Chapter 9
GRASS GIS

M. Neteler, D.E. Beaudette, P. Cavallini, L. Lami and J. Cepicky

Abstract GRASS is a full featured, general purpose Open Source geographic information system (GIS) with raster, vector and image processing capabilities. There has been constant development of the software since 1982, with recent major improvements reflecting renewed efforts by the international development team to make it one of the core components of the Open Source geospatial software stack. It can handle 2D and 3D raster data, includes a topological 2D/3D vector engine, network analysis functions, and SQL-based attribute management. This chapter presents an overview and practical examples of the GRASS 6 capabilities relevant to environmental and planning applications including new functionality. Enhancements to 3D visualization and approaches to environmental models are also discussed, as well as image processing routines pertaining to LIDAR and multi-band imagery. Integration of GRASS with other Open Source software packages for geostatistical analysis, cartographic output and Web GIS applications are described. Trends for future development are also discussed.

M. Neteler
Fondazione Mach - Centre for Alpine Ecology, 38100 Trento (TN), Italy,
e-mail: neteler@osgeo.org

D.E. Beaudette
Department of Land, Air and Water Resources, University of California, Davis, CA 95616, USA,
e-mail: debeaudette@ucdavis.edu

P. Cavallini
Faunalia, Piazza Garibaldi 5, 56025 Pontedera (PI), Italy, e-mail: cavallini@faunalia.it

L. Lami
Faunalia, Piazza Garibaldi 5, 56025 Pontedera (PI), Italy, e-mail: lami@faunalia.it

J. Cepicky
Help Service - Remote Sensing s.r.o., Cernoleska 1600, 25601 - Benesov, Czech Republic,
e-mail: jachym.cepicky@gmail.com

G.B. Hall, M.G. Leahy (eds.), *Open Source Approaches in Spatial Data Handling*.
Advances in Geographic Information Science 2,

9.1 Introduction

The Geographic Resources Analysis Support System (GRASS, http://grass.osgeo.org) was born in the early 1980s at the Construction Engineering Research Laboratory (CERL) of the United States Army as a software management tool for military applications. It has evolved into one of the most comprehensive, general purpose free and open source for geospatial (FOSS4G) systems in existence. GRASS is a raster/vector geographic information system (GIS) combined with integrated image processing and data visualization subsystems. It includes hundreds of modules for management, processing, analysis and visualization of geo-referenced, spatial data. It includes unique algorithms and methods that are implemented in a portable environment making it one of the few GIS in the world that works under a variety of different operating systems and platforms. GRASS is fundamentally a desktop GIS, however recent efforts are in place to extend functionality to Web-based and network-programmable interfaces.

The limitations of computer hardware in the early 1980s, coupled with the innovation of skilled and motivated young researchers lead to a first working version of GRASS. It was designed as a modular system written in the C programming language. The advent of the Internet facilitated the spread of the software among universities and other governmental organizations. As CERL and GRASS evolved through the late 1980s and early 1990s, CERL created the Open GRASS Foundation which evolved into the Open GIS Consortium (OGC, now known as the Open Geospatial Consortium). The discontinuation of development of GRASS by CERL in 1996 lead to the formation of the GRASS Development Team two years later.

A first release as Free Software under the GNU General Public License (GPL) was published in October 1999. Based on academic efforts, GRASS is today developed by a worldwide group of scientists, programmers, power users and enthusiasts. The project is attractive to a new generation of users and developers due to the modernization of the software. The release of GRASS 6 reflects new efforts to one of the core components in the FOSS4G software stack. Quality management is enforced by transparent development methods such as a centralized source code repository (in a concurrent versions system (CVS) server) and the immediate peer-review of source code changes by email broadcast.

In 2006, the Open Source Geospatial Foundation (OSGeo, http://osgeo.org) was created to bundle several FOSS4G software projects, including GRASS, into a joint organization. This chapter presents an overview of the GRASS 6 capabilities relevant to environmental and land management applications, including the updated and new tools for 2D and 3D raster data, the vector modules based on the redesigned topological 2D/3D vector engine, and SQL-based attribute management. Approaches to linking with other FOSS4G tools and environmental models are also discussed.

Table 9.1 Functionality classes of GRASS commands

Prefix	Function Class	Type of Command
d.*	display	graphical output
db.*	database	database management
g.*	general	general file operations
i.*	imagery	image processing
m.*	misc	miscellaneous commands
ps.*	postscript	map creation in Postscript format
r.*	raster	2D raster data processing
r3.*	3D raster	3D raster data processing
v.*	vector	2D and 3D vector data processing
d.*	display	graphical output
db.*	database	database management
g.*	general	general file operations
i.*	imagery	image processing
m.*	misc	miscellaneous commands

9.2 The Structure of GRASS

GRASS GIS is a suite of modules designed to facilitate rapid analysis of raster, vector and tabular data. These modules were designed to be as robust and efficient as possible at performing a specific task. Following the UNIX methodology of chaining together multiple small programs to perform complex tasks, spatial analysis is usually performed in GRASS by linking together multiple modules either directly through pipes (an operating system construct, used to pass data between processes), or indirectly through intermediate files. Each of the modules expects a small number of command line arguments, and either returns a new spatial data file (raster, vector, etc.), or modifies an existing map or attribute table in-place. Commands are clearly organized by functionality through a command class prefix shown in Table 9.1.

Although this approach gives tremendous power to a user who learns to wield the full set of GRASS modules, it does not lend itself well towards the graphical user interface-driven desktop application that most people are familiar with. Indeed, the core GRASS code is better described as a professional analytical engine rather than a desktop GIS. However, several graphical interfaces to the command line-driven modules have been produced over the years with the TclTk toolkit, and more recently with the wxWidgets Python toolkit (http://www.wxwidgets.org). GRASS contains several libraries dedicated to simplifying access to and manipulation of very large files including, the “rowio” library, “segments”/“spatial index” libraries (GRASS raster and vector data), and interoperability (reading and writing external formats such as GeoTIFF). A parallelized numerical library has recently been added to the GRASS code base, which assists with converting complex single-thread algorithms into multi-threaded versions that can fully take advantage of multi-processor or multi-core hardware.

9.2.1 Installation

GRASS software can be downloaded freely from the main GRASS Web site (http://grass.osgeo.org). This site is mirrored in several countries for faster access, including the United States GRASS mirror at http://grass.ibiblio.org. Source code (the portable version for all operating systems) as well as the latest ready-to-install binaries for GNU/Linux, MacOSX and MS-Windows (native or optionally with the Cygwin tools) are all available on this site. GRASS is also available on CDROM/DVD from various providers. The MS-Windows version of the popular Quantum GIS (QGIS, http://qgis.org) package includes a native GRASS installation. QGIS is a user friendly geographic data viewer with some analytical capabilities that runs on GNU/Linux, Unix, MacOSX, and MS-Windows. It supports vector, raster, database formats, OGC Web Services and includes a GRASS toolbox. QGIS is licensed under the GNU GPL.

9.2.2 Functionality and Command Structure

GRASS contains over 300 programs and tools to render maps and images both on-screen and on paper, to manipulate raster and vector data, to process multi spectral and time series image data, and to create, manage, and store spatial data. As noted earlier, GRASS uses both an intuitive graphical user interface as well as command line syntax for ease of operations. Approximately 200 of the modules that are available in GRASS GIS are integrated in pull down menus. The most frequently used modules can thus be easily accessed with the use of a mouse.

GRASS 6 introduces a new topological 2D/3D vector engine and support for vector network analysis. Attributes are managed in DBF files or a SQL-based database management system (DBMS). The NVIZ visualization tool displays raster data, 2D and 3D vector data as well as voxel volumes. GRASS project databases ("locations") can be auto-generated by a European Petroleum Survey Group (EPSG) code number or from geo-coded data sets. GRASS is fully integrated with GDAL/OGR libraries to support an extensive range of raster and vector data formats (see Chap. 5 of this book), including the OGC Simple Features specification. To facilitate usage by non-English speakers, user messages have been translated to various languages including Asian languages. GRASS supports work groups through its *LOCATION/MAPSET* directory structure concept which can be set up on shared/network disk devices. Through this, team members can simultaneously work in the same project's database.

9.3 GRASS Features

Although GRASS has evolved into a general purpose GIS, its major strength remains environmental modeling and analysis. With respect to older versions, signifi-

cant improvements have been implemented to strengthen its viability. In this regard, interoperability is seen as a crucial element in GRASS development, leveraging from widely used data exchange libraries. Key motivations for using the software are its strong analytical capabilities for both raster and vector data (in 2D and 3D space) as well as graphical data exploration and visualization.

9.3.1 Data Exchange: Interoperability

GRASS data are maintained in their own directory structure in so-called "locations" and "mapsets". The idea behind this structure is to provide a multi-user environment with access control. Locations can be maintained on a centralized server. Extensive capabilities of data exchange are essential for daily GIS work. For interoperability, GRASS profits from an external project, the GDAL/OGR library, which allows the conversion among many raster and vector formats, including the internal GRASS formats. This library is also used by global data vendors as well as in some proprietary GIS applications. Additional GRASS modules allow importing from and exporting to other formats.

GRASS uses a topological vector architecture. To support Simple Features vector data (conformal to OGC standards), these data sets are converted upon data import. Conversely, export into Simple Features is also possible. Instead of importing data sets, vector maps may also by linked to the GRASS database as virtual maps. The module for importing vector maps, *v.in.ogr*, uses the OGR library (for a full list of supported formats, see the GDAL/OGR Web site, http://www.gdal.org/). This module also allows merging of a series of different vectors, limiting the import to a user defined spatial subset, and creating a new GRASS location on the basis of the projection system of the imported data. Additional modules allow the management of a variety of data formats, including ASCII files, 2D and 3D drawings from computer assisted design (CAD) software, common vector formats, Gazetteer and global positioning system (GPS) data, as well as data import from Web Feature Services (WFS).

The *r.in.gdal* module, which uses the GDAL library, enables raster datasets to be imported. The module also allows the import of single bands from multi-band imagery, and the creation of a new GRASS location from the imported dataset. Additional modules allow the import of various ASCII and binary files including all common raster and imagery formats including elevation models, aerial and satellite imagery, and data from Web Mapping Services (WMS). It is also possible to create 3D raster maps from 2D map slices, or from importing 3D volume files.

Once in GRASS, raster maps can be converted to vector maps and vice versa. Vector geometries can also be converted between different types (points, lines, polygons), and a series of raster layers can be transformed into raster volumes and vice versa. 3D points can also be converted or interpolated to raster volumes. Exporting files is essentially similar, with the same formats supported (with minor differences). GRASS raster and vector maps can be exported also to rendering software (e.g., POV-Ray or Paraview). 3D fly-throughs can be created as MPEG format animations.

9.3.2 Raster Data Analysis: Pixels and Voxels

Raster analysis is the historical core of the GRASS project. In addition to 2D raster maps (pixel-based), GRASS can also manage 3D (volume) raster maps (voxel-based). Many modules are available for raster processing, and they can be subdivided into the following groups:

- *Map management*: editing, extent, management of null values, resampling, reprojecting etc.;
- *Colour management*: setting colour tables, weighted merging and splitting of RGB/HIS and normal maps;
- *Buffer creation*: single and multiple buffers;
- *Mask creation*: to apply analyses to a limited subset of the working map;
- *Proximity analysis*: analysis through a moving window to create maps in which every cell is the result of a function applied to the surrounding cells; calculation of minimum distances among raster objects;
- *Map overlay*: statistics, temporal series, patching maps, etc.;
- *Solar radiance and shadowing*: calculation of solar irradiation and shadows on the basis of date and time;
- *Terrain analysis*: calculation of total cumulative cost of movement over a digital terrain model (DTM) or a cost map, search of a minimum cost path, profile analysis, calculation of exposure and slope maps, texture analysis, visibility analysis;
- *Raster transformation*: finding islands of contiguous cells of the same type, growing and linearizing raster islands;
- *Specialized models*: hydrology, landscape ecology, fire spread;
- *Reclassification*: on the basis of user rules, of island size, recoding and rescale;
- *Random raster maps*: generating raster points (also on the basis of a second raster map);
- *Surface generation*: density maps (kernels) from vector points, planes, fractal surfaces, Gaussian derivates, random surfaces with spatial dependency;
- *Surface interpolation*: bilinear, bicubic, inverse distance weighted (IDW), and regularized splines with tension (RST);
- *Reports and statistics*: general statistics, correlation and linear regression between raster maps.

GRASS raster map processing is always performed in the current region settings. That is, the current region extent and current raster resolution is used. If the resolution differs from that of the input raster map(s), on-the-fly re-sampling is performed (nearest neighbour re-sampling). If this is not desired, the input map(s) has/have to be re-sampled beforehand with one of the dedicated modules. GRASS applies two general rules in this context:

1. Raster/imagery output maps have their bounds and resolution equal to those of the current region.
2. Raster/imagery input maps are automatically cropped/padded and rescaled (using nearest-neighbour resampling) to match the current region.

9.3.3 Vector Data Analysis

The vector engine of GRASS has been completely overhauled during the last few years. Geometry and attribute management are clearly separated, giving more flexibility in data storage. The vector geometry was extended to manage 2D and 3D vector data. Vectors are fully topological, with spatial relationships (e.g., connection, adjacency, inclusion) embedded in the information linked to each vector element. For instance, a border line between two polygons is not replicated, but is singular, and linked to the left- and right-sided polygon centroids. This new internal vector data format is also portable across 32bit and 64bit platforms.

A new spatial index system as well as category index accelerates data access. The support of topology enables the user to perform data cleanup to ensure data consistency. Support for vector map overlays, intersections and extraction of features is implemented. A new digitizing tool permits users to create or update vector features with attributes on-screen. The attribute management includes full and flexible integration of database management systems (PostgreSQL, MySQL, DBF, SQLite and ODBC are currently supported). Structured query language (SQL) statements, used to manage attributes, are directly passed to the underlying database system. The set of supported SQL commands depends on the RDMBS and selected driver. Graphical updating of vector attributes has been implemented as well. Available vector data types are:

- *Point*: a single data location;
- *Line*: a directed sequence of connected vertices with two endpoints called nodes;
- *Boundary*: the border line to describe an area;
- *Centroid*: a point within a closed boundary;
- *Area*: the topological composition of centroid and boundary;
- *Face*: a 3D area;
- *Kernel*: a 3D centroid in a volume (*in development*);
- *Volume*: a 3D corpus, the topological composition of faces and kernel (*in development*).

GRASS vector maps may include different vector types (points, lines, areas etc.) in the same layer. Available vector analysis methods are quite advanced, and are categorized into several groups:

- *Map management*: editing, topology rebuilding, cleaning up of non-topological imported vectors, adding centroids to polygons, merging and splitting of elements, geometry conversions (2D and 3D; rebuilt on the basis of a DTM; extrusion), re-projection;
- *Attribute management*: connecting vectors to attribute tables (on various database backends);
- *Reports and statistics*: general statistics, export of geometric characteristics to database, normality tests on point distribution, correlation of values and corresponding raster cells;

- *Extraction and selection*: on the basis of a geometry or a table query, selection of a vector on the basis of the relations with the geometries of another vector buffer;
- *Proximity analysis*: calculation of minimum distance matrix among objects;
- *Map overlay*: patching and overlaying vectors, management of attributes;
- *Generation of reference maps*: creation of grids of polygons or points;
- *Reclassification*: management and reclassification of categories associated to geometries;
- *Points*: import of point vectors from an x,y[,z] table, creation of random points, minimum convex polygons, Voronoi tessellation and Delaunay triangulation, raster values extraction;
- *Lidar data analysis*: outlier detection, Digital Surface model (DSM) and DTM separation and interpolation;
- *Network analysis*: creation of a network, finding the shortest path, allocation of sub-nets for nearest centers, Steiner tree, or resolution of the traveling salesman problem;
- *Linear referencing system*: creation, editing and management.

9.3.4 Attribute Data Storage via a DBMS

GRASS provides tools for management and analysis of vector maps, including the attributes that can be stored in a DBMS. GRASS can be connected to various relational DBMS (RDBMS) and embedded databases. It supports SQL to create, retrieve, update, and delete data from RDBMS. GRASS unifies the different drivers in an abstraction layer named the database management interface (DBMI) to assist the user.

Usually an attribute table is linked to vector geometry data. The attribute table must contain a column named “cat” to store the category numbers (the vector IDs) which connect the individual vector objects to attributes. Each table row corresponds to a category number. Several vector objects can be assigned to the same category numbers (table row).

It is possible to link the geographic objects in a vector map to one or more tables. Each link to a distinct attribute table is called a layer. A link defines the database driver, the database name, and the table name that are used. Each category number in a geometry file corresponds to a row in the attribute table. Using *v.db.connect*, layers can be listed or maintained. GRASS layers do not contain any geographic objects, but they consist of links to attribute tables in which vector objects can have zero, one, or more categories. If a vector object has zero categories in a layer, then it does not appear in that layer. Some vector objects may appear in some layers but not in others. The practical benefit of this approach is that it allows placement of thematically distinct but topologically related objects into a single map (e.g., forests and lakes). These virtual layers are also useful for linking time series attribute data to a series of locations that do not change over time. By default the first layer is

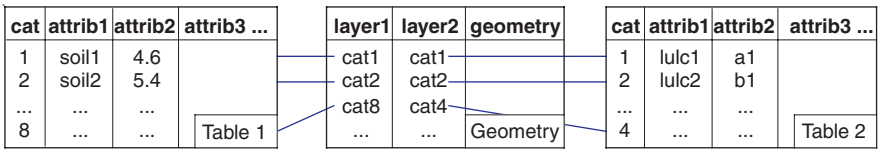

Fig. 9.1 Example for layer concept in GRASS vector files: geometry file with two layers connected to two different attribute tables (linking field "cat")

active (i.e., the first table corresponds to the first layer). Further tables are linked to subsequent layers. Figure 9.1 illustrates the layer concept with an example for a vector map with two layers, connecting two different attribute tables to the vector geometry.

9.3.5 Visualization: 2D, 3D and Animations

Besides the creation of 2D maps, GRASS also provides the possibility to create impressive 3D visualizations and animations of surfaces and volumes using the NVIZ module (Fig. 9.2). This module uses raster data to define both height and attribute data, and to allow the overlay of vector data (points, lines and areas). It also has illumination tools to add shadow effects. After starting NVIZ, a graphic window and a control window are opened, and the individual features are displayed. A series of images can be saved and combined into a GIF animation or an MPEG movie file via external programs. This is especially interesting when analyzing changes over time.

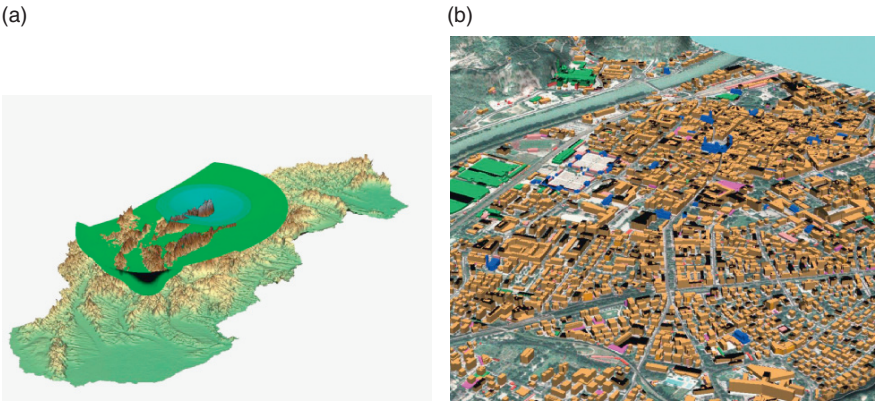

Fig. 9.2 Some examples of how NVIZ displays both volumetric (**a**) and conventional 3D (**b**) raster and vector data types

9.4 Raster Applications

Given the raster data processing origins of GRASS, it is not surprising that the raster modules are among the oldest and best developed in its toolbox. Several common applications of the GRASS raster modules such as surface analysis, geomorphometric classification, interpolation, and solar radiation modeling are included in this section. A more complete discussion of these modules can be found in Neteler and Mitasova (2008).

9.4.1 Cost Surface Analysis: Wilderness Navigation

Cost surface analysis is a method of automatically identifying a path which minimizes the amount of accumulated "cost" between a series of points. Conceptually, this approach is very similar to the actions of a mountain hiker seeking to minimize the total change in slope encountered while traversing an area. It follows the principle that walking *up or down* steep hills is hard, and walking a *long distance* up or down a steep slope is even harder. In GRASS, the modules *r.cost* and *r.walk* are used to compute accumulated cost from some start point to an end point. The module *r.drain* is then used to identify the path of least cost between end and start points, akin to the movement of a drop of water down an elevation gradient. A slope map is commonly used as an input to *r.cost*. However, more complex systems can be modeled by altering a slope map with *r.mapcalc*.

Consider the example of a trip through rugged terrain between numerous subalpine lakes. The region is mostly wilderness, and therefore trails are few and do not directly connect specific lakes that fall along a desired route. Identifying the paths between points of interest which minimize slope traversal has been done with paper topographic maps for decades. Cost surface analysis is well suited to this type of problem. For example, the task of navigating between Bill Lake and Bonnie Lake shown in Fig. 9.3(a) is a good case in point. In this example, additional sources of "cost" are added to the slope map as follows:

- traversing lakes is not acceptable (extreme travel cost added to slope map);
- traversing wooded areas is faster due to shading (moderate travel cost subtracted from slope map).

By specifying start and stop points, along with the composite travel "cost" map, the least-cost path between these points can be computed via the GRASS commands *r.cost* and *r.drain*. It is possible to see how *r.drain* actually simulates the flow of a drop of water down an elevation surface in an energy minimizing operation (Fig. 9.3(b)). However, in this case, the elevation surface is actually a travel cost map (computed by *r.cost*). The path of the theoretical drop of water represents the least-cost route from the starting point to the end point. Note that the start point for *r.cost* (point of origin) is actually the end point for *r.drain*. By specifying very large travel costs for traversing lakes (peaks), and slightly lower travel costs for wooded areas, the computed least cost path goes "around" lakes and "through" wooden areas.

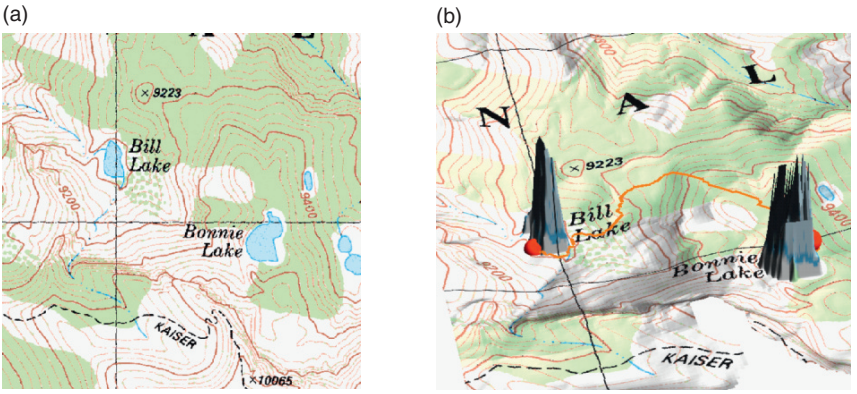

Fig. 9.3 An example least-cost path calculation. A hiker wishes to travel from Bill Lake to Bonnie Lake (**a**); the accumulated cost surface and least-cost path between the two points (**b**)

9.4.2 Interpolation Functions

GIS are often used to interpolate continuous raster surfaces from regularly, irregularly, or even scarcely distributed point data. These surfaces usually represent elevations, temperatures, or other continuous phenomena. Interpolations can also be performed from contour data. Re-sampling to a different resolution can also be seen as a special case of interpolation.

GRASS supports a range of re-sampling and interpolation methods such as nearest neighbour re-sampling, bilinear and bicubic interpolation (*r.resamp.interp*), inverse distance weighted (IDW in *v.surf.idw*, Lo and Yeung 2006; Issaks and Srivastava 1989) and regularized splines with tension (RST) in *v.surf.rst* as well as *v.vol.rst* for volumetric data (Mitasova and Mitas 1993; Mitasova and Hofierka 1993).

Data interpolation in GRASS is explained in detail in Neteler and Mitasova (2008). The RST method especially gives much control over the behaviour of the interpolator through the smoothness and tension parameters. To obtain reasonable results, an integrated cross-validation algorithm helps to find input parameters which minimize the interpolation error. Integrated segmentation allows interpolation of massive data sets. Additional related methods are re-sampling of point data and raster maps using spatial aggregation (*r.in.xyz* and *r.resamp.stats*) and temporal aggregation (*r.series*).

9.4.3 Geomorphological Analysis

Landscape processes, including water and sediment flow, are both caused and influenced by the geometry and properties of the land surface. Basic land features (e.g.

planes, pits, peaks, ridges etc.) can be identified with the module *r.param.scale*. GRASS includes also an extensive set of modules for deriving land-surface parameters and performing spatial analysis that involves elevation data (Hofierka et al. 2008).

Fundamental parameters of the surface can be calculated using partial derivatives of the mathematical function describing the surface as local parameters (based on a point and its immediate surroundings):

- Slope (steepest angle of the slope);
- Aspect (slope orientation, direction of gradient, steepest slope or flow direction);
- Profile curvature (surface curvature in the direction of gradient);
- Tangential curvature (surface curvature in the direction of contour tangent);
- Mean curvature (an average of the two principal curvatures).

Additionally, GRASS provides a wide array of tools to analyze water flow and watershed parameters, starting from basic parameters (flow accumulation, upslope contributing area, stream network, watershed (basin) area, flow path length) to complex algorithms for flow routing:

- Single flow direction to eight neighbouring cells moves flow into a single downslope cell (*r.watershed*);
- Single flow to any direction (D-inf or vector-grid approach) (*r.flow*);
- Multiple flow direction (MFD) to two or more down slope directions (*r.terraflow*, *r.topmodel*);
- 2D water movement simulation based on overland flow differential equations (*r.sim.water*).

Various models and parameters can be combined, and new models can be developed through the use of the map algebra module *r.mapcalc*.

9.5 Vector Applications

The GRASS vector model contains a description of topology. From the user's perspective, topological operations enable verification and enforcement of data integrity and quality. By default, GRASS 6 always builds the topology after importing, creating or modifying a map. A considerable amount of detail about these algorithms can be found in the relevant literature (Lo and Yeung 2006; Neteler and Mitasova 2008)

9.5.1 Overlay Operations and Selections

Geometric operations are performed through specialized modules. By selecting an operator, features from two input maps are geometrically elaborated and the result is written to a new output vector map. A typical set of operators and tools are available:

- *and*: also known as “intersection”;
- *or*: also known as “union”;
- *not*: features from map A not overlaid by features from map B;
- *xor*: features from either map A or map B but not those from A overlaid by B;
- point-in-polygon selection;
- vector extraction (spatially, or based on SQL statements).

9.5.2 Network Analysis and Linear Reference System

The integrated Directed Graph Library (DGL) provides support for vector network analysis. Available algorithms include shortest path, traveling salesman (round trip), allocation of sources (sub-networks), Minimum Steiner Trees (star-like connections), and iso-distances (from centres). Costs may be assigned both to nodes and arcs. Both directions of a vector line can be used, which permits definition of a forward and a backward direction, and storing of their attributes in the related attribute table. An example for shortest path routing on a vector network is shown in Fig. 9.4.

The following modules are available:

- *Shortest-Path Analysis*: the commands *d.path* and *v.net.path* allow calculation of the least expensive (by default the shortest path as the length of the vectors is used as the measure of cost) distance between two chosen points with two methods. Additional information about the vectors (e.g., speed limit on the road or road status) can be used for calculating a path. Cost information can also be assigned differentially to both vector directions. Attributes of the nodes (e.g., cycle times of the traffic lights at a crossroad) can also be considered.
- *Subnets within a vector network*: a given vector network can be subdivided into sub-networks with the module *v.net.alloc*. For instance, regions in a city can be identified that are best served by a limited number of fire stations.

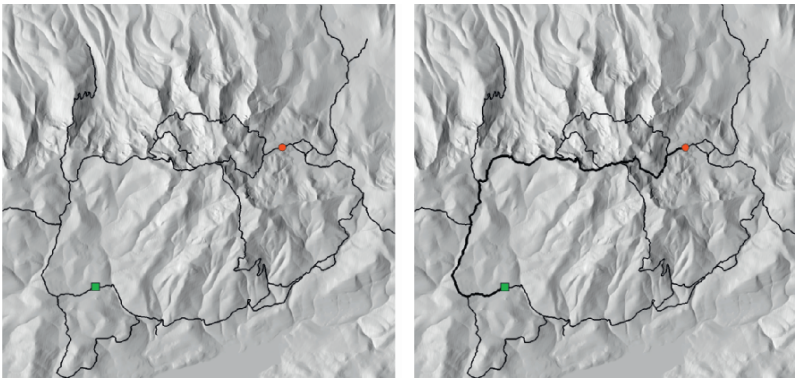

Fig. 9.4 Simple example of a shortest-path calculation from node 1 (*green square*) to node 2 (*red circle*) along a network

- *Minimum Steiner Tree problem*: the optimal connection of nodes within a network (star) can be described by the Minimum Steiner Tree. For instance, several hospitals distributed in a region need new fibre-optical network cables for telemedical services. The aim is to lay the necessary cable along existing roads, while using the least amount of cable to connect all hospitals. The GRASS module *v.net.steiner* is used for this purpose.
- *Traveling-Salesman problem*: the classic problem of calculating the best route joining a series of points (either by distance or time) can be solved with the GRASS module *v.net.salesman*. As an example, a route minimizing cost for a company representative visiting a series of customers can be calculated.
- *Cost analysis*: finding iso-distances (concentric distances around a series of points) on a network can be performed with the GRASS module *v.net.iso*. For instance, "run-length" (e.g. for sewage channel systems) can be calculated based on the vector length or, as in previous modules, other attributes. Another application is the search for reachable points of interest within a five minute walk from a metro-station.

Besides vector networking, GRASS also supports Linear Reference Systems (LRS; Blazek 2005). A LRS is a system where features (points or segments) are localized by a measure along a linear element. The LRS can be used to reference events for any network of linear features, for example roads, railways, rivers, pipelines, electric and telephone lines, water and sewer networks. An event is defined in LRS by a route ID and a segment. A route is a path on the network, usually composed from more features in the input map. Events can be either points or line segments.

9.5.3 LIDAR Data Processing

Airborne Light Detection and Ranging (LIDAR) is one of the most recent technologies in surveying and mapping. A laser on board a plane sends out pulses to the ground in order to determine the distance to an object or surface. This distance is determined by measuring the time delay between transmission of a pulse and detection of the reflected signal. The horizontal and vertical accuracies are in the centimetre range. Up to four range measurements can be performed for each pulse. LIDAR offers various new applications including terrain change analysis to study vertical and horizontal changes in terrain (e.g. beaches) and allowing for calculation of area and volume change. Another application is hazard mapping (e.g. detection of avalanches or other morphological hazards). In the area of energy analysis, solar energy can be estimated at building detail for alternative energy evaluation, since roof inclination and structure details can be derived from LIDAR data. The LIDAR toolset in GRASS provides advanced methods to compute the digital surface (DTM or DSM) based on radial basis functions and spline functions with the Thykhonov regularizer (Brovelli et al. 2004).

The data elaboration procedure is started with the import of LIDAR point clouds (first and last pulse) with *v.in.ascii*. In this case, the topology is not built to avoid redundant computations. Subsequent outlier detection is done with *v.outlier* on both first and last pulse data. At the next step, edges are detected from the last pulse data with *v.lidar.edgedetection*. The DSM (buildings, vegetation etc.) is generated with *v.lidar.growing* from detected edges. The resulting data are post-processed with *v.lidar.correction*. Finally, the DTM and DSM are generated with *v.surf.bspline*.

The *r.in.xyz* module can be used to perform binning of points derived from raw sensor measurements, or coordinates exported from the common American Society for Photogrammetry and Remote Sensing LIDAR Exchange Format (LAS), to grid cells of a given size. This module can use several common statistics such as min, max, mean, and range for cell-wise aggregation. With careful tuning of region settings (extent and resolution), it is possible to generate grid point data where the number of features exceeds the practical limits imposed by the GRASS vector engine. Tests have shown that *r.in.xyz* performs well with over 25 million input features.

9.6 Image Processing

Remote sensing is a rapidly advancing technology for gathering environmental data using a wide range of airborne and satellite platforms, and it plays a major role in spatio-temporal earth surface monitoring. Image data within GRASS are treated identically to raster data. However, several commands are explicitly dedicated to image processing. GRASS supports import of common satellite data and imagery formats, geocoding of imagery data, visualizing (true) colour composites, calculation of vegetation indices, calibration of channels, image classification and image fusion as well as time series processing.

Satellite imagery and orthophotos (aerial photographs) are handled in GRASS as raster maps and specialized tasks are performed using the imagery (i.*) modules. All general operations are handled by the raster modules.

9.6.1 Common Operations

Besides visualization, all common steps of radio and geometric pre- and post-processing are supported. Some of the available techniques are illustrated in the following sub-sections.

9.6.1.1 Visualizing True Colour Composites

The GRASS command *d.rgb* can be used to combine the first three channels quickly to a near natural colour image. The graphical GIS manager (*gis.m*) offers an easy

interface to work with colour composites. The procedure assigns each channel to a colour which is then mixed while displayed. With some user optimization of the grey scales of the channels, nearly perfect natural colours can be achieved. The *i.landsat.rgb* can be used to adjust automatically LANDSAT red, green, and blue channels to closely match the colours perceived by the human eye. Channel histograms can be displayed with *d.histogram*. However, this command operates on any raster map.

9.6.1.2 Calculation of Vegetation Indices

A common proxy for chlorophyll content, utilizing both red (LANDSAT channel 3) and near infrared (LANDSAT channel 4) wavelengths, is the normalized difference vegetation index (NDVI; Jensen 2000). A NDVI map is commonly used to infer water stress in plants, vegetative intensity, or potential crop yields. Calculation of this and other vegetation indices can be done in a single step with simple map algebra, as implemented in the GRASS command *r.mapcalc*:

$$\text{ndvi} = 1.0^{*}(\text{tm4} - \text{tm3})/(\text{tm4} - \text{tm3})$$

where:

ndvi the resulting map,
tm3 and tm4 the LANDSAT channels existing as GRASS raster maps.

The command *r.colors* can be then used to create an optimized colour table. The Kauth-Thomas "Tasseled Cap" transformation for LANDSAT-TM sensor data can be performed directly with the GRASS command *i.tasscap*. A comprehensive list of vegetation indices and their respective formulae can be found in Jensen (2000).

9.6.1.3 LANDSAT Operations

The encoded digital numbers of a thermal infrared channel can be transformed to degrees Celsius (or other temperature units) that represent the temperature of the observed land surface. This requires a few algebraic steps with *r.mapcalc* which are outlined in the literature to apply gain and bias values from the image metadata (Neteler and Mitasova 2008).

A downscaling approach commonly applied to channels 1, 2, 3, 4, 5, and 7 (30 meter resolution) can be performed in GRASS with the Brovey transform (*i.fusion.brovey*) method. This approach *fuses* the panchromatic channel (15 metre resolution) with selected channels to produce a new "pan-sharpened" color composite at 15 metre resolution. Colour composites can be displayed with *d.rgb*, as described earlier, or saved with *r.composite*.

9.6.1.4 Time Series Processing

GRASS also offers support for time series processing (*r.series*). Statistics can be derived from a set of co-registered input maps such as multi-temporal satellite data. Common univariate statistics and linear regressions can be calculated. For example, Rizzoli et al. (2007) used MODIS time series data processed in GRASS to predict the highest risk areas for increased tick-borne encephalitis virus activity in the Province of Trento, Italy.

9.6.1.5 Rectification/Ortho-Rectification

GRASS is able to geocode raster and image data of various types, including unreferenced scanned images of maps by defining four corner points, unreferenced satellite data from optical and Radar sensors by defining a certain number of ground control points (*i.group*, *i.target*, *i.points*, *i.rectify*), and orthophotography based on DEM data (*i.ortho.photo*). Also, digital photographs from handheld cameras can be geocoded using a modified procedure for *i.ortho.photo* (Neteler et al. 2005).

Geo-referencing in GRASS is done by first importing the images into a generic x,y (non-projected) location. Several bands of the same images can be treated together, defining a group. The user is then prompted to identify a series of ground control points. A Root Mean Square is automatically calculated during the process, providing a means to keep track of the overall accuracy while inserting points. Referenced images are then transformed into a new geo-referenced location by polynomial transformation.

9.6.2 Image Classification

A thematic map can be generated from one or more input channels. These input channels are usually derived from aerial or satellite data. Multispectral data can be considered as a stack of raster maps with identical spatial references. During the image classification procedure the spectral response of objects is analyzed and assigned to classes. The resulting map contains a set of classes which may represent land use or land cover. An example of unsupervised classification, performed with an IKONOS multi-band image is depicted in Fig. 9.5. Modules *i.class* and *i.maxlik* were used for this analysis.

GRASS supports multiple channels which can be grouped together with the *i.group* module. Then either an automated statistical analysis can be performed on the input channels (i.e., an unsupervised classification), or training areas can be digitized by the user to define known land use/land cover areas (i.e., a supervised classification). GRASS derives spectral signatures for the desired classes and runs the final analysis on all pixels of all input channels, assigning each pixel to a class. In the case of unsupervised classification, the classes are arbitrarily numbered, while

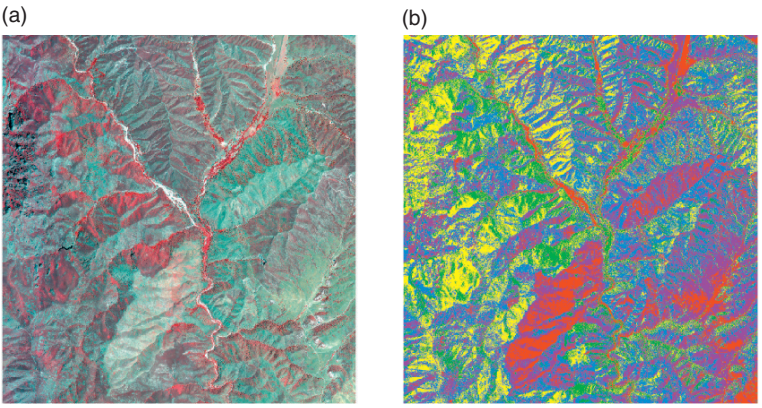

Fig. 9.5 Unsupervised classification of an IKONOS 3-band pseudocolour image (**a**) into 5 discrete classes (**b**)

in the case of supervised classification they correspond to the names of the training areas (Neteler et al. 2005).

Single and multispectral data can also be classified to user-defined land use/land cover classes. In the case of single channel data, segmentation or partitioning of images is used. GRASS supports the following methods:

- Radiometric classification;
- Unsupervised classification (*i.cluster* and *i.maxlik*) using the Maximum Likelihood classification method;
- Supervised classification (*i.gensig* or *i.class*, *i.maxlik*) using the Maximum Likelihood classification method;
- Combined radiometric/geometric (segmentation based) supervised classification (*i.gensigset*, *i.smap*);
- Kappa statistics can be calculated to validate the results with *r.kappa*.

Machine learning, the science of discovering and recognizing patterns in data with algorithms that improve automatically through experience, has been frequently applied to image processing in recent years. Based on samples, a classification or regression function is synthesized in order to predict unknown observations (for example, for a land use/land cover classification). Several machine learning technologies have been implemented in a new set of GRASS modules. The recently added *i.pr* family of image processing modules implement several additional modern classification routines including:

- k-NN (multiclass);
- classification trees (multiclass);
- maximum likelihood (multiclass);
- support vector machines (binary).

Robust estimation of classes is gained with bagging and boosting. Feature normalization within each module allows for the inclusion of predictor variables with

non-normal distributions or different ranges. There is planned support for feature selection techniques such as re-sampling and cross-validation, for all classification algorithms available in *i.pr*.

9.7 GRASS Development

The FOSS4G concept offers a license scheme which defines the extent of the usage, modification, and redistribution of original and derived software. For GIS, unlimited access to the source code is of particular interest, as the underlying algorithms are often complex and have significant influence on the results of spatial analysis and modeling. While an average user may not be able to trace errors within complex source code, there are a number of specialists willing to test, analyze and fix any errors identified within the code. This framework is embedded into an Internet-based development model with a high frequency of new releases. The diversity in background and expertise of the developers contributes greatly to network synergy. Overall, this type of software development model leads to faster and more efficient production, along with stable and robust products.

9.7.1 The Development Model: Community-Based

Access to others who are using GRASS, and to the individuals who are actively maintaining the GRASS code base is critical to the OS development model. Several GRASS-related mailing lists and Internet Relay Chat (IRC) channels serve as windows into the GRASS user base and development communities. Additionally, a bug and wish tracking system is used. New and seasoned users alike can find answers, submit bugs, or even contribute code and documentation suggestions through these resources. This style of development fosters a two-way relationship between users and developers. In many cases users find that over the course of a couple of years they can progress from a simple interested party to a part-time contributor of documentation and source code suggestions. Several of the core GRASS development team have followed this path, and they are now in charge of a large and complex code base.

The development tool set includes a server-based code repository with immediate change notification via email for peer-review of the changes. The GRASS Project Steering Committee (PSC) is responsible for granting write access to new developers. To control and improve the source code quality, an automated code monitoring system was established to apply software engineering metrics and clone detection (i.e., identification of undesired source code copies which are identical or rather identical) on every change happening in the source code repository (Bouktif et al. 2006). This GRASS quality assurance system will be further improved and extended in the future and may potentially be made available to further projects under the auspices of the Open Source Geospatial Foundation (OSGeo) (see Chaps. 1 and 2, among others, of this book).

9.7.2 GRASS Programming

GRASS provides a unique opportunity to improve and extend basic GIS capabilities through new code development and the support of the GRASS developer community. The complete GRASS source code is available on the GRASS Web site referenced earlier. The code base is written in ANSI C programming language and is portable across common operating systems and architectures. To make the development of GIS tools more efficient, GRASS provides a large GIS library with documented application programming interfaces (API) (C and C++). Two programming models exist, namely wrapping the core GRASS modules in user-created scripts (bash shell, Python, TclTk, etc.), or directly interfacing to the GRASS API with code written in C or C++. Most repetitive or iterative tasks can be accomplished with the "script programming" approach, while more advanced users can extend existing code via the GRASS API. Several of the modules now included in the GRASS code were originally developed by users to solve highly specific research-based spatial analysis problems within research projects which were then generalized for common GIS usage.

The modular concept of GRASS is also important for facilitating further development. Most modules are also usable from the command line, which allows their integration into UNIX shell, PERL, PHP or Python scripts. The C API is exposed to other programming languages through a SWIG interface (currently PERL and Python). While GRASS provides only limited support for parallel computing (only the partial differential equations library), it is possible to use it in a simple scripted approach on clusters (Neteler et al. 2005). Significant performance gains can be accomplished through this approach, especially in image processing, where processing of image tiles can be parallelized to take advantage of multiple processor cores.

9.7.3 Room for Development: Where GRASS Needs Work

Despite the rich history and diversity of modules which make up GRASS, users have identified several weak points. Seasoned GRASS users are familiar with difficulties associated with producing publication-ready maps with GRASS, tied to the original focus on analysis by the GRASS development team. Modules such as *d.out.file* and *ps.map* can be used to make simple maps, but external vector or image editing applications are commonly employed to produce a final product (Inkscape, Skencil, etc.). The recent addition of direct PostScript output from the low-level display commands has improved this situation, however a more unified solution will probably not be integrated until the release of GRASS version 7. In the meantime, several people have discovered that publication-quality maps can be created by coupling GRASS to external, specialized applications such as Generic Mapping Tools (GMT) and MapServer. An ongoing project (until 2009) at FBK-IRST/Municipality of Trento (Italy) is aiming to develop new vector editing functionality and a graphical front-end to hard copy map production.

Automatic visualization of thematic vector data (i.e., the display of choropleth maps, with automated colour palette selection) is cited as a key feature of a GIS, and is notably missing from GRASS. Since vector analysis was added to GRASS from version 4.0, there has been a shorter amount of time for developers to implement thematic vector display. It is possible to define manually a colour palette based on individual attributes, and display each category one at a time to "build up" a final image. However, this approach is far from optimal, especially for new users. External interfaces to GRASS data such as QGIS or uDig have thematic vector display capabilities. As these applications continue to mature it is possible that they will fill the role of thematic map creation.

9.8 A Complete Geospatial Toolkit

GRASS contains nearly all of the functionality required for most types of GIS work. However, there are several cases where external applications are better suited for specialized tasks. Detailed statistical analysis, hard copy map production (as discussed earlier), or complex database queries are issues that numerous users have identified. In this context, the existing FOSS4G software stack can fill the gaps (Jolma et al. 2006). Instead of re-inventing the wheel, the GRASS development team has worked hard to leverage external, specialized applications where appropriate. Figure 9.6 illustrates some of the commonly used external tools within the context of what they are used for.

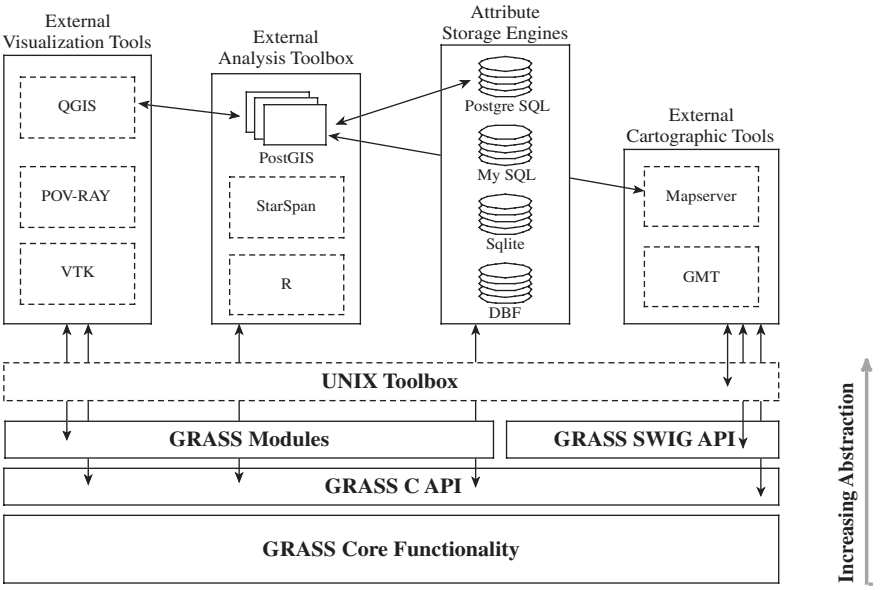

Fig. 9.6 An example of the extended GRASS geospatial toolkit: a set of visualization tools; external analysis toolbox; external attribute data management; online and press-ready map production

These applications, when coupled with the core GRASS modules through both loose (i.e., using intermediate files of commonly structured text or binary data to transfer information between applications) and tight (i.e., using program APIs and inter-process messaging to pass data between applications) coupling, can fulfill even the most demanding spatial analysis needs.

9.8.1 External Attribute Storage

The default storage engine for tabular, or attribute, data is the well-known xBASE (DBF) table format. The GRASS database abstraction routines provide a simple SQL interface to the xBASE files. However, advanced SQL constructs and data types are not supported. These are, however, available through the use of industry-strength RDBMS such as SQLite, PostgreSQL, or MySQL as the repository for attribute data. GRASS provides "tight integration" to these applications, allowing the user to switch freely between RDBMS backends with the *db.connect* module.

9.8.2 Graphical User Interfaces

Quantum GIS (QGIS, http://qgis.org) has emerged as a favourite FOSS4G 2D visualization environment for GRASS users, providing a modern user interface and map element symbology editor. The current version of QGIS contains a graphical interface to most GRASS tools, a graphical data catalogue, and a native vector digitizer. Javagrass (http://www.jgrass.org), a client-server implementation, is an alternative user interface which includes 3D visualization. It comes with special focus on hydrological and geomophological analysis. The fusion with the uDig software project is currently ongoing, adding 3D visualization and further GIS analytical capabilities to uDig.

9.8.3 Visualization

The built-in visualization module NVIZ is very efficient and powerful. It supports raster and vector data including 3D and raster volumes. Flight-simulator mode, high-resolution output, and custom animations complete the suite.

The Persistance of Vision Raytracer (POV-Ray, http://www.povray.org/) has been used for over a decade by professional graphic designers to render complex 3D scenes, complete with realistic lighting and surface interaction. The modules *r.out.pov* and *v.out.pov* convert GRASS raster and vector data types into a format suitable for inclusion in a POV-Ray scene file (Fig. 9.7(a)).

The Visualization Toolkit (VTK) has recently emerged as an exciting new way to interact with GRASS raster, vector, and volume datasets. Paraview, an integrated

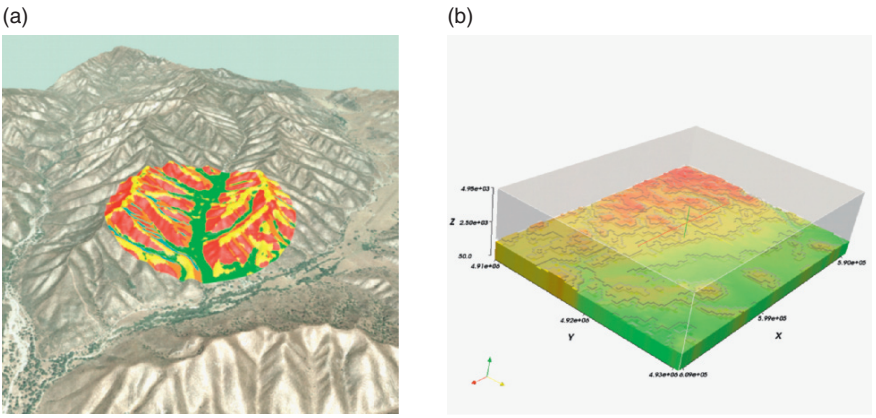

Fig. 9.7 Examples of external applications, POV-Ray (**a**) and Paraview (**b**), commonly used to visualize complex 2, 3, and 4 dimensional GRASS datasets

visualization application based on VTK, is a modern analogue to NVIZ and in its latest version is implemented with the cross-platform Qt GUI toolkit. The modules *v.out.vtk*, *r.out.vtk*, and *r3.out.vtk* are used to export GRASS raster, vector, and volume datasets (Fig. 9.7(b)).

9.8.4 Statistical Analysis

While several modules within the core GRASS system can perform simple statistical summaries or compute correlation coefficients, R (R Development Core Team 2006), the OS implementation of the "S" statistical language, is the FOSS platform of choice for detailed statistical analysis. R has a long history of compatibility with GRASS, and connecting the two is as simple as installing the necessary packages in the R environment. The interface between GRASS and R has been under rapid development over the last two years, and the current implementation has simplified the process significantly (Bivand 2007). R software is hosted at SourceForge (http://r-spatial.sourceforge.net), and includes *spgrass6* (the main R-GRASS interface), *sp* (the basic spatial object classes), *spGDAL* (a wrapper for functions in the *rgdal* package which interfaces to the GDAL library – see Chap. 5), and *spmaptools* (an interface to the *shapelib* library – see Chap. 5).

Regional or "zonal" statistics can be calculated within GRASS using the *v.rast.stats* script, however Starspan (http://starspan.casil.ucdavis.edu/), a more mature implementation of these functions written in C++, can be used instead. Starspan uses the GDAL library to read natively GRASS raster and vector data for the calculation of zonal statistics. Results are saved in comma separated values (CSV) format, which can easily be read into R for statistical analysis or back into GRASS.

Several examples of descriptive, geo-statistical, and other standard functions in R can be found in the GRASS Wiki site (http://grass.osgeo.org/wiki, see also Figs. 9.8

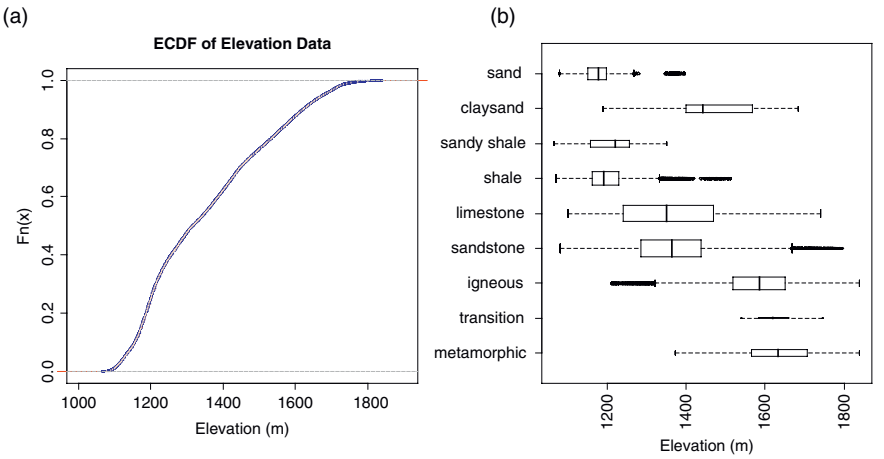

Fig. 9.8 Simple exploratory statistics performed in R on elevation (**a**) and geology (**b**) raster data

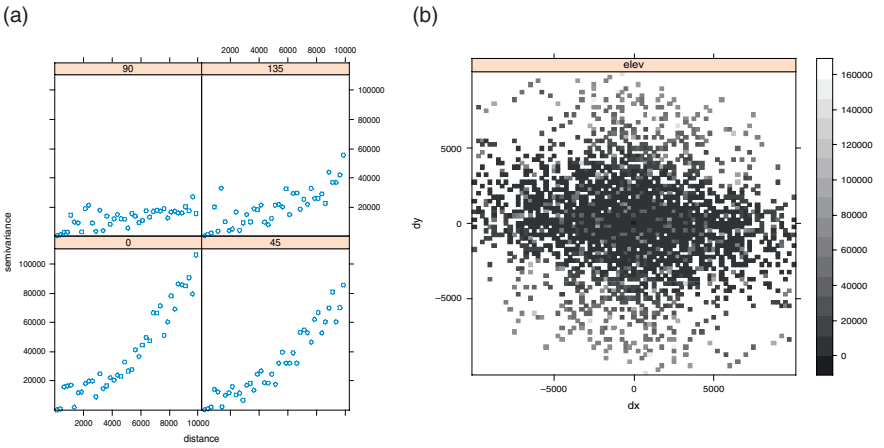

Fig. 9.9 An exploration of anisotropy with directional variograms (**a**) and a variogram map (**b**)

and 9.9). The examples used here are based on the "Spearfish" sample data location (South Dakota, USA, 103.86W, 44.49N), the classical GRASS sample data set.

9.8.5 Cartographic Tools

As noted earlier, simple maps can be printed with the help of QGIS as an interface, while several other options are available for more complex tasks. MapServer (Lime 1996 – see Chap. 4 of this book) is a common gateway interface (CGI) application targeted at the production of online maps. It functions as the primary

cartographic engine behind many popular online mapping applications. MapServer can directly read GRASS raster and vector data through the GDAL library, making it possible to serve maps online immediately after they are produced without the need for intermediate conversion. The expressive map description language used by to define symbology, coupled with features such as automated label placement and collision detection make MapServer a powerful map making tool.

The GMT suite (http://gmt.soest.hawaii.edu/) consists of 60 specialized map making commands which output to publication-quality PostScript format. With GMT in hand, it is possible to fill the gap in hard copy map production which has troubled GRASS users for years. Several attempts have been made at integrating GRASS and GMT (Beaudette 2007). However, most are based on the "loosely coupled" approach of using intermediate files. Recent developments in the GDAL library, maturation of the Python/SWIG API, and planned Python integration in GMT 5.0 suggest that a more generalized and coherent fusion of GRASS and GMT will be possible in the near future.

9.9 The GRASS is Growing: Reaching the World

The source code accessibility and the modular character of GRASS render the software an ideal platform for further development. The core system with its analytical capabilities can be accessed externally using several programmable approaches in order to build graphical user interfaces upon it, or to use GRASS as the backbone for external applications.

9.9.1 Making GRASS Accessible with a GUI

In place of a monolithic graphic interface to the GRASS modules, several small GUI systems have emerged over the years. Initially based on the TclTk GUI framework *tcltkgrass*, *d.m* and its successor *gis.m* have provided a simple interface to common GIS analysis and visualization. These modules use a simple palette construct for defining a list of commands like *d.rast* or *d.vect*, with common GUI constructs such as buttons, check boxes, and forms for defining style elements. The GUI for NVIZ (the 3D interface to GRASS raster, vector, and volumetric data) and *v.digit* (the GRASS vector digitizer tool) were also built with the TclTk framework. TclTk has long been used as a robust, multi-platform GUI toolbox. However, it is starting to show its age.

The new GUI, *wxGRASS*, provides a modern-looking, compact layout of the core GRASS functionality along with operating system-native graphical elements (also known as widgets). Python/wxPython provides numerous GUI primitives (such as color pickers, combo box menus, etc.) which simplify the underlying code, and enable users to adjust display properties quickly. In addition, the conversion from

TclTK to the popular Python language opens development to a wide audience of Python programmers. Object-oriented design, simple extensions of functionality though loadable modules, and advanced array handling are just some of the features Python has to offer. The *wxGRASS* interface contains several new and updated features. A built-in attribute management system with query support simplifies table manipulation operations. Improved type rendering and printing support also facilitate rapid map production.

9.9.2 SWIG/Python Interface

The Simplified Wrapper and Interface Generator (SWIG-http://www.swig.org/) is a framework for connecting internal functions (i.e. GRASS C and C++ API) to an assortment of popular scripting languages such as TclTk, MATLAB, Perl, Python, PHP, JAVA etc. The SWIG compiler reduces the tedium of writing language-specific extension
modules by automating the entire process. A functional prototype GRASS-SWIG interface was recently implemented. The GRASS-SWIG interface opens the complex GRASS core-API functionality to developers who may be more proficient in, or prefer the flexibility of, different scripting languages. In addition, as the number of SWIG-enabled projects increases it will become possible to create a custom meta-API (in Python, for example) which bridges multiple distinct applications. More information on the GRASS-SWIG interface can be found on the GRASS Wiki.

9.9.3 GRASS: A GIS Backbone for Web Applications

Web-based spatial services are developed for many applications. In particular, opportunities for integrating data from various Spatial Data Infrastructure (SDI) portals into Internet-based services have grown. Interoperability and standardization are accomplished by following standards such as those set by the OGC for spatial data and related information technologies. While internally data storage and processing may differ from proposed OGC standards, all recent FOSS4G systems provide data exchange interfaces consistent with industrial standards. FOSS4G applications can either directly read GRASS raster and vector data through GDAL/OGR interoperability, or, GRASS vector data can be stored in a PostGIS-compliant spatial database, which is then linked to the application.

The Python Web Processing Service (PyWPS, http://pywps.wald.intevation.org) is a relatively new project started in spring 2006, which aims at implementing the OGC Web Processing Service (WPS) standard. A WPS can be configured to offer any sort of GIS functionality to clients across a network, including access to pre-programmed calculations and/or computational models that operate on spatially

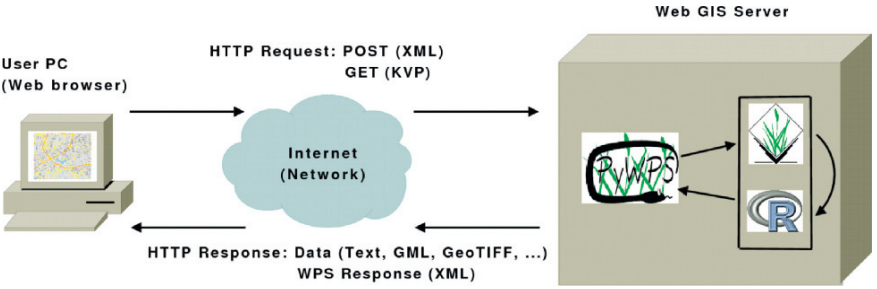

Fig. 9.10 Scheme of PyWPS (Python Web Processing Service)

referenced data. A WPS may offer calculations as simple as subtracting one set of spatially referenced numbers from another (e.g., determining the difference in influenza cases between two different seasons), or as complicated as a global climate change model. The data required by the WPS can be delivered across a network, or be available at the server.

PyWPS acts as a as translator between requests from a client and the working tool (GRASS), installed on the server. In this way, the capabilities of GRASS are extended from a simple desktop application to a complete network-oriented system. Figure 9.10 illustrates the conceptual scheme of this approach.

The WPS standard is similar to other better known OGC standards, for example the Web Mapping Service (WMS) or Web Feature Service (WCS). It accepts three types of requests, namely *GetCapabilities*, *DescribeProcess* and *Execute*. Each server offers several processes (tasks), which it is able to provide. Each process has defined inputs and outputs. Data inputs and outputs are *ComplexValue* (raster or vector maps or reference to them). Other inputs can be of type *literal value* or *bounding box value*. With GRASS, all possible tasks can be performed, which do not require direct interaction with the map display. With the combination of WPS-clients (which can be a Web application in a Web browser, or plugin to a desktop GIS), a user does not have to install large GIS packages on his/her desktop as the application is running on a remote server.

9.10 The Future: GRASS in the FOSS4G Arena

Ongoing development will lead to GRASS Version 7 which will address several important issues. The GRASS raster libraries need an overhaul in order to take advantage of modern storage mechanisms such as tiling and caching. In preparation for new (computational) challenges, parallelization of numerical operations will be extended in order to support better calculations on computer grids and distributed systems. Virtual linking of raster and vector data sources is planned, which will reduce the number of import operations for most projects. Improvements in the image processing library are directly linked to changes in the raster library. However, linking

development to the Open Source Security Information Management (OSSIM) API might be a sensible way to avoid parallel development. Along with these changes, better support will be added for time series in GRASS, based on SQL constructs.

The vector processing modules are in need of transaction support. In particular, a technology is required which would preserve geometry in the event of an incomplete or otherwise unexpected termination of an editing session. Furthermore, better use of spatial indexing should reduce the computational overhead required to build or maintain topology during and after geometric operations. The net result of most of the planned changes will result in better response times, and will enable GRASS to serve better as an analytical backbone for Web GIS and Web Processing Services.

The 2007 and 2008 Google Summer of Code (SoC) program has sponsored several GRASS-related projects including improved line generalization algorithms, least-cost path calculation based on global minima, a new module to compute least-cost paths in vector space (as opposed to the current raster-based approach) and others. These projects address several long standing issues which have been raised on the GRASS mailing list and feature-request system. The SoC program will be an important developer recruitment facilitator, and future projects are already being planned.

From the end user's perspective, GRASS is one of the few complete analytical FOSS4G projects available. It has evolved from its humble origins as a land management tool for military installations into one of the most comprehensive, general purpose GIS available. Support for environmental applications has been an integral part of its twenty plus years of development. Environmental and social data from various sources can be easily integrated into a GRASS project and this makes the software an attractive option to satisfy a wide range of application needs.

References

Beaudette DE (2007) Producing press-ready maps with GRASS and GMT. J Open Source Geospatial Foundation 1:29–35

Bivand R (2007) Using the R-GRASS interface. J Open Source Geospatial Foundation 1:36–38

Blazek R (2005) Introducing the linear reference system in GRASS. Int J Geoinformatics 1(3):95–100

Bouktif S, Antoniol G, Merlo E, Neteler M (2006) A novel approach to optimize clone refactoring activity. In GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM Press, New York, USA, pp 1885–1892

Brovelli M, Cannata M, Longoni U (2004) LIDAR data filtering and DTM interpolation within GRASS. Trans GIS 8(2):155–174

Hofierka J, Mitasova H, Neteler M (2008) Terrain parameterization in GRASS. In: Hengl T, Reuter H (eds) Geomorphometry: concepts, software, applications, Developments in Soil Science, Vol. 33, Elsevier, Amsterdam pp. 387–410.

Issaks EH, Srivastava RM (1989) An introduction to applied geostatistics. Oxford University Press, England

Jenson JR (2000) Remote sensing of the environment: and earth science perspective. Prentice Hall, New Jersey

Jolma A, Ames D, Horning N, Neteler M, Racicot A, Sutton T (2006) Free and open source geospatial tools for environmental modeling and management. In: Voinov A (ed) Proc. iEMSs 2006, Session W13, July 9–13, 2006, Burlington, Vermont, USA

Lime S (1996) UMN MapServer, University of Minnesota, USA, [Computer Program]. Available: http://mapserver.gis.umn.edu

Lo CP, Yeung AKW (2006) Concepts and techniques of geographic information systems. Prentice Hall, New Jersey

Mitasova H, Hofierka J (1993) Interpolation by regularized spline with tension: II. Application to terrain modeling and surface geometry analysis. Math Geol 25(6):657–669

Mitasova H, Mitas L (1993) Interpolation by regularized spline with tension: I. Theory and implementation. Math Geol 25(6):641–655

Neteler M, Grasso D, Michelazzi I, Miori L, Merler S, Furlanello C (2005) An integrated toolbox for image registration, fusion and classification. Int J Geoinformatics 1:51–61

Neteler M, Mitasova H (2008) Open source GIS: A GRASS GIS approach. 3 edn. Springer, New York

R Development Core Team (2006) R: A language and environment for statistical computing. R foundation for statistical computing, Vienna, Austria. ISBN 3-900051-07-0

Rizzoli A, Neteler M, Rosà R, Versini W, Cristofolini A, Bregoli M, Buckley A, Gould E (2007) Early detection of TBEv spatial distribution and activity in the Province of Trento assessed using serological and remotely-sensed climatic data. Geospatial Health 1(2):169–176