} E_{17} E_{18} E_{19} E_{20} E_{21}
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 F_{10} F_{11}
K_1

Figure 5a: 3D object with a hole in GRASS GIS topology model

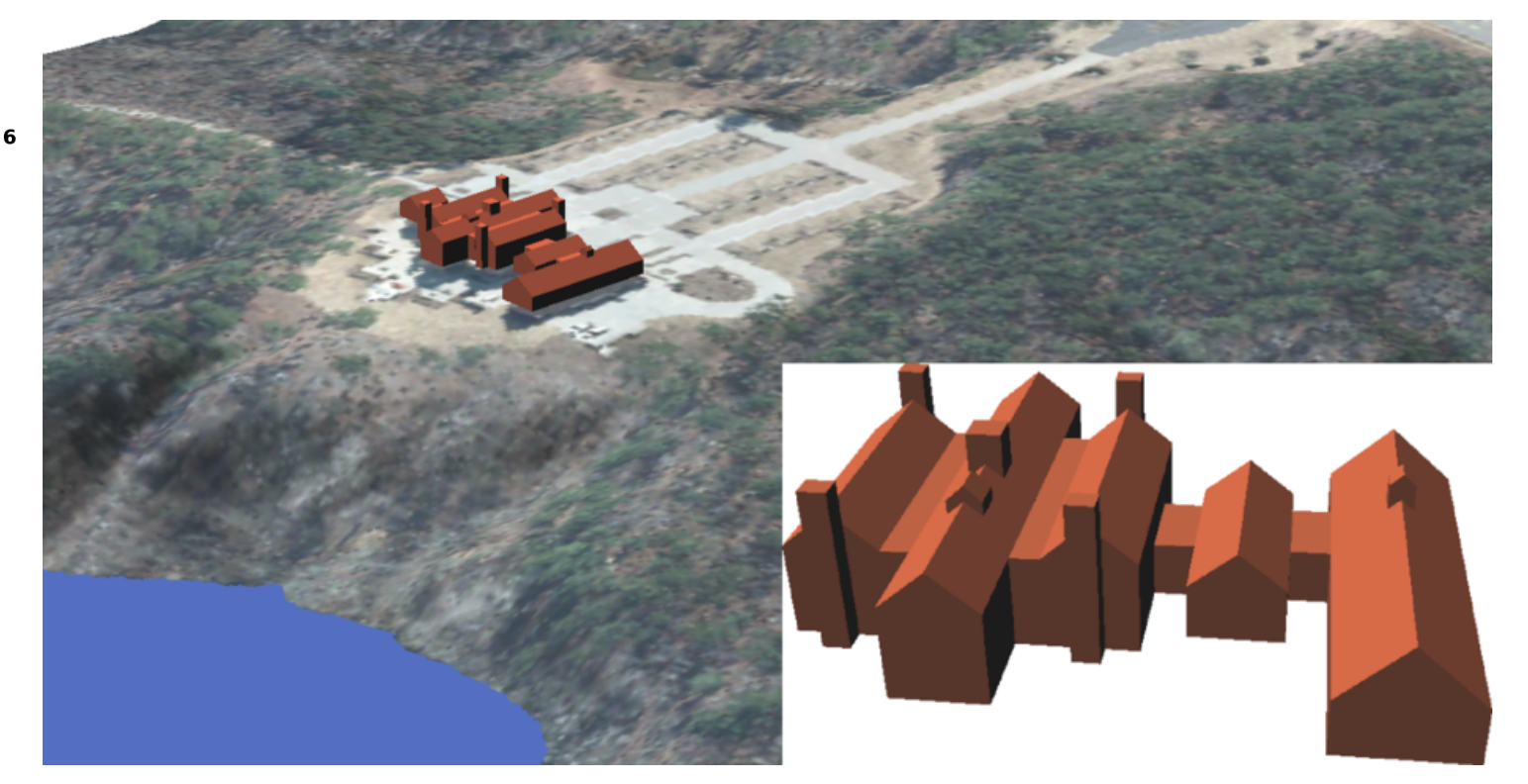
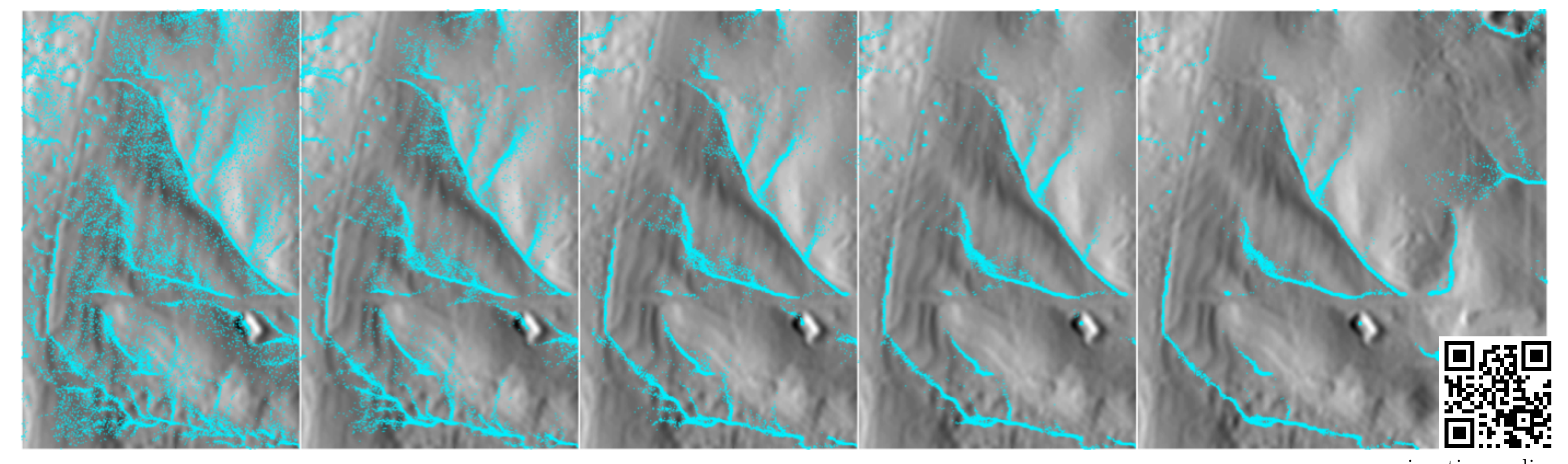


Figure 5b: 3D model of Chancellor's House (NC State University) visualized in GRASS GIS

TGRASS: temporal framework

TGRASS [Gebbert 2014] provides support for large spatio-temporal data handling and analysis, and is fully integrated into GRASS GIS 7. It introduces the concept of space-time dataset as a series of vector, raster, or 3D raster data with temporal metadata. In the simple example below, we computed a series of vector data representing particles in a water flow simulation, created a space-time dataset and registered the vectors to this dataset based on time stamps assigned during the vector creation. The result can be then quickly visualized using GRASS GIS Animation Tool.

```
r.sim.water -t elevation=elev_lid792_1m dx=dx dy=dy depth=depth outwalk=walker outiter=1
t.create output=particles type=stvds temporaltype=relative semantictype=mean title="Particles" desc="Particles"
t.register input=particles maps=`g.mlist --q type=vect pattern=walker* separator=,` type=vect
g.gui.animation stvds=particles
```



see animation online

References

[Chemin 2011] Chemin & Rabbani, 2011. International Journal of Geoinformatics, 7(3):1-6.
[Gebbert 2014] Gebbert & Pebesma, 2014. TGRASS: A temporal GIS for field based environmental modeling, Environmental Modelling & Software 53:1–12.
[Landa 2013] Landa, 2013. Vektorová architektura systému GRASS GIS [GRASS GIS Vector Architecture]. PhD thesis, CTU in Prague, Czech Republic.
[Neteler 2005] Neteler & Grasso & Michelazzi & Miori & Merler & Furlanello, 2005. International Journal of Geoinformatics, 1(1):51-61.
[Neteler 2012] Neteler & Bowman & Landa & Metz, 2012. Environment & Modeling Software, 31:124-130
[Zambelli 2013] Zambelli & Gebbert & Ciolli, 2013. PyGRASS: An Object Oriented Python API for GRASS GIS. ISPRS International Journal of Geo-Information 2.1:201-219.

www.osgeo.org

grass.osgeo.org