GTZ
GIS
GRASS

INSTEAD OF A FOREWORD

This GRASS handbook is compiled on behalf of the Gesellschaft für Technische Zusammenarbeit (GTZ), the german governmentally owned agency resposible for giving technical aid to developing countries. They financed a project that included the evaluation of a number of geographical information systems (GIS) for their applicability in projects of rural development. As for GRASS, one result of this research was the requirement a) for a handbook and b) for a portation to a public domain PC-based system.

This is what you could call an α-version of the handbook. It certainly needs to be tested in the field. For a number of reasons, a project like this is a never-ending process:

- the development of new GRASS programs gains pace faster than ever
- chapters on XGRASS and external programs like *mapgen*, *LT+*, etc. need to be added
- experiences from **you**, the reader will help to enhance the quality of this handbook

Depending on field experiences, version 1.0 of this handbook will be published at some time during the second half of 1993. By that time, it should include chapters on XGRASS and additional information on the LinuX version (the above mentioned port to a public domain PC-based Unix). Besides error corrections, I especially invite suggestions concerning the proceeding of the chapters. While readers with experience in GIS may skip chapter I, I urgently advise everybody, who is not familiar with GRASS, to study chapters II and III. Together with the following two pages, they build the foundation for all work with GRASS. Chapter VI (Spatial Analysis with GRASS II) is sort of special, in as it consists mainly of shell script listings. They are nice examples of what might be done with(in) GRASS, once you have tackled the beginnings.

Words and phrases in *italics* refer to terms described in the glossary of chapter XIII or are GRASS programs.

This handbook can not (!) replace the manual pages for GRASS, nor the tutorials that exist for special applications of GRASS or related programs. Instead it is conceived as a pretty wholesome introduction to GRASS that bears in mind, that projects of the GTZ usually do not have the support of an academic environment with its campus computer facilities.

THANKS:

First of all, I have to thank Willi Zimmermann from the GTZ, who organized the above mentioned project and without whose support this handbook would never have been initiated. Most important, he raised the money to finance this project.

Then there are all the people in the GRASS community, who either through their participation on the grass-u user list or at GRASS Users' conferences communicated a lot of the information that can be found on the following pages.

Finally, this work depended a lot on manuals, written by the staff at CERL (the new hyper text system in GRASS4.1 was written in parallel to this handbook) and Fred Limps's lecture notes for his courses given at the Center for Advanced Spatial Technologies (Univ. of Arkansas).

WHAT IS GRASS?

The name GRASS stands for 'Geographical Ressources Analysis Support System'. Although often called a geographical information system, it rather is a workbench of tools for the analysis of geographic phenomena. The reason for this differentiation lies in the intention of its original authors to provide a really open system.

This system consists of more than 200 user programs comprised of about 4000 software files made of more than 300,000 lines of C program code. GRASS in itself is not a complete GIS though. It depends on a number of other public domain packages like MAPGEN, GTCP, pbmplus, etc. for cartographic output, coordinate transformation, hardcopy output device drivers and other purposes.

GRASS operates under the UNIX operating system. Users therefore have access to the UNIX commands on their systems, as well as to GRASS commands. UNIX commands can be run at the system level (i.e., outside of GRASS) and can also be run while the user is running GRASS. This capability is extremely useful.

The GRASS programs comprise different layers of routines, reflecting the history of the system. The core consists of some 140 programs that were all either written by the original authors at U.S. Army Construction Engineering Research Lab (CERL) or contributed by others and certified through a long process of validation. These are supported by 40 programs which were developed by other agencies or individuals. They are called to be in α-stage, known to run well but aren't part of the official GRASS release (yet). Then there are those relatively new programs that are contributed from outside and finally a range of common UNIX programs known as the GRASS garden. The latter are independent utilities that come handy because the existence of these programs allows developers of GRASS to concentrate on those parts of a GIS that can not be achieved by other UNIX programs.

GRASS is not conceivable without its community of users. The probably largest single leap in its development was the decision of the US Soil Conservation Service (SCS) to put a GRASS system in every single of its some 2000 county offices. Other large bodies of users can be found in the U.S. National Park Service, NASA, NOAA, U.S. Geological Survey, the U.S. Census Bureau and an increasing number of universities world-wide. Meanwhile, the bulk of development for GRASS is done at the SCS, USGS, and universities like Purdue, Arkansas, Wyoming and Illinois.

GRASS' biggest advantage is - up to now - also it's greatest handicap. The open character of its only loosely coupled programs allow to use this system like a programming language rather than a usual GIS. This workbench character, however, makes it difficult for a novel user (and non-programmer) to gain some insight. A couple of companies (Osiris, X, Y) took this opportunity and started in 1992 to offer X-Windows application generators that can be accustomed to certain user communities who hence don't have to learn GRASS anymore.

The GRASS project initiated by the german GTZ GmbH, Eschborn, is an endeavour to develop a scalable shell for GRASS that starts with a high degree of abstraction from the basic GRASS programs. Using this shell, applications can be developed for local users who have limited analysis demands that are called upon repetitively. They need never to be bothered with the scope and strength of the full-blown general purpose system (an approach first pursued by the SCS in its county offices). Through several steps of course work and practice the user will

learn more about the underlying programs, leave the restrictions of the X-Windows based shell and eventually write own macros and subroutines.

At any level of experience a connection to the world-wide Internet would be of great advantage. There are lists especially for GRASS users and programers, each. New programs are added each week, and GRASS being a workbench, starts to develop numerous sibblings like a Hydrologic Modeling System, a Forest Management System, a Habitat Modeling System or an Archeological Sites Management System.

To the date of writing of this document, there are two aspects of GRASS that many users (and programers) are missing, and that therefore are the main target for future development. GRASS has no explicit support for any database management system (other than RIM, which is rather awkward) and it deals with integer values only. With the release of GRASS4.2 in 1994, this situation is supposed to change.

Future developments include image segmentation, visualization, parallel processing, more vector functions and the inclusion of the LT+ package from SCS. The latter allows for automated (on-screen) digitizing, contouring and raster editing. By the end of this decade, GRASS will change it's workbench character to become a framework. By then this Post-GRASS system is supposed to use terrabyte object-oriented DBMSs, distributed over a network of (parallel-processing) workstations, capable of modeling finite-element simulations.

CONFIGURATION

GRASS runs on just about any hardware that supports some flavor of the UNIX Operating System. Validated version of GRASS are known for Sun, Concurrent, Intergraph, Silicon Graphics, DEC, Aviion, HP (Apollo), RS-6000, AT&T 3B2, Tektronix, PC 386 and Apple MacIntosh. The minimum hardware requirements for the software to run on depend on the scope of applications that GRASS is supposed to be used for. While some basic work can be done with 8 MByte of RAM and a 100 MByte harddisk, the usual GRASS workstation will have 16+ MByte of RAM and a 240 MByte harddisk in a network or 700+ MByte as a standalone system. A monitor capable of 8 bit graphics is as necessary as a mouse; a printer is desirable, whereas a digitizing tablet and plotter are optional items.

Currently supported digitizers are : Altek, Calcomp, Geographics, Hitachi, Kurta, Numonics
Currently supported plotters include : Diabolo CL50, EPSON LQ-2500, Genicom 3310, Shinko 635, Tektronix 4695/97, Versatec and Xerox

GRASS is intimately tied to UNIX and its graphics are handled through X-Windows (although some other graphics environments are supported, too). As on any UNIX system, file I/O via tape or CD-ROM is handled by the operating system, as is the case with networking. If the user works with a standalone system, some familiarity with UNIX is inevitable. In a networking environment, the system administrator should provide you with some information on how to access distributed file systems and use the mailing facilities.

Although GRASS is known to run with numerous window managers, the user should try to stick either to Sun's OpenWindows or the Motif Window Manager. The latter is an absolute necessity if you want to use the XGRASS shell.

Rather unusual for users accustomed to the world of micro-computers is the usage of the keyboard. <Ctrl>-C is the standard key for cancelling commands, whereas the <Esc> key is used for confirmation. Within menus, the <Enter> key is used to move to the next menu item and <Space> returns to the next higher level of the menu.

CONTENTS

Instead of a Foreword..... I
Thanks..... I
What is GRASS ?..... II
Configuration..... III
Contents..... V
List of Tables..... VII
List of Figures..... VIII

I. GIS Intro..... 1
I.1 Basic Map Components..... 1
I.1.1 Vector Data in GRASS..... 2
I.1.2 Raster Data in GRASS..... 4
I.1.3 Vector - Raster Conversion..... 5
I.1.4 Site Data in GRASS..... 6
I.2 Hardware..... 6
I.3 GIS Function: Data Analysis..... 7
I.4 GIS Function: Data Manipulation..... 8
I.5 GIS History..... 9

II Setting up a GRASS Database..... 11
II.1 GRASS Directory Tree..... 11
II.1.1 GRASS Raster Data Directories..... 12
II.1.2 GRASS Vector Data Directories..... 12
II.1.3 GRASS Site Data Directories..... 12
II.2 MAPSET Files..... 13
II.3 Setting up a GRASS Database: LOCATIONs and MAPSETs..... 15
II.4 Coordinate Systems and Porjections..... 17
II.5 Resolution..... 18
II.6 How to Find Out About File Contents..... 18
II.7 General File Management Commands..... 29
II.8 Application: Development of a GRASS Database..... 21
II.8.1 Data Development From Point Data Sources 21
II.8.3 Data Development From Raster Data Sources..... 23
II.8.4 Database Quality..... 24

III Visualization..... 29
III.1 GRASS Graphics Display Commands..... 29
III.1.1 Management of the Graphics Display Monitor..... 29
III.1.2 Management of Display Frames on the Graphics Display Monitor..... 30
III.1.3 Windowing..... 31
III.1.4 Map Display Reporting Functions..... 34
III.1.5 Graphical Manipulation of Map Contents..... 35
III.1.5.1 Color Modification of Screen Graphics..... 36
III.1.5.2 Display of Raster, Vector and Site Data..... 36
III.1.5.3 Visual Query of Map Contents..... 36
III.1.5.4 Creating on-Screen Maps..... 37
II.1.5.5 Graphical Manipulation of Raster Data..... 38
III.2 Hardcopy Map Production Modules..... 39

IV Digitizing and Other Forms of Data Input into GRASS..... 41
IV.1 Digitizing..... 41
IV.1.1 Preparing a Map for Digitizing..... 41
IV.1.2 Thresholds..... 43
IV.1.2.1 The Digitizing Threshold..... 43
IV.1.2.2 The Map Threshold..... 44
IV.1.2.3 The Snapping Threshold..... 44
IV.1.2.4 The Snapping of Nodes..... 44
IV.1.2.5 Setting the Snapping Threshold Value..... 45
IV.2 Data Conversion in GRASS..... 45
IV.2.1 Moving Data to Another Computer (ASCII/Binary Conversion)..... 46
IV.2.2 Raster / Vector / Site Conversions..... 46
IV.2.3 GRASS to/from other File Format Conversions..... 47
IV.3 Extracting Data from Magnetic Tapes..... 48
IV.3.1 Imagery Data..... 48
IV.3.1.1 Data Corrections, Map Projections and Data Resampling..... 49
IV.3.1.2 Technical Intricasies..... 50

V Analyzing Data in GRASS..... 51
V.1 Overview of the Analyzing Capabilities of GRASS..... 51
V.1.1 Analyzing Site Data..... 51
V.1.2 Analyzing Vector Data..... 51
V.1.3 Analyzing Raster Data..... 51
V.1.3.1 2-D and 3-D Manipulations..... 52
V.1.3.2 Single Cell Map Reporting Functions..... 52
V.1.3.3 Multi-Cell Coincidence Tabulations and Ccomparisons..... 52
V.1.3.4 Neighborhood Analysis and Filters..... 52
V.1.3.5 Regional Analysis and Filters..... 53
V.1.3.6 Proximity Analysis..... 53
V.1.3.7 Boolean Overlay Functions..... 53
V.1.3.8 Terrain Analysis..... 54
V.2 Macro Development..... 55
V.2.1 Basic Data Manipulation Functions..... 55
V.2.1.1 *r.reclass*..... 55
V.2.1.2 *r.rescale*..... 56
V.2.1.3 Computing Map Statistics - Mean, Variance, Standard Deviation of Map Category Areas..... 56
V.2.1.4 Computing Statistics on Map Category Values..... 57
V.2.1.5 Computing Spatial Statistical Surfaces..... 57
V.2.2 Advanced Data Manipulation Functions..... 59
V.2.2.1 Map Overlay with *r.weight*..... 59
V.2.2.2 Sample Script to Eliminate Clumps Below a Specified Size..... 60
V.2.2.3 Sample Script to Assign Category Values to Clumps Which Reflect Their Size Order (Smallest is 1, Largest is .. Max Number)..... 61
V.2.2.3 Spatial Sampling..... 62
V.2.2.5 Least Cost Paths, Modeling Alternatives..... 63
V.2.2.6 Identifying Spatial Discontinuities..... 63
V.2.2.7 Dynamic Models (Cellular Automats)..... 65
V.2.2.8 Viewshed Modeling..... 66
V.3 MAPCALC Intro..... 66
V.3.1.1 Map Recoding and Selections..... 67
V.3.1.2 Slope and Aspect..... 67

V.3.1.3 Hydrologic Simulation..... 67
V.3.1.4 Routing and Network Analysis..... 68
V.3.2.1 Spectral Ratios..... 70
V.3.2.2 Merging Panchromatic Data with Multispectral Data..... 71

VI Image Processing..... 73
VI.1 Tape Extraction..... 73
VI.2 Management of Image Files..... 73
VI.3 Image Classification..... 74
VI.4 Geometric Rectification..... 75
VI.5 Radiometric Correction..... 75
VI.6 Image Enhancement..... 76

VII Report Generation..... 77
VII.1 GRASS Commands for Report Generation..... 77
VII.2 Customizing Reports..... 77

VIII Interfaces between GRASS and Other Software..... 79

Appendix A: Sources of Digital Data..... 81

Appendix B: Digitizer Example..... 87

Glossary..... 89

Bibliography..... 97

List of Tables

1 History File Formats for GRASS Map Layers..... 26
2 Overview to GRASS Programs for Display, Input and Screen Capture..... 35

LIST OF FIGURES

		Seite
1	Basic Map Components	1
2	GIS Data	2
3	Vector Data	2
4	Arc Node Structure for Linear Data Types (With and Without Attributing)	3
5	Raster Data Structure	4
6	Polygon To Cell Conversion	5
7	Conversion from Vector to Cell	5
8	Conversion of Linear Structure to a Cell Form (Large Cells)	5
9	Equipment Required to Run a GIS	6
10	Vector (Polygon) Overlay and Dissolve Functions	7
11	Digital Terrain Analysis	7
12	Analytical Functions	7
13	Changing Geometric Properties	8
14	Converting Cells to Vectors and Vice Versa	8
15	Map Generalizations	8
16	The GRASS Directory Tree	11
17	Environment Variables for GRASS Directories	13
18	GRASS Entry Screen (Example 1)	14
19	GRASS Entry Screen (Example 2)	15
20	Creating a New Location	16
21	Defining a New Geographic Region	17
22	Raster Map Layer Resolution (Example)	18
23	Exiting GRASS	20
24	Workflow With Point Source Data	21
25	Workflow With Vector Source Data	22
26	Workflow With Raster Source Data	23
27	Sample Output of s.support	24
28	Digitizing Error (slivers)	24
29	Usage of *r.patch*	25
30	Screen Components (within *d.display*)	30
31	The Initial Display Screen of *d.display*	31
32	Display Windowing	32
33	Creating Screen Frames	33
34	*d.display* Regions	34
35	Usage of *d.zoom* to Modify the Geographic Region Boundaries	34
36	Example of *d.legend*	37
37	Examples of *d.3d*	38
38	Comparison of a Vector Map with its Raster-Converted Equivalent	47
39	*r.weight* (Example)	59
40	Comparison of Map Layers Before and After Running a Clump Script	61
41	Comparison of the Original Elevation Map With a Map of Local Deviation	64
42	Basic Structure of a Dynamic Model	61
43	Expanding the Basic Model	66
44	MapCalc Example for Detection of an Intersection	68
45	The 47 Possible Network Conditions After TOMLIN	69
46	MapCalc Example for Detection of a Vertical Line	70

I. GIS INTRO (TO GRASS GIS)

I.1 BASIC MAP COMPONENTS

Spatial data are objects that have dimensions or physical size. Objects that have dimensions in relation to the earth's surface are called geographic spatial data. They have:

- a *location* relative to some coordinate system
- attributes that are unrelated to their position
- some sort of spatial interrelations (*topology*)

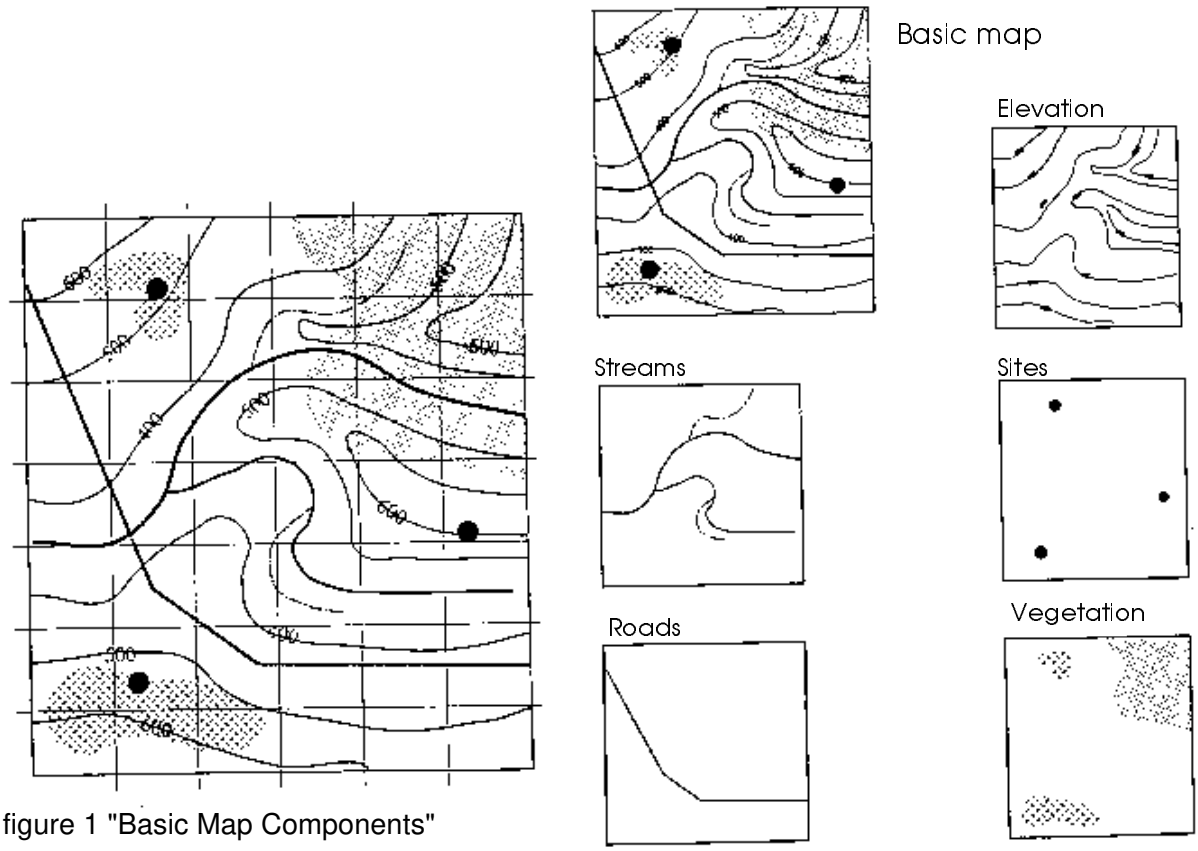


figure 1 "Basic Map Components"

Maps consist of items that show us, how these items are spatially distributed. A base map usually consists of elevation information (both contour lines and single peak heights), streams, roads, individial locations (*sites*) and areal information such as vegetation cover. On a simple map this information has qualitative character only (simply stating that something is there or is not), whereas on complex maps some attributive characteristics are given as well.

In a *GIS* data is separated into spatial / geometric information and tabular resource information as it is usually provided by a *database*. In general, most systems store their data either in *vector* or *raster format*.

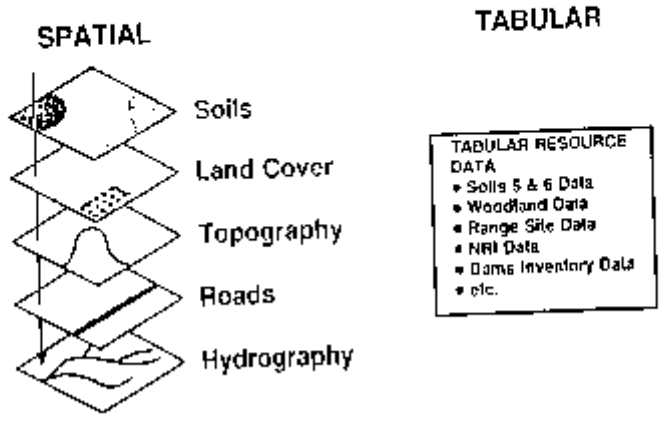


figure 2: *GIS* Data

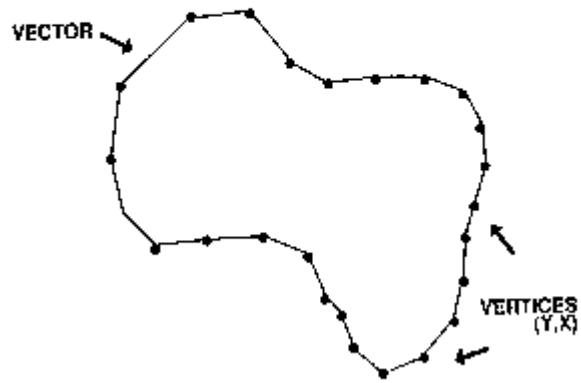
I.1.1 VECTOR DATA IN GRASS

In GRASS *vector format*, area, and linear features are represented as the series of *arcs*. An *arc* (also known as *vector*) is stored as a series of x,y coordinate pairs. The two end-points of an *arc* are called *nodes*. Two consecutive x,y pairs define an *arc* segment (also known as *line* segment).

Arcs, either singly or in combination with other *arcs*, form higher-level *map features* like linear features (e.g., roads or streams) and *area* features (e.g., soil types or timber cutting units). *Arcs* that form linear features are sometimes called *lines*, and *arcs* that outline areas are called *area* edges or *area* lines. Areas are also called *polygons*. In GRASS *vector format*, polygons are not stored explicitly; they are reconstructed by finding the series of *arcs* that form their perimeters.

A GRASS *vector data file* is generally associated with other GRASS *supporting files* describing *category* values, *category* labels, and *file topology*. Together, these files constitute a *vector map layer*. Read the information in chapter II, if you wish further information on the *location* and content of *vector data* and *supporting files*.

Typically, data are *digitized*, imported, and exported in *vector format* and are analyzed in their *raster format*. Data can be displayed and output in either format.



In the *vector format*, a *point* is represented by a pair of X,Y coordinates, a *line* is represented by a series of straight *line* segments, and an *area* is represented by a series of connecting lines. This format allows for a very compact data storage while representing an object as exactly as it is possible to measure.

figure 3: *Vector data*

Examples:

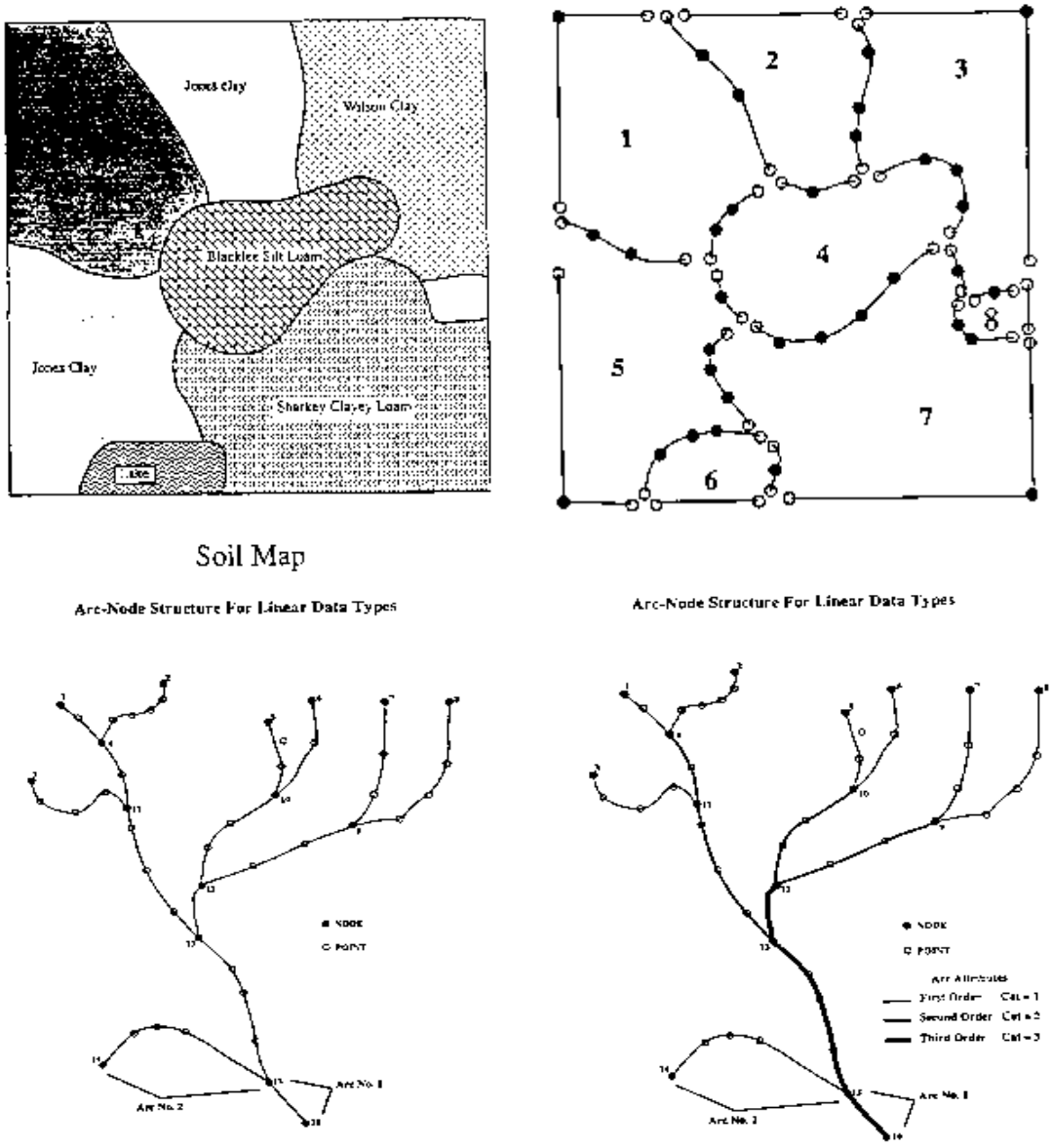


figure 4: *Arc-node* Structure for Linear Data Types (With and Without Wttributing)

Vector format is appropiate for areas with clear boundaries such as property ownerships, road limits, waterline locations, electrical systems and most urban types of data.

I.1.2 RASTER DATA (IN GRASS GIS)

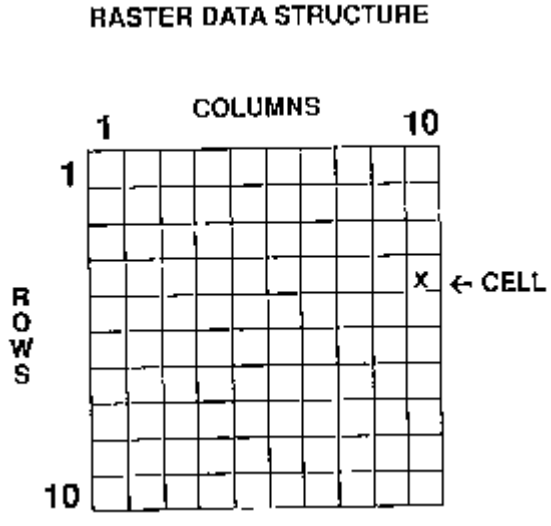
Raster data format uses the *cell* as its fundamental unit of information. Data are stored on a grid containing rectangular *cells* of uniform size. *Cell* dimensions denote data *resolution*. Each *cell* within the grid can be assigned a data value. Smaller grid *cell* size therefore allows more data to be stored for a given geographic *area* (i.e., it can present data at a larger *scale*).

Each *cell* is assigned a value (also known as *category value*) and a *label* (also known as *category label*) describing the type of data found there. Each *cell* is also associated with a specific geographic x,y coordinate location.

Each GRASS *raster data file* describes a specific thematic map in terms of its geographic coordinates. Supporting files describe the *category* values, *category labels*, map title, history, etc., corresponding to this *raster data* file. All of the files relating to a given *raster data file* are collectively termed a *raster map* layer.

Raster data files are stored under the "cell" directory under each *mapset*. For more information on the *location* and content of *raster data* and *supporting files*, see chapter II.

Although users can input, display, and output maps in GRASS *vector* and *sites file* formats, most GRASS analysis programs function on *raster data*. If the user wishes to analyze data, (s)he generally must convert the data to *raster format*. Imagery data are already in *raster data* format.



In the *raster format*, a *point* is represented by a single cell, a *line* is represented by a *line* of neighboring cells in a given direction, and an *area* is represented by a cluster (*grouping* of neighboring cells). This format is the simplest to store in a computer. The whole region of interest is set up as a raster of m rows by n columns. Each *cell* is assigned a single value. The world is viewed as a flat cartesian plane. *Raster format* is best when used for continously varying surfaces such as elevation, slope, slope properties and most environmental analysis using remotely sensed images and numerical modeling.

figure 5: *Raster data* Structure

I.1.3 VECTOR - RASTER CONVERSION

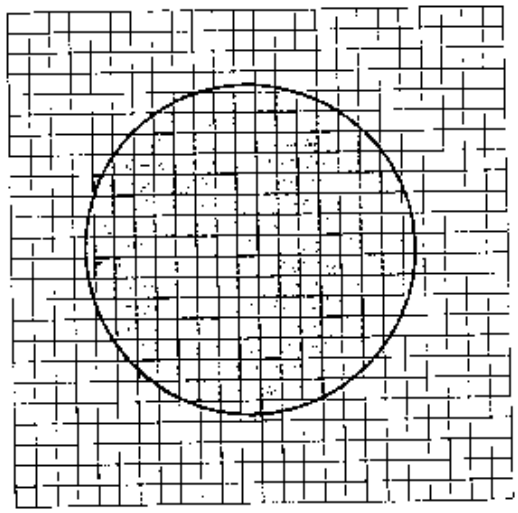


figure 6 "Polygon to Cell Conversion"

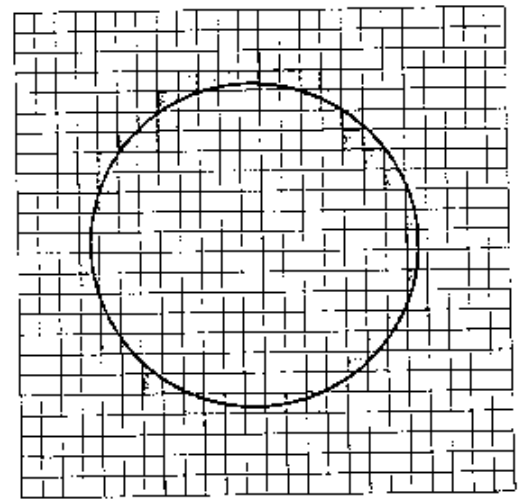


figure 7 "Conversion of Vector to Cell"

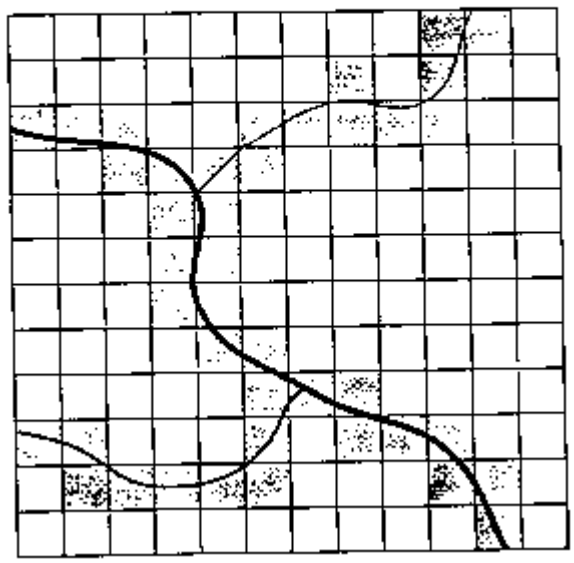
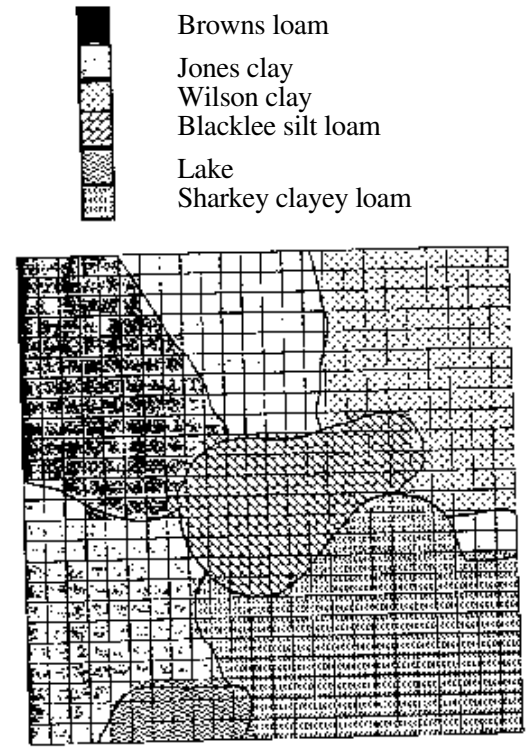


figure 8 "Conversion of Linear Strucure to a Cell Form (Large Cells)"



I.1.4 SITE DATA (IN GRASS)

Site data are *point* data. The spatial *location* of each *site* is given as an x,y coordinate pair. *Category values*, *labels*, and graphic icons may also be associated with sites. In GRASS, *site* data are stored in ASCII files under the "site_lists" directory under each *mapset*.

Site data can be analyzed in GRASS either in their site_lists file format or after conversion to *raster data* format.

Read the information in chapter II, if you wish further information on the *location* and content of GRASS site_lists files.

I.2 HARDWARE

The equipment generally required to operate a *GIS* includes:

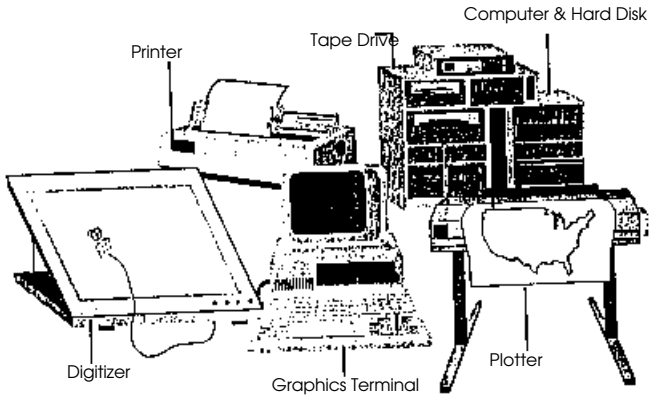


figure 9 "Equipment Required to Run a GIS"

- Computer / Hard disk for computation and storage
- Graphics terminal for display of spatial and tabular data
- *Digitizer* and/or scanner for input of spatial data
- Printer for hard copy output of statistical listings
- Plotter for *hardcopy* output of maps and models
- Tape drive (cassette / 9-track) for input of (mainly) tabular data

I.3 GIS FUNCTION: DATA ANALYSIS

Presumably the main purpose of *GIS* in general, and especially of GRASS, is its ability to analyze data. These abilities may be classified by the following function categories:

- *vector* (polygon) *overlay*and dissolve functions which involve the compositing of multiple map themes in order to create new map data and the associated tabular information

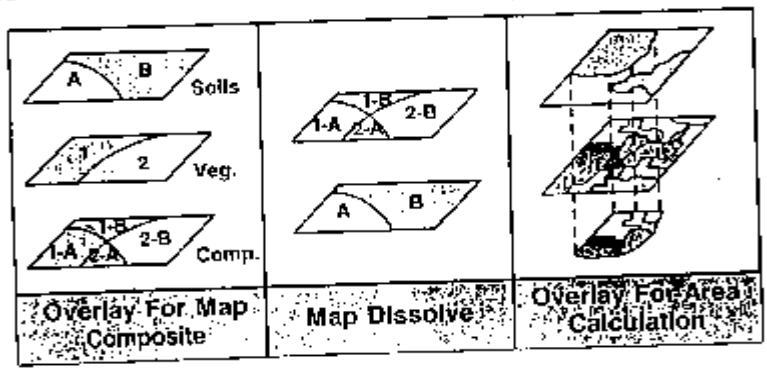


figure 10: *Vector* (Polygon) *Overlay*and Dissolve Functions

- digital terrain analysis which involves the computation of a variety of outputs such as:

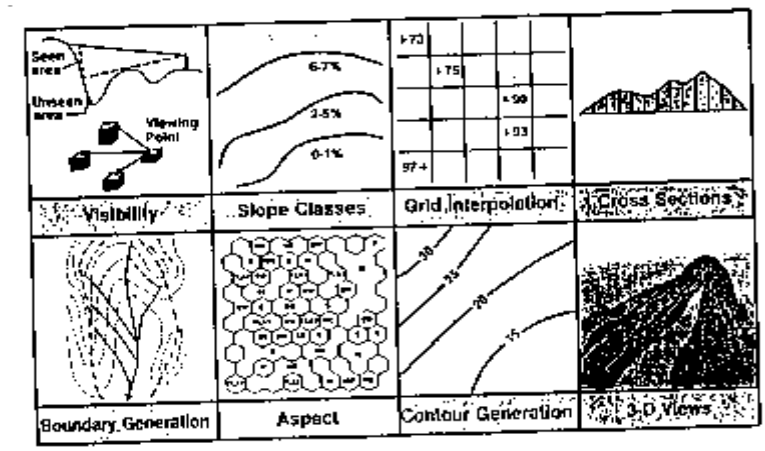


figure11: Digital Terrain Analysis

- analytical functions using grid-*cell* and *vector data*, including:

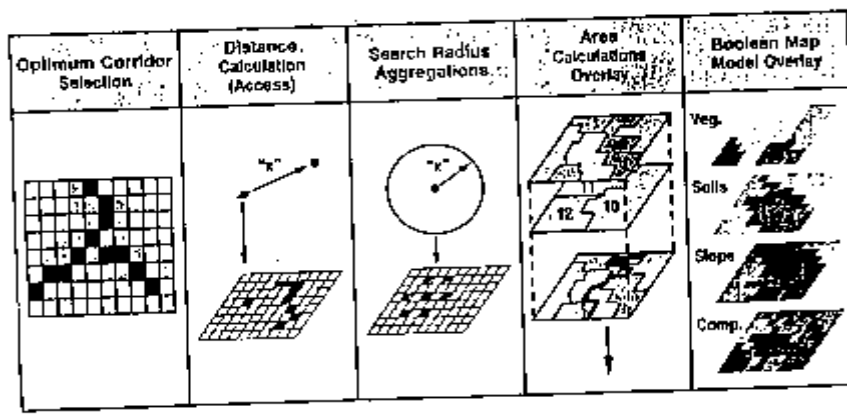
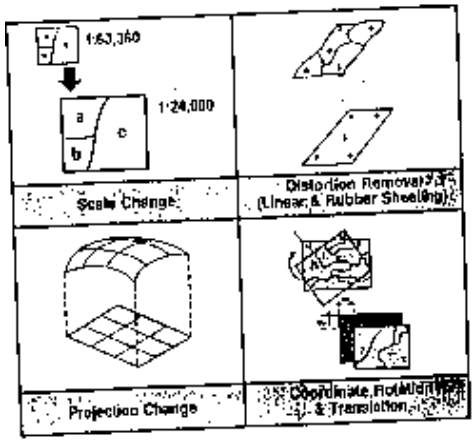


figure 12: Analytical Functions

I.4 GIS FUNCTION: DATA MANIPULATION

A *GIS* can manipulate the digital map information contained in a *database* by



- changing the *scale*
- eliminating distortions (linear and rubber sheeting)
- changing the *projection*
- rotating and translating coordinates

figure 13 "Changing Geometric Properti

- converting grid-*cell* map data to a *vector format* or vice versa

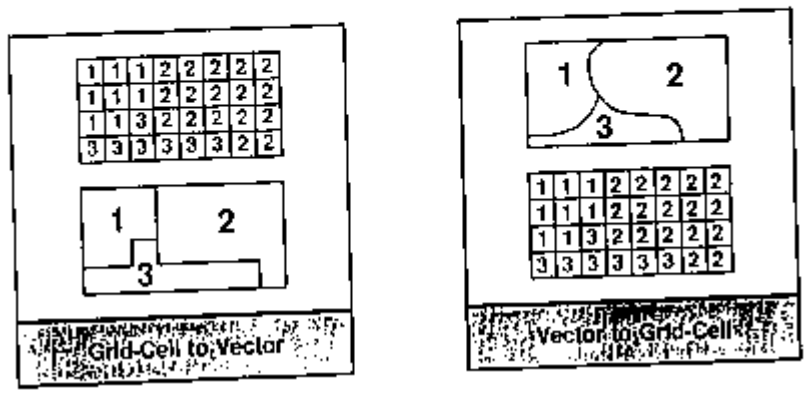


figure 14: Converting Cells to *Vector*s and Vice Versa

- a number of *GIS*s contain functions to make *map generalization*s

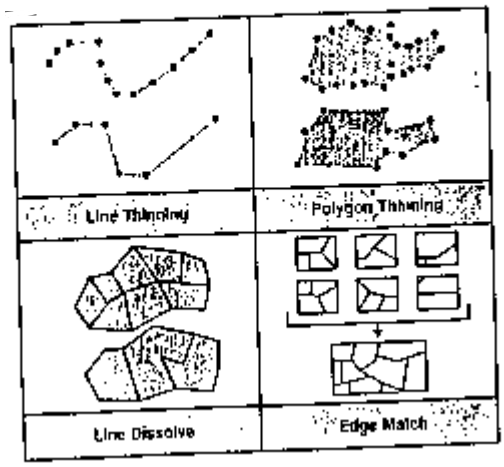


figure 15: Map Genaralizations

I.5 GIS History

Modern *GIS* originates from two sources: computer assisted drafting / mapping (CAD-CAM) and *remote sensing* (analysis of satellite data). The first group resulted in *vector* (*arc-node*) systems whereas the latter led to raster (*cell*) systems. Each has strength and weaknesses; both are merging. Because the early computer systems had small hard disks and memory, display and *hardcopy* devices were *vector*-oriented. *Vector*-based systems were used to produce traditional maps and drawings, while their primary concern was cartographic quality, they provided little for analysis. However, because of slivers (see II.9) and weird polygons this data structure caused problems when used in analysis. The solution to this was the *arc-node* structure which required that each *line* is digitized only once.

Cell data (from satellites) is inherently digital. It amounts to very large volumes (hundreds of MBytes) and lends itself to numerical analysis. However, *remote sensing* hardware was / is very expensive and the fixed *cell* size for each *band* results in all data being in the same *cell* size. So, while it was easy to do numeric analysis, to consider multiple *map layer*s at once, to program and add new features, the early hardware made *remote sensing* not a good match. Therefore, although raster-based systems were much more promising, to the end of 1970s the dominant *GIS*s were *vector*-based. With the decline in harddisk and memory prices, and a new technology that caused all display devices to be raster-based, this is changing now.

II SETTING UP A GRASS DATABASE

The files that GRASS uses and creates are stored in a special place on your system. Exactly where GRASS programs and data are stored on your system will depend on where you install GRASS. However, whereever GRASS is installed on your system, it will use a pre-defined directory structure.

II.1 GRASS DIRECTORY TREE:

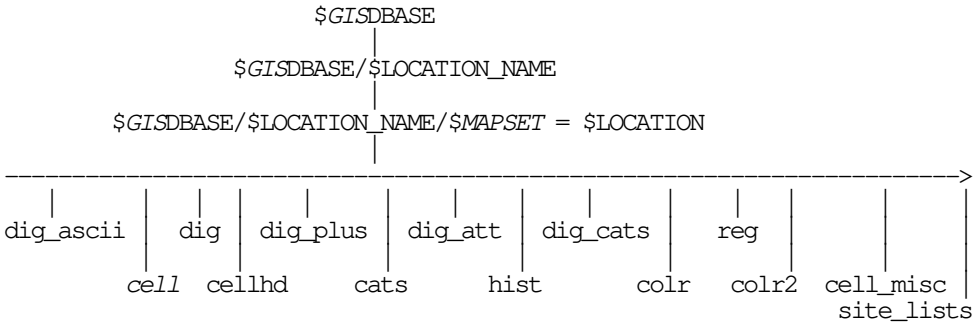


figure 16: The GRASS Directory Tree

It is rather important to adhere to the directory structure given by GRASS. A number of programs read and write to a particular branch within this tree. Both, names and position of these directories are hard-coded into the program; e.g. the import/export routines for ARC/INFO expect a sub-directory 'arc' and certain programs such as *r.combine* or *r.weight* create their own sub-directories.

All users' GRASS data are stored under a directory which is also known by the (*UNIX*-) variable name $GISDBASE. The $GISDBASE variable stands for "GIS Data Base directory". For example, at your site, the $GISDBASE directory might be /usr/home/grass/data.

Beneath this directory, there are subdirectories which are created when the user enters GRASS. Each subdirectory of the $GISBASE directory is termed a GRASS "location", and is known by the variable $LOCATION_NAME. The *location* combines all data of a specific *geographic region*. Here, the *map coordinate system* and minimum and maximum coordinates are stored. All *database* queries and modifications pertain to the chosen *location* only; it is not(!) possible to access multiple locations. Typically, each of these *location* directories will store data for a different geographic location. However, this is up to the user.

Beneath each *location* subdirectory, there will be further subdirectories called "*mapset*" subdirectories. Each "*mapset*" contains a set of maps. All of the *mapset* directories stored beneath a given *location* should contain data relevant to the same general *geographic region*. All of the *mapset*s stored under the same *location* directory must(!) use the same *map coordinate system* and map *projection*. Within the location, at least one *mapset* needs to be defined. The *mapset*s relate to the different project subjects as for example infrastructure, landuse, etc.

In addition to each user's individual *mapset*, each *location* directory must(!) contain a special *mapset* directory named "PERMANENT". This directory contains default information for all of the *mapset*s under a given location. It is made automatically by GRASS whenever the user establishes a new *location*. GRASS always defines a *mapset* 'PERMANENT' which is supposed to be used as an archive. Under normal conditions, you will work in a self-defined

mapset while using some base maps stored in 'PERMANENT'. The *mapset*s are transparent to reading operations, but writing can only be done in the current *mapset*. To change a *mapset*, you have to exit GRASS and choose another one upon starting a new session.

Under each *mapset* directory, GRASS data directories are stored. These directories are created by GRASS as they are needed. There will also be some files under each *mapset* directory. The major data directories and files found under a GRASS *mapset* directory are listed below:

II.1.1 GRASS RASTER DATA DIRECTORIES:

cell:	Contains data files in GRASS *raster format*.
cellhd:	Contains header information for data files in GRASS *raster format*.
cell_misc:	Contains histogram and range data for GRASS *raster files*.
cats:	Contains *category value* and describing *label* information for files in GRASS *raster format*.
hist:	Contains history files noting how GRASS raster *map layer*s were created.
colr:	Contains color table files for raster *map layer*s in the user's current *mapset*.
colr2:	Contains color table files modified by the user for raster *map layer*s not(!) in the user's current *mapset*.

II.1.2 GRASS VECTOR DATA DIRECTORIES:

dig_ascii:	Contains ASCII files in GRASS *vector* (dig) format.
dig:	Contains binary files in GRASS *vector* (dig) format.
dig_plus:	Contains *topology* information for files in GRASS *vector* (dig) format.
dig_att:	Contains the locations at which *category labels* will be placed for *files* in GRASS *vector* (dig) format.
dig_cats:	Contains the *category* text *labels* used for files in GRASS *vector* (dig) format.
reg:	Contains the geographic coordinates of *point*s used to register to a known coordinate system a (*vector*) map *layer* digitized with the GRASS "digit" program.

II.1.3 GRASS SITE DATA DIRECTORIES:

site_lists:	Contains ASCII files in GRASS *site* data format.

II.2 MAPSET FILES:

Some general information is stored in directories with capital letters. Two of these files, "DEFAULT_WIND" and "MYNAME", are special files found only under the "PERMANENT" *mapset* under each "location".

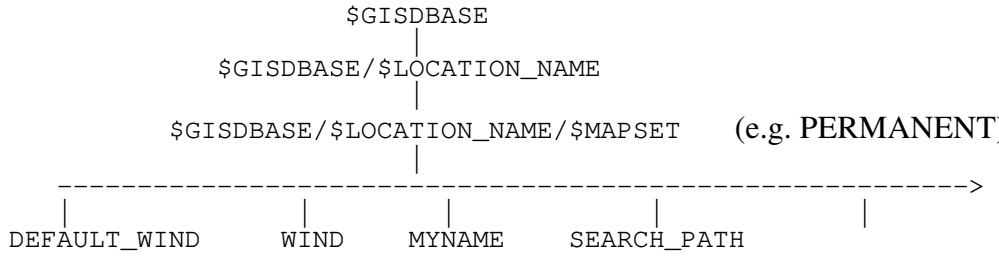


figure 17: Environment Variables for GRASS Directories

The "WIND" *file* is the user's "current window", and contains the geographic boundaries and data *resolution* used by GRASS programs. The user can alter these settings as (s)he wishes during a GRASS session.

The "DEFAULT_WIND" *file* contains the geographic boundaries and data *resolution* to be used if the user's current *mapset* does not contain a "WIND" *file* with this information. This *file* is stored under the PERMANENT "*mapset*" directory.

The *file* "MYNAME" simply contains the name of the *geographic region* to which maps under the "current location" refer. For example, this *file* might simply say "Bizerte, Tunisia". This *file* is stored under the PERMANENT "*mapset*" directory. Each *mapset* has at least one defined window which - in GRASS - is called a region. The region is defined by its northernmost, southernmost, easternmost and westernmost *point*s and its *cell resolution*. North-south and east-west *resolution* need not to be the same, thus allowing non-square units to exist. The region window is always directed to the north, therefore imagery data usually have to have a region definition that is larger than their actual rows by columns size. The concept of (geographic) regions is important in GRASS, because most data handling is limited to the currently active region. Data import, for example, will neglect any information that doesn't fall into the current region.

The "SEARCH_PATH" *file* contains a listing of the "*mapset*s" whose data files the user wishes to use, in the order these "*mapset*" directories are to be accessed. This allows a user to have non-destructive access to other relevant maps for a geographic *area* without modifying other users' original data. It is possible to prevent others' access to your *mapset* directories.

The below screen is the initial entry screen all users see upon entering GRASS. The "location", "*mapset*", and "*database*" directories that the user names here will become that user's "current location", "current *mapset*" and "current *database*" until the user exits the current GRASS session.

```
*****************************************************************************
                              GRASS 4.0

LOCATION: This is the name of an available geographic location.
          For example, "spearfish" is the sample data base location for
          which all GRASS tutorials are written.

MAPSET:   Every GRASS session runs under the name of a MAPSET.  Associated
          with each MAPSET is a rectangular COORDINATE REGION and a list
          of any new maps created.

DATABASE: This is the UNIX directory containing the geographic databases

          The REGION defaults to the entire area of the chosen LOCATION.
          You may change it later using the command:  g.region
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

LOCATION:   bizerte_______          (enter list for a list of locations)
MAPSET:     ndvi__________          (or mapsets within a location)

DATABASE:   /usr/home/grass/data_____________________________


             AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                           (OR <Ctrl-C> TO CANCEL)
*****************************************************************************
```

figure 18: GRASS Entry Screen (I) (example)

For example, assume that you choose to create the new *LOCATION* "bizerte" and the new *MAPSET* "ndvi", and that the directory in which users' GRASS data are stored on our system is "/usr/home/grass/data". Enter this information on the blanks provided on the GRASS Entry Screen, using the "Return" (or "Enter") key to move the cursor from *line* to *line*. When done, press the "ESC" (escape) key to save this information and continue on in GRASS.

II.3 SETTING UP A GRASS DATABASE: LOCATIONS AND MAPSETS

Each time you enter GRASS, you will be asked to select a "current *location*" and a "current *mapset*" with which to work, and to name the *"database"* directory in which this *location* and *mapset* are stored.

While in GRASS, any data that the user creates are stored under the user's "current *mapset*". However, the user may copy, and otherwise use, data stored under any of the "*mapset*" subdirectories of the user's "current location". The user may not destroy or alter data stored under *mapset*s other than the "current *mapset*".

If you wish to create a new GRASS *database* location, simply assign names to the *LOCATION* and *MAPSET*, and enter the full pathname of the main *database* directory in which your data will be stored, on the GRASS Entry Screen. Because the "LOCATION" and "*MAPSET*" will become the names of the *UNIX* directories in which your data are stored, you must assign them valid *UNIX file* names.

```
LOCATION:   bizerte_______          (enter list for a list of locations)
MAPSET:     ndvi__________          (or mapsets within a location)

DATABASE:   /usr/home/grass/data_____________________________________


            AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                        ( OR >Ctrl-C> TO CANCEL )



LOCATION    <bizerte>  -  doesn't exist

Available locations:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
tunis  sfax  touzeur  gafsa
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you like to create location <bizerte> ? (y/n)
```

figure 19: GRASS Entry Screen (II) (example)

If we choose a "location" called "bizerte" that did not previously exist, GRASS will show us a screen that lists already existing locations, and then asks us to supply information for the new *location* "bizerte":

```
**********************************************************************
To create a new LOCATION, you will need the following information:

1. The coordinate system for the database
       x,y (for imagery and other unreferenced data)
       UTM
       State Plane
       Latitude-Longitude
2. The zone for the database
       (except for x,y and Latitude-Longitude databases)
3. The coordinates of the area to become the default region
   and the grid resolution of this region
4. A short, one-line description or title for the location

Do you have all this information for location <bizerte> ? (y/n)
**********************************************************************
```

figure 20: Creating a New Location

The user is prompted for the above information and simply inputs the appropriate data. After all data is entered, the data is saved and the user is now ready to continue within GRASS.

After gathering this information GRASS prompts for the coordinate system to be used at the new location. If appropriate, GRASS will next prompt the user to enter the *projection zone* in which data under his *mapset* will lie.

The user will also be prompted to enter a one-line description of the *location* (e.g., "Bizerte, Tunisia"). This description will be saved in the special *file* named "MYNAME" and stored under the "PERMANENT" *mapset* under the new *location* "bizerte". It will be assumed to pertain to all *mapset*s under the "bizerte" *location* directory.

After this, the user is asked to define default *geographic region* settings. This information will be saved in the special *file* named "DEFAULT_WIND" and stored under the "PERMANENT" *mapset* under the new *location* "bizerte". Any *mapset* stored under this *location* ("bizerte") will default to use of the geographic boundaries and *cell resolution*s shown in the "DEFAULT_WIND" *file* if(!) the *mapset* does not contain a *file* called "WIND" with these settings. These geographic boundaries should be large enough to encompass all of the user's Bizerte data. Using the *g.region* command the user can modify the "WIND" *file* containing these settings under his own *mapset* later, in GRASS, as necessary.

After entering the geographic boundaries and *cell resolution*s the "ESC" (escape) key is used to save this information and continue.

GRASS then asks the user about the *mapset* (s)he wishes to create. After the user enters this information, the "GRASS" prompt will appear, signalling that the user's new GRASS *location* and *mapset* directories are set up and GRASS is ready to go.

In the sample screen below, the user is asked to set default region settings for a *UTM* coordinate system *database location* in *zone* 10. Depending on the *projection* and data source used, edge coordinates usually are 6 - 8 digit figures. In case of imagery data, however, the edges reflect the coordinates of a window that encompasses the whole image, measured in number of pixels (pixels, or picture *elements*, are the *cells* that make up an *image* or raster file). The grid *resolution* describes width and height of each *cell* (usually in meters).

```
**********************************************************************
                        DEFINE THE DEFAULT REGION
```

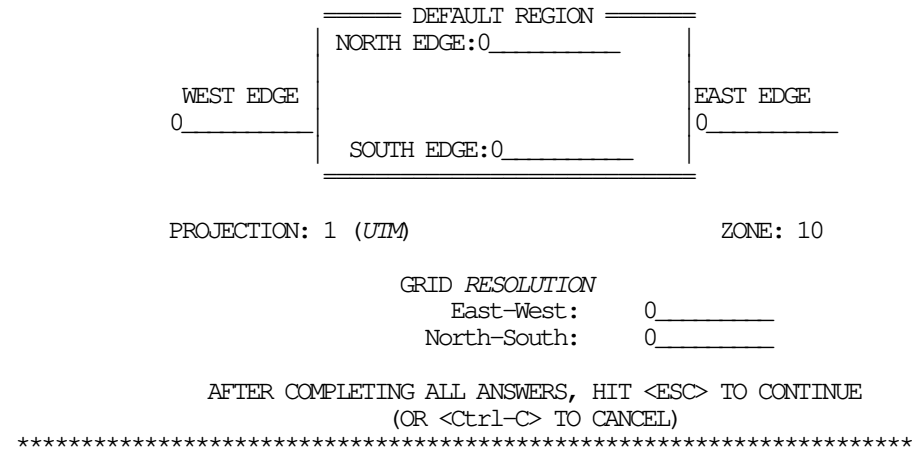


figure 21 "Defining a New *Geographic region*"

II.4 COORDINATE SYSTEMS AND PROJECTIONS

A *map coordinate system* is either a geographic coordinate system (like *UTM* or latitude/longitude) or an arbitrary coordinate system (like the x,y coordinate system initially assigned to importe satellite imagery) used to reference the locations of objects in a two-dimensional space.

A map *projection* describes the manner in which the spherical surface of the earth is represented on a two-dimensional surface. It attempts to minimize distortion in area, shape, distance and direction, but these are mutually exclusive aims. Therefore users ought choose the *projection* that is the most apropriate for their needs.

All of the data stored beneath the same GRASS "location" directory must use the same *map coordinate system* and map projection. This means that latitude/longitude data, Universal Transverse Mercator data, State Plane coordinate system data, and unreferenced (x,y) imagery data, should be stored under different GRASS "locations" and "*mapset*s".

Often, one wishes to combine data that are available in different coordinate systems and/or different map projections. To do this, users must convert these data to a common *map coordinate system* and projection. The commands listed below are useful to those importing and exporting data to and from GRASS. They can be used to perform *map coordinate system* and map *projection* conversions.

m.datum.shift	*r.mapcalc*	*v.transform*
m.gc2ll	*r.in.ll*	
m.ll2gc		
m.ll2u		
m.u2ll	*m.region.ll*	*m.tiger.region*

Satellite *imagery* data are generally made available in an x,y coordinate system. In order to analyze imagery data in combination with other GRASS data layers, it is necessary that all these data be referenced to the same *map coordinate system* and projection. Several GRASS programs are used to rectify imagery data.

i.points	*i.rectify*	*i.rectify.blk*

See also the references for:

"Map Projections - A Working Manual" (Snyder, 1989),
"The Department of Defense World Geodetic System 1972 (Seppelin, 1974).

II.5 RESOLUTION

In *raster data* format, data are stored on a grid containing rectangular *cells* of uniform size. The *cell* is the smallest unit in which information can be stored. *Cell* dimensions are defined by the user.

Each *cell* within the grid can be assigned a data value and a *category label* describing the type of data found there. Because each *cell* covers an *area* rather than a single *point*, this data value may be a generalization of all the actual data values found within this *area* on the earth. Whether the data values assigned to each *cell* represent a generalization, or represent all of the actual data found within the cell, will depend on the cell's dimensions and the specific nature of the data. The user should define *cell* dimensions carefully.

Grid *cell* size is linked to map *scale* and *resolution*. Smaller grid *cell* size allows more data to be stored for a given geographic *area*. Therefore, smaller grid cells are associated with a larger *scale* and a finer *resolution* than are larger grid cells. The *cell resolution* represents the size (area) of the piece of the earth to be represented by one *cell* in a raster *map layer*.

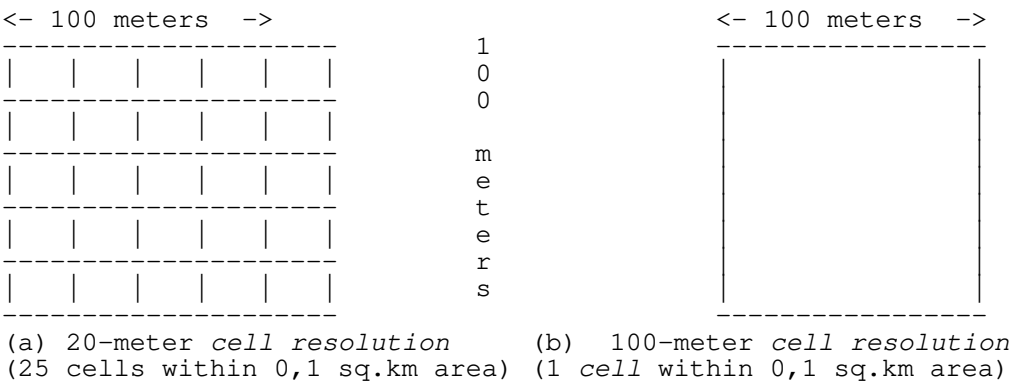


figure 22: *Raster map Layer Resolution* (example)

The map *layer* at left (in which each *cell* is 20 meters long and 20 meters wide) can hold 25 times the amount of data as can be contained in the map at right (in which each *cell* is 100 meters long and 100 meters wide).

Each *cell* is also associated with a specific geographic x,y coordinate location. In GRASS, the user specifies *cell* dimensions in the x (east-west) and y (north-south) directions, for each map. It is permissible to make the east-west *resolution* different from the north-south *resolution*.

Once the *resolution* of a new *raster map layer* is set, the spatial information available in that map *layer* is set. The user may subsequently use the GRASS *g.region* command to change the working *resolution* of a raster *map layer*, but the original *resolution* of the map remains intact and available to the user regardless of any resampling performed as a result of changes in the *cell resolution* designation in the current *geographic region*. Increasing the *resolution* of an

existing *raster map layer* will only subdivide the information from the existing cells. It is important to understand that increasing the *cell resolution* of an existing *raster map layer* cannot actually access more data than were originally present; even worse: misleading interpretation may be caused by a such increased *resolution*.

II.6 HOW TO FIND OUT ABOUT FILE CONTENTS

There are a number of commands for obtaing information about existing GRASS data. To find out, what files of a certain data type are existing within a given *mapset*, you enter *g.list*. While most users will prefer to use some *file* manager program which is part of their operating system, this command is particulary useful in shell scripts, which shouldn't rely on any other program. Although strenuous at the beginning, the GRASS user is invited to include meta information whereever this is possible. One command that makes use of this is *r.info*. It will display name, type, size, projection, data source, data description and comments attributed to a raster *file* (once the user has entered this information). Sometimes though, we want to select a raster *file* because of its characteristic *category* values. This is the kind of information that *r.describe* will give us. In case these *category* values aren't descriptive enough, *r.cats* will print the labels (text describing *category* values) together with the values themselves. An even more complex account of the content of a raster *file* gives *r.report*. Here, the number of cells per value is counted (and translated into percentage cover or units of measurement).

Examples :

g.list	*type=rast*	lists all raster files in the current *mapset*
r.info	*map=bizerte*	brief statistics on the map "bizerte"
r.describe	*map=bizerte*	description of the map "bizerte"
r.cats	*map=bizerte*	lists the *category* values in map "bizerte"
r.report	*map=bizerte units=hectacres > filename*	summarizes the combined *area* (in hectares) for each category

II.7 GENERAL FILE MANAGEMENT COMMANDS

Although it is possible to use *UNIX* commands to handle GRASS files, this is not advisable unless you are an experienced GRASS user. A single map may consist of numerous files (e.g. *cell* header, color table, categories) in different directories that are internally linked. Therefore, to copy, rename or delete a map, you are advised to use the appropiate commands *g.copy*, *g.rename* and *g.remove*. While the user is confined to work during one session within one *location* only, the current *search path* for *mapset*s (window or scale/*resolution*) may be changed with *g.mapsets*.
One other *g.* command of greater importance is *g.region*. This program allows the user to manage the boundary definitions of the current *geographic region*. The region definition files are stored in the *windows* directory in the user's current *mapset*. One region *file* is DEFAULT_WIND that defines the extent of the data base that is always stored under the *mapset* PERMANENT. *d.zoom* allows to set the region boundaries interactively. (see chpt. III)
Examples:

g.copy vect=roads, streets
g.mapsets mapset=PERMANENT, *DEM*, ispa, medine
g.region n=n+1000 *e=e*+1000 *w=w*1000 *s=s*1000 *raster=DEM*

g.region save=new_window_name

```
Mapset <PERMANENT> in Location <bizerte>
GRASS 4.0 > exit

GRASS SESSION WRAPUP

You have just finished working on mapset: <PERMANENT>

The following Raster maps belong to it:
aspect  geology roads   landuse soils_ph

The following VECTOR maps belong to it:
elev    soils   transport

The following SITES maps belong to it:
arch    elev    temp    rain

Do you wish to selectively remove data files? y/n [n]

        GOOD BYE from GRASS
```

figure 23: Exiting GRASS

II.8 APPLICATION: DEVELOPMENT OF A GRASS DATABASE

II.8.1 DATA DEVELOPMENT FROM POINT DATA SOURCES

Point Source Data

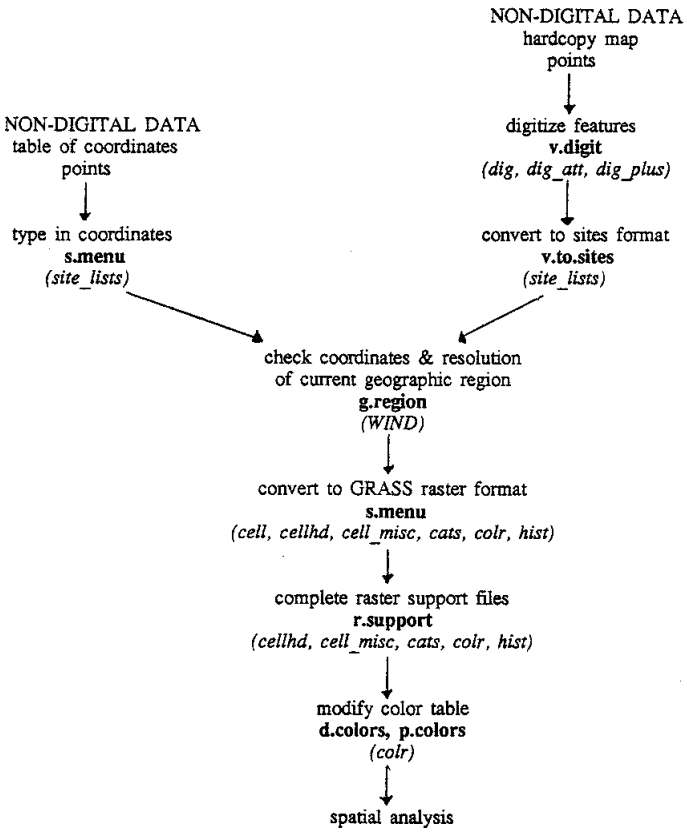


figure 24: Workflow With *Point* Source Data

Import and Manipulate *Site* Data:

GRASS sites have the following ASCII format:

Easting | *Northing* | #n label

where

Easting is the *Easting* coordinate
Northing is the *Northing* coordinate
#n is the *category value* (attribute)
- if missing, a binary map is created
label is the *category* description

II.8.2 Data Development From *Vector data* Source

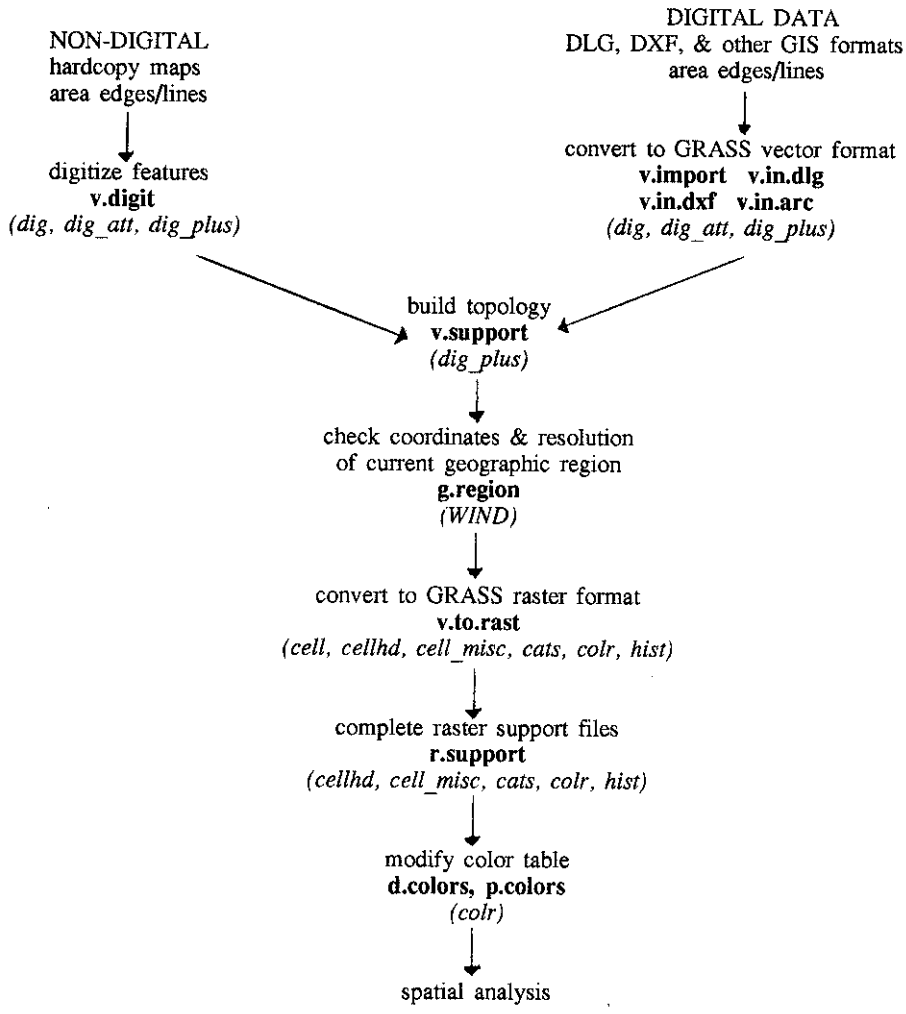


figure 25: Workflow With *Vector* Source Data

v.import - creates binary GRASS *vector file*s from ascii format

v.in.arc - converts ARC/INFO's "Generate" format into GRASS *vector*s
1. export from ARC/INFO with "Ungenerate"
2. create $LOCATION/*arc* directory
3. copy ARC/INFO *file* to this directory
4. run *v.in.arc*
5. run *v.support*

v.in.ascii - converts ascii *vector* map into a binary *vector* map
1. create $LOCATION/dig_ascii directory
2. copy ascii *vector file* to this directory
3. run *v.in.ascii*

v.in.dxf - converts DXF files into GRASS ascii or binary *vector file*s
1. create $LOCATION/dxf directory 2. copy dxf *file* to this directory
3. run *v.in.dxf* 4. run *v.support*

II.8.3 DATA DEVELOPMENT FROM RASTER DATA SOURCES

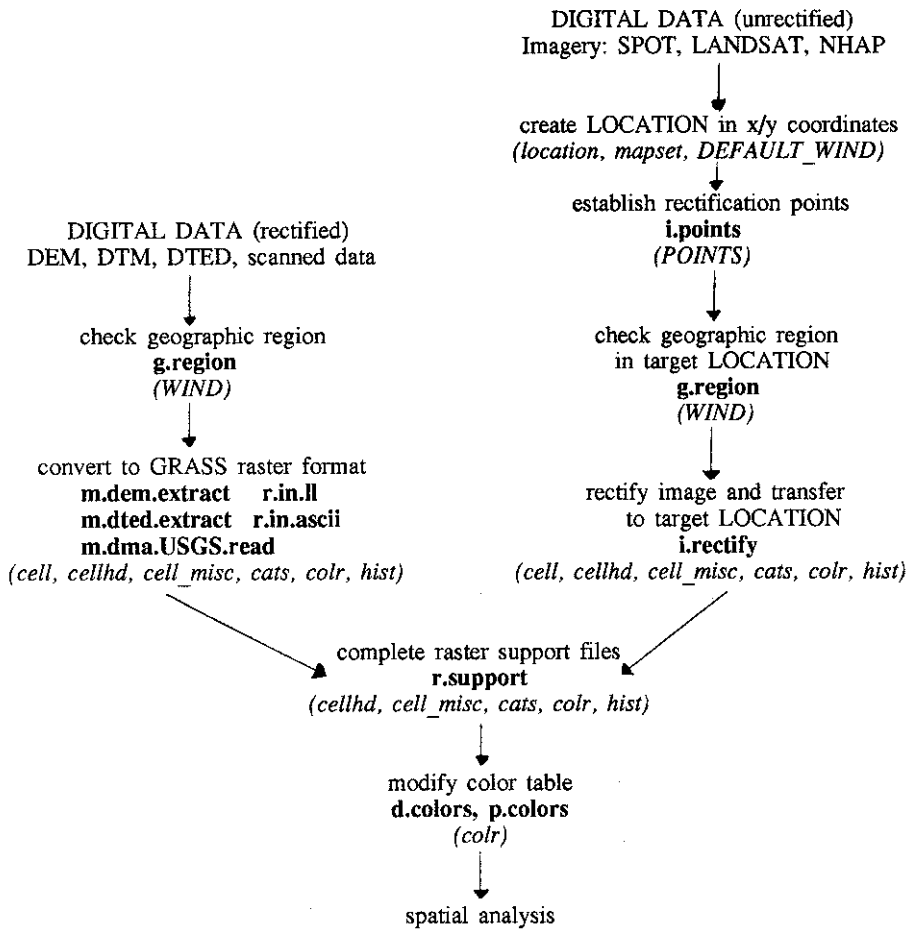


figure 26: Workflow With Raster Source Data

Import / rectification of remotely sensed (raster) data:
(there is a special chapter on *image* processing)

Create an X,Y coordinate location

north edge	-0.50	south edge	: -1000.50
west edge :	0.50	east edge	: 1000.50
resolution :	1		

i.tape.other - extracts scanned aerial imagery, *SPOT* and *TM* imagery from ½ inch tape
i.group - creates and edits groups of imagery files
i.target - "targets" an imagery group *file* to a GRASS *LOCATION* with a geographic coordinate system
i.*point*s - enables user to mark coordinate system *point*s on an *image* to be rectified
i.rectify - rectifies an *image* by computing a coordinate transformation for each pixel in the image

II.8.4 DATABASE QUALITY

Introduction of *GIS* into an agency often forces more systematic data planning. While building a (new) *database*, the developer has to cope with all kinds of barriers. Data may have to be collected, compiled and input to support analysis. Even if data has already been collected, attention should be paid to their accessability, cost, age, areal coverage and format. Problems may also arise because of inapropriate original map *scale* and/or the density of observation *point*s. Accuracy of the work done should be controlled by verification plots and ground truthing. Although strenous in the beginning, the usage of meta data will enhance the quality of future applications. History files as they may be created while executing a number of GRASS commands, make a first step in building such a *database* of meta information (see table 1 "History *File* Formats for GRASS *Map layer*s" at the end of this chapter). Finally, good working conditions light correct lighting at the workspace or a large *monitor* are as necessary as detailed contracts with outside providers of data.

When combining *vector map layer*s the user will eventually encounter errors like duplicate or missing lines, duplicate labels and *snap*ping nodes. The programm *v.support* (see the following example of an example output running v.support) is very useful for *editing* such errors.

Labeling error:
(interpreting a *v.support* script)

Number of lines:	49
Number of nodes:	30
Number of areas:	20
Number of isles:	1
Number of atts:	19
⇒ 1 *attribute* is missing !	
Number of unattached atts:	0
Snapped lines:	0

figure 27 "Sample Output of v.support"

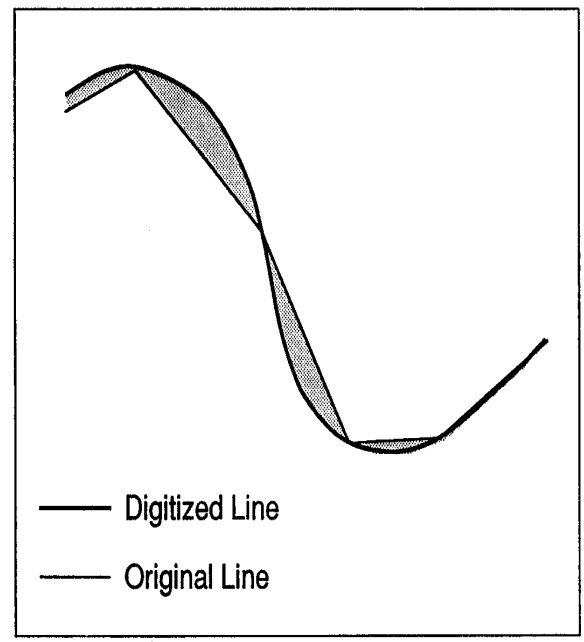


figure 28: Digitzing Error (slivers)

Especially caution should be used when combining multiple raster *map layer*s. The order of maps used as input layers to programs such as *r.combine* or *r.patch* has a significant influence on the resulting output map *layer* (see figure on next page).

Map 1

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

Map 2

2	2	2	2	2	2
2	2	2	2	2	2
2	2	2	2	2	2
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

r.patch
first layer Map1
second layer Map2
resultant map Full.Map1

--- or ---

r.patch Map1 Map2 Full.Map1

Full.Map1

1	1	1	2	2	2
1	1	1	2	2	2
1	1	1	2	2	2
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

r.patch
first layer Map2
second layer Map1
resultant map Full.Map2

--- or ---

r.patch Map2 Map1 Full.Map2

Full.Map2

2	2	2	2	2	2
2	2	2	2	2	2
2	2	2	2	2	2
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

figure 29: Usage of *r.patch*

Table 1: History *File* Formats for GRASS *Map layer*s

GRASS Map Layer History Report

Layer name:
Location:
Mapset:
Layer title:
Version of GRASS used to create layer:
Type of layer:
Number of categories:

Region Used to Create Layer

Coordinate system (*UTM*/latlong):
Zone:
N:
S:
E:
W:
N-S *resolution*:
E-W *resolution*:

Raster Layer Information

Rows:
Columns:
Total grid cells:

General Information About Source Data

Name of source data:
Date source data was published/created:
Publisher/creator of source data:
Scale of source data:
Source media (tape, paper, mylar, etc):
Condition of source data (good, wrikled, accuracy problems, etc.):
Format of source data if tape (ascii/binary, *vector*/raster, *SPOT*/*DEM*, etc.):
If source data is imagery, *SPOT*/*LANDSAT* identification number:
If source data is a format other than GRASS, programs used to *import* data:
Comments:

(Re)compilation of Source Data

Date of (re)compilation:
Name of person/agency who (re)compiled source data:
Procedures/problems with (re)compiling:
Comments:

Digitizing/Scanning of Source Data:

Date source data was digitized/scanned:
Name of person/agency who digitzed/scanned:
Digitzing treshold/*resolution*:
Map treshold/*resolution*:
Min/max RMS *residual*s from re*GIS*tration (list *residual*s for each map sheet used to create layer):
Procedures/problems in digitzing/*scanning* data:
Hours spent digitizing/*scanning*:
Comments:

Conversion from Import Format to GRASS Format

Date *import layer* was converted to GRASS format:
Name of person/agency who converted *import layer* to GRASS format:
Procedures/problems in conversion from *import* format to GRASS format:
Hours spent converting to GRASS format:

Comments:

Conversion of Vector Layer to Raster Layer

Date *vector layer* was converted to raster layer:
Name of person/agency who converted *vector layer* to raster layer:
Procedure/problems during *vector*/raster conversion:
Hours spent converting from *vector* to raster:
Comments:

Analysis, Application, Reclassification, etc

Data analysis/application was perforemd:
Name of person/agency who performed analysis/application:
Procedure for analysis/application:
Hours spent for analysis/application:
Re*class* rule used:
Mapcalc equations used:
Comments:

III VISUALIZATION

III.1 GRASS GRAPHICS DISPLAY COMMANDS

While most of the *file* managing and data analysis may be performed in front of an alphanumerical terminal (*CRT*), visualization of results requires a graphic display terminal (capable of displaying at least 256 colors) that is used in conjunction with a *mouse* or plotter. On the *monitor* side, GRASS relies completely on X-Windows. Within a single session one may open up to eight monitors (processes, each possessing a graphic window either on the same hardware screen or distributed among a network of graphics terminals). This differentiation between graphics terminals, logical monitors and windows (within one monitor) causes a lot of confusion among beginners in the *UNIX* world.

Some systems allow more than one color *monitor* to be utilized by the same processor, although additional monitors will be competing for CPU time. In case of a dual terminal configuration the *CRT* is used to enter GRASS commands. This keeps the entire screen of the color *monitor* available for graphic display. Otherwise, if only a single graphic display terminal is available, this unit serves the functions of both the graphics display device and for entry of commands.

GRASS provides users with two sets of functions for designing and outputting maps: the "display" functions, for generating GRASS output to *graphics monitor*s, and the "print" functions, for generating GRASS outputs to *hardcopy* output devices.

The display section of this chapter is divided into 5 categories, involving:

1 Management of the *graphics monitor*,
2 Management of display frames on the *graphics monitor*,
3 Windowing,
4 Reporting of information about the displayed image, and
5 Graphical manipulation of map contents.

III.1.1 MANAGEMENT OF THE GRAPHICS DISPLAY MONITOR

The GRASS program *d.mon* allows the user to start, select, list, query the status of, release control of, stop, and unlock control of, available *graphics monitor*s. See the manual entry for *d.mon* for specific information on program syntax.

See also the discussion of *d.erase* and *d.frame* commands on the following pages.

To display on-screen GRASS graphics, the user must start and select a *graphics monitor*. This is accomplished by the program *d.mon*, which can be run either in *interactive* mode or with the parameters *start*, *select*, *stop* on the command line. The user can start several *graphics monitor*s running simultaneously, if the hardware configuration supports this. Desired monitor(s) can be started once, and left running even when no GRASS session is being run. because the *monitor* program runs in the background. When a *monitor* is started, it is also automatically selected for output, unless the user directs otherwise. The user can also explicitly select a *monitor* that has already been started. This will be necessary if the user has started

multiple *graphics monitor*s running. After a *monitor* is selected, a blank graphics frame should appear on whatever terminal the user is using to display graphics.

Only running (started) monitors can be selected for graphics output. Once the user has selected a *monitor* for output, no other user can use this *monitor* for graphics output until the *monitor driver* is either released (by the user) or unlocked (by any user on the system). The user can run multiple *graphics monitor*s by simply starting each of the *graphics monitor*s *driver*s to which (s)he wishes to direct output. Only the user who has selected a *monitor* for output can release control of it. However, other users can achieve the same effect by running *d.mon* from the command *line* and unlocking another user's control of a monitor. The *d.mon* program can regulate control of *graphics monitor*s both in systems using multiple monitors and in systems using a single *graphics monitor*.

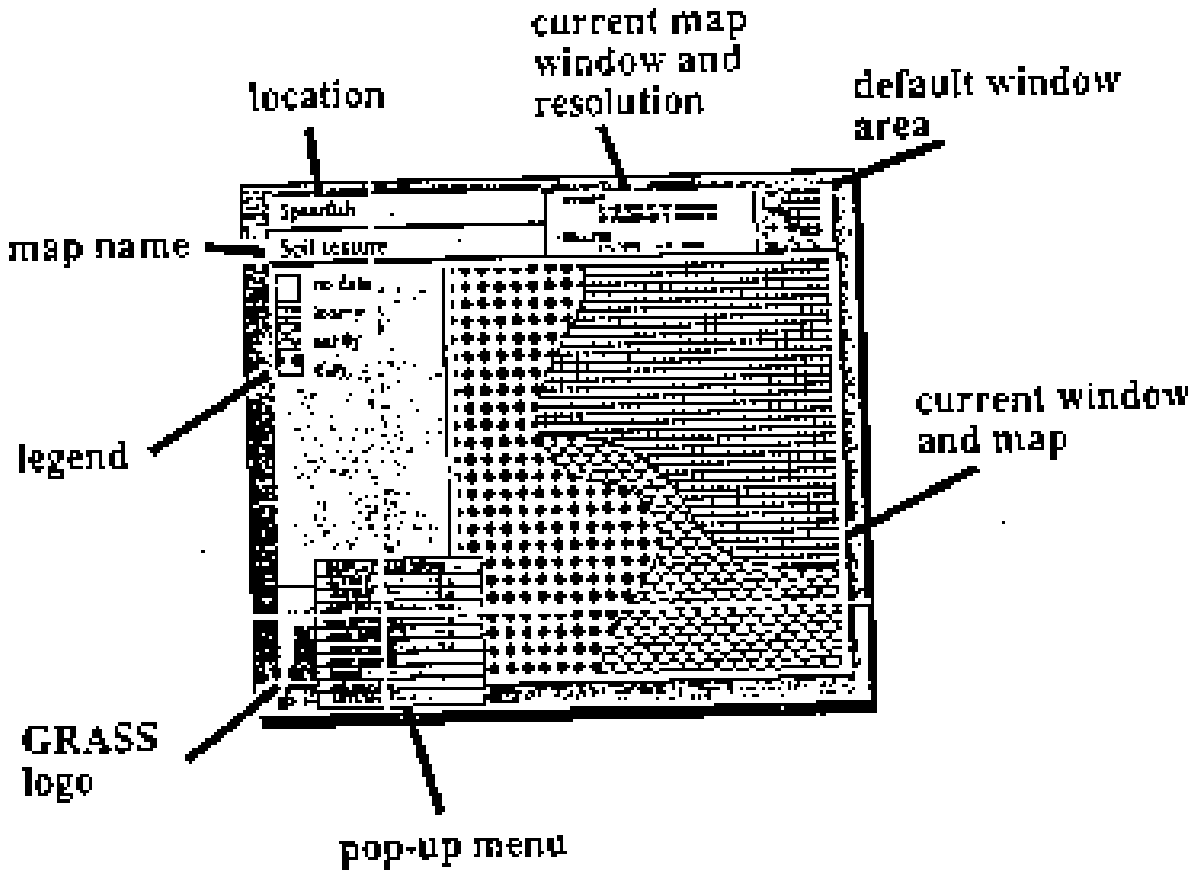


figure 30: Screen Components (within *d.display*)

III.1.2 MANAGEMENT OF DISPLAY FRAMES ON THE GRAPHICS DISPLAY MONITOR

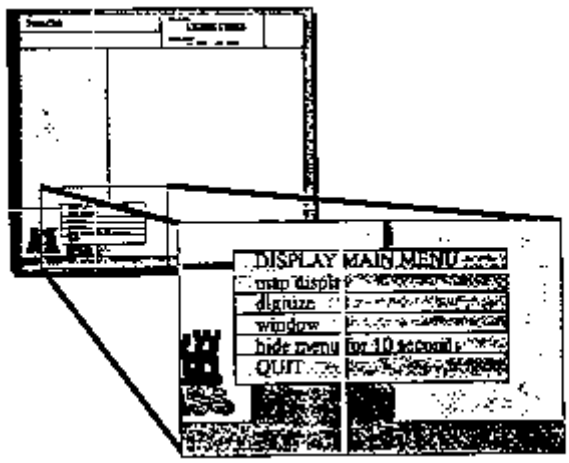
After the user has started and selected a *graphics monitor* for output (using the *d.mon* command), the user can subdivide the *monitor* into different "frames", and display different GRASS outputs in each frame. For example, the GRASS program *i.points* divides the *monitor* into multiple frames to allow the user to *target rectification point*s on an *image* to (user-known) *point*s on another (raster) *map layer*. Similarly, the GRASS macro *d.view.sh* divides the *monitor* into nine frames to demonstrate the 3-d viewing function of GRASS.

The boundaries of a user's current *geographic region*, which define which map areas will be displayed, should not be confused with those of the current display frame, which define the place on the *graphics monitor* in which results will be displayed.

A number of display management functions are integrated into a single program *d.display*. It automatically starts a *graphics monitor* that contains a small *menu* which allows to display maps, sketch simple graphics and manipulate the viewing window within the *graphics monitor*. This program, however, is somewhat closed; it allows to perform only those functions offered by the menu. Measurement, selection and manipulation can be done with *d.mon* only. For the beginner though, this program summarizes the most often used *d.* functions. These commands affect only the display of graphics, and do not alter the user's data.

d.display

Initial display screen



map display -- display cell, *vector* and 3D maps along with sites, labels and map legends. Also color table interaction and measurements.

digitize -- do on-screen digitizing

region -- change map region

hide *menu* -- remove *menu* from screen

figure 31: The Initial Display Screen of *d.display*

III.1.3 WINDOWING

Several GRASS commands exist to manage the two types of 'windows' available in GRASS: *graphics display frame*s and *geographic region*s. Although both are sometimes referred to as 'windows', the user should be careful to distinguish between the two. The boundaries of a user's current *geographic region*, which define which map areas will be displayed, should not be confused with those of the current display frame, which define the place on the *graphics monitor* in which results will be displayed. The following commands are used to manage the frames in which graphics are displayed on the user's graphics display device. These commands affect only the display of graphics, and do not alter the user's data.

d.erase *d.frame*

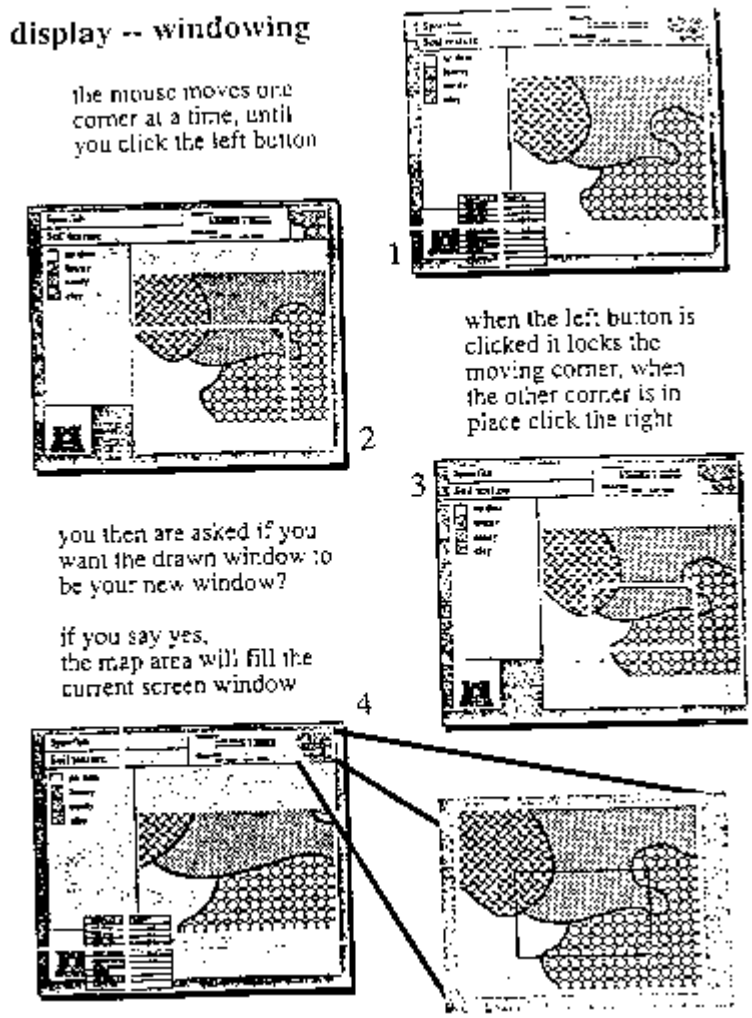


figure 32 "Display Windowing"

Frames are for a *monitor* what a *monitor* is for a terminal. When a *graphics monitor* is started, it contains a single frame occupying the whole display area. The user may however define with *d.frame* certain areas on a *monitor* that the output is directed to. If there is more than 1 frame, the user has to select one to be active. Frames may overlap, but the contents of the frames beneath are not retained (this being a difference to *graphics monitor*s, where each allocates enough memory to restore its contents). Multiple frames within one *graphics monitor* may be used to display different regions in each window defined by a frame. *d.erase* erases the current frame (see figure 33 "Creating Screen Frames" on next page).

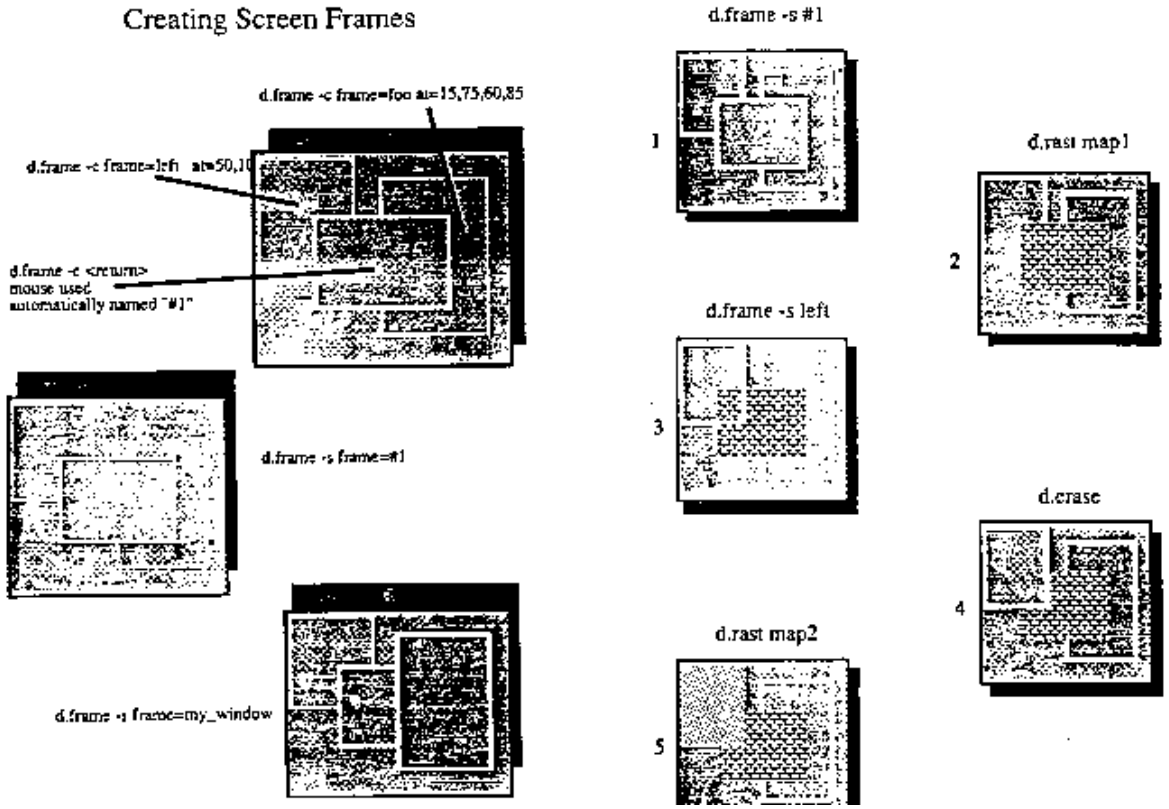


figure 33 "Creating Screen Frames"

Other GRASS commands can be used to modify the settings of the user's *geographic region*. The settings define the geographic boundaries, map *projection*, coordinate system, and *resolution* of data sets. GRASS commands commonly ignore data falling outside of this region. For example, when the user digitizes with *v.digit*, only those areas of the map that fall inside the user's current *geographic region* will be saved to the *vector file* produced by *v.digit*. When the user displays data to the *graphics monitor*, only map areas within the user's current *geographic region* are displayed. When raster analysis programs are run on *map layer*s, only those map areas that fall inside the user's current *geographic region* will be analyzed and included in the resultant raster *map layer*. When the user prints map output, current geographic settings are also used.

If the user does not explicitly modify these region settings, GRASS will use the default settings established when the user's current *location* was first set up; these are stored in the DEFAULT_WIND *file* under the PERMANENT *mapset* and copied to the *file* "WIND" in the user's own *mapset*. However, these settings can be altered using the *g.region* program, or by simply *editing* the "WIND" *file* using a text editor.

The user may wish to use different settings for different purposes and frequently alter the current region settings. These "tailored" settings can be stored under the "windows" directory, located under the user's current *mapset* for possible later use.

The geographic boundaries described in a location's "DEFAULT_WIND" *file* should be large enough to encompass all of the map data contained in any of the *mapset*s under the same location. Its data *resolution* should also be sufficiently fine for data extraction purposes. Note also that all *mapset*s under the same *location* should use the same *map coordinate system* and *projection*.

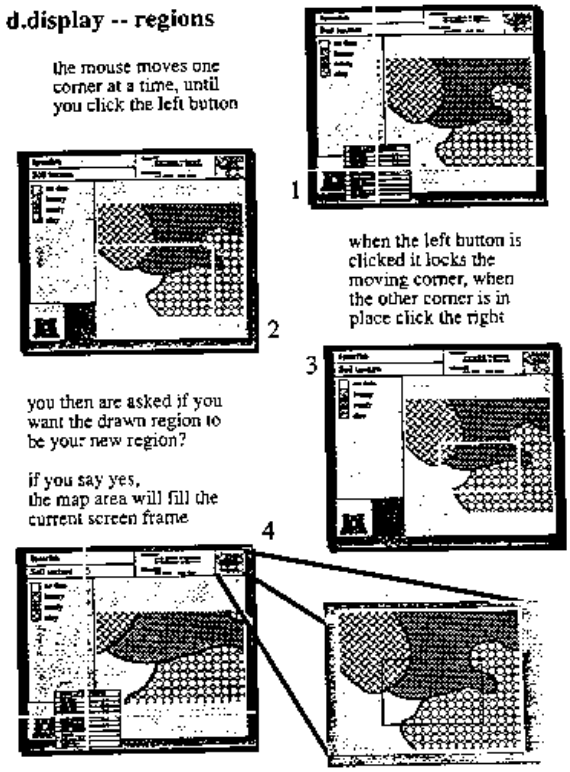


figure 34 "*d.display* - Regions"

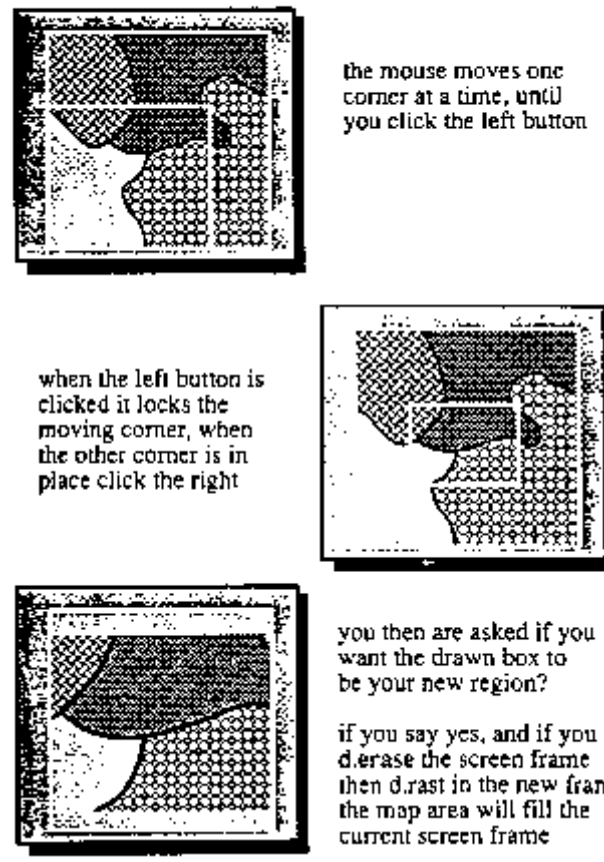


figure 35 "Usage of *d.zoom* to Modify the *Geographic region* Boundaries"

The name of d.*zoom* is somewhat misleading. It does not(!) just change the the scope of view, but sets new boundaries for the *geographic region*. Remember, at the beginning of this chapter we learned that GRASS uses the term (geographic) region to define the window that limits the scope of all data handling.

III.1.4 MAP DISPLAY REPORTING FUNCTIONS

The GRASS functions listed below report information about *map layer*s that are displayed in the current frame on the user's *graphics monitor*. Some of these functions use the map *layer* displayed merely as a frame of reference, and allow the user to query the contents of other maps in his *search path* at a user-cited coordinate location.

- *d.geodesic*
- *d.histogram*
- *d.measure*
- *d.rhumbline*
- *d.rast.arrow*
- *d.rast.num*
- *d.what.rast*
- *d.what.vect*
- *d.where*
- *d.ask*

The graphic equivalent to *g.ask* is *d.ask*. It prompts the user with a list of *map layer*s within the current *mapset*. *d.menu* allows to place menus within the current frame. An example of the usage of this kind of graphics menus is the program *d.display*.

III.1.5 GRAPHICAL MANIPULATION OF MAP CONTENTS

Map display, input, and capture functions give users flexibility in designing maps for display to *graphics monitor*s. GRASS macros can be built to run demos making use of these displays. Screen dumps can be made, to capture the displayed *image* and reformat it for output to a printer.

The display, input, and capture functions can be further categorized as to whether they involve map display, input, or screen capture, and as to whether they involve *raster data*, *vector data*, *site* data, text, or (unreferenced) screen graphics.

	GRASS Programs for		
	Display	Input	Screen Capture
Raster	– d.display – d.rast – d.his – d.rgb – d.profile – d.3d – 3d.view.sh – slide.show.sh	– d.rast.edit – d.save – p.ppm	– d.savescreen
Vector	– d.display – d.vect – d.vect.dlg	– d.display – v.digit – p.ppm	– d.savescreen – d.save
Sites	– d.display – d.*points* – d.sites – d.icons	– d.savescreen – d.save – p.ppm	
Text	– d.display – d.menu – d.text – d.label – d.labels – d.paint.labels – d.legend – show.fonts.sh	– d.display – d.menu – d.font – d.title	– d.savescreen – d.save – p.ppm
Screen Graphics	– d.display – d.scale – d.grid – d.colortable – d.graph – d.mapgraph – d.geodesic – d.rhumbline – d.histogram – grass.logo.sh	– d.colors – d.display – d.colormod	– d.savescreen – d.save – p.ppm

table 2 "Overview To GRASS Programs for Display, Input and Screen Capture"

III.1.5.1 COLOR MODIFICATION OF SCREEN GRAPHICS

Each raster *file* has a color table *file* that associates colors with categories of that map. You may redefine the color for each category. GRASS uses either a fixed state or floating mode to represent colors. In the first case each of multiple maps on the screen is displayed with its original colors as defined in the color table *file* belonging to the map. In the second case, the display commands ignore these color table files and display all maps according to the table of the currently active monitor. Usually, this is not what the user really wants, but by setting the colormode to a floating state, the user may watch the results of changing a color table interactively.

d.colors allows the user to interactively change the color table of a *raster map* displayed on the *graphics monitor* (if *d.colormode* mode=float had been issued before). The actual *editing* is done within a *menu* in the GRASS command window. The results may be checked on the currently active *graphics monitor*. A shortcut to the lengthy process of creating a color table interactively is the program *r.colors*. It uses predefined palettes which are stretched over the whole range of *category* values.
Examples:

r.colors spot grey
r.colors elevation *colorrules* green brown yellow *end*

III.1.5.2 DISPLAY OF RASTER, *VECTOR* AND *SITE* DATA

Once a *graphics monitor* is established, raster, *vector* or *site* data may be displayed in the currently active display frame (remember: unless otherwise defined, the display frame covers the full size of a *graphics monitor*). With *raster data* you may also perform a visual overlay, that is, non-zero values are drawn only. *Vector* maps have no color table associated with them. Their default color is white, you may however use 9 other colors to display *vector*s (one *vector* map can't have multiple colors though). The same holds true for *site map*s. *Sites* are represented by icons of a size to be specified.
Examples:

d.rast map=elevation
d.rast -o map=contours
d.vect map=railways *color*=violet
d.sites map=stations *color*=magenta *size*=8 *type*=diamond

III.1.5.3 VISUAL QUERY OF MAP CONTENTS

Some functions allow to query the screen, displaying some information on the geographic *location* of the cursor *d.where*, the *category* content of (multiple) *raster map*(s) *d.what.rast* as well as their labels *r.what* or the *category* content of (multiple) *vector* map(s) *d.what.vect*. If the map is not in *UTM*, *d.where* needs a specification of the spheroid to calculate the correct position. Up to 15 maps may be queried at once while using *d.what.rast* or *d.what.vect*.

With *d.measure* you use a *mouse* to trace the out*line* of an area. This will give you the size and perimeter of that specified portion of the map. *d.rhumbline* and *d.geodesic* are used to display a *line* of constant angle (rhumbline) or the shortest distance (geodesic) between two geographic *point*s. To display a grid in units of the current map *projection* (degree or meters) you would call *d.grid*.

III.1.5.4 CREATING ON-SCREEN MAPS

A map is more than just displaying the extent of some *cell* categories within a given geographical extent. It should include a legend, a title, a *scale* bar, a north arrow and possibly some text and/or additional graphics. While the suite of tools in the MapGen package (outside GRASS, but considered as "garden" around it) gives you all of this plus many more sophisticated functions, there are still a few programs within GRASS that deliver similiar functionality. First to be mentioned, there are all the *p.* tools, which concern the *hardcopy* output, and second - belonging into this section of display commands - there are those programs that display this kind of information on the screen, however, without saving it (to a file).

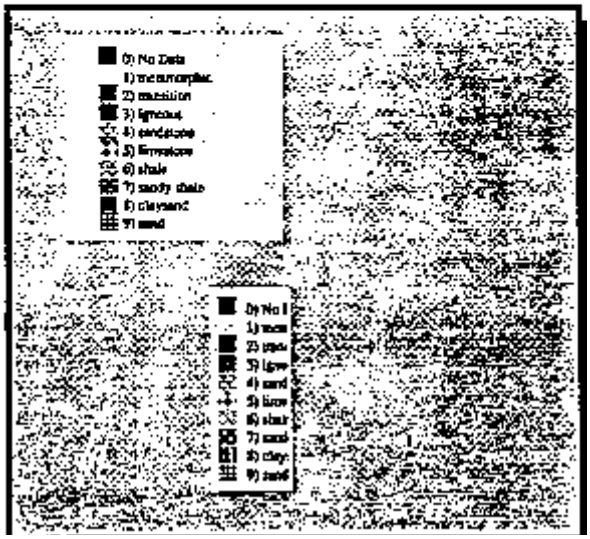


Figure 36 "Example of *d.legend*"

The *d.legend* command places a *legend* within the currently active window. It adjusts its size automatically to the vertical dimensions of this window. If the *category* names are too long or the window is odd sized, you have to adjust the *legend* width with the lines option. *d.scale* allows to place a *scale* bar as well as a north arrow. Simple *vector* (turtle) graphics can be displayed with the commands *d.graph* and *d.mapgraph*. The only difference between the two are the coordinate systems used. While *d.graph* works with percentages of screen width and height, d.mapgraph accepts *projection* units of the displayed raster layer. The graphics commands will usually be placed in an extra *file* and their output redirected with the *UNIX* '<'. This way, simple company/institute logos can be placed on screen.

Several commands allow to display text anywhere on screen. *d.label* places a single *line* of text that it prompts for. *d.labels* does the same, but allows up to four lines. Also, it allows -just as *d.paint.labels* - the text to be saved to a paint *file* (paint files are interpreted when a *hardcopy* output is created). Finally there is *d.text*, which does not requiere a *mouse* for positioning the text, but expects a *line* number. This allows for creating justified columns. The title is treated somewhat special, for the command *d.title* (which should be *UNIX* piped '|' into *d.text*) creates an input *file* which contains the raster *map layer*'s name, as well as *MAPSET*, LOCATION, *geographic region* boundary coordinates and *cell resolution*. The default font type is 'romans'. This can be changed with the command *d.font* to any of 18 further Hershey fonts. A shell script *show.fonts.sh* displays a sample of all fonts.

Three commands allow to display *point* data or symbols. *d.points* and *d.sites* place simple shaped markers at the positions specified in a sites *file* (d.sites) or in an additional *UNIX file* (*d.points*). The latter allows to place these markers outside the current *geographic region*. Marker types are 'x' (default), '+', diamond and box. More complex symbols can be created with the command *p.icon*. The results are saved in special files in an icon directory. With *d.icons*, these symbols can be placed on screen. The positions are read either from standard input (keyboard) or from a special *point*s file, similar as with *d.points*.

II.1.5.5 GRAPHICAL MANIPULATION OF *RASTER DATA*

Two commands allow to merge up to three raster *map layer*s. *d.rgb* is used to combine seperate color bands of red, green and blue. *d.his* creates images that ideally resemble an air photo of the mapped terrain. For the technical intricasies of these commands, the user is referred to the GRASS Reference manual. A display of the possible color distribution of a user-specified raster *file* can be achieved with *d.histogram*. It plots a pie or bar chart of the *category value* distribution. Similiar statistical information can be diplayed with a shell script called *d.correlate.sh*. The color table of a given *raster map layer* can be displayed with *d.colortable*. This command fills the entire active frame unless the user specifies a different sized matrix for the colors to be displayed. The thirteen primary named colors, as used by GRASS can be displayed with the shell script *show.color.sh*.

Another quite impressive visual manipulation can be performed with the program *d.3d*. This program drapes an *image* over a three-dimensional display of another *raster map layer* (usually signifying elevation). Viewpoint, view direction, vertical exageration and field of view allow for manifold manipulations. The performance of this program depends completely on the line frequency / *resolution*, therefore, the user should start fizzling around with a coarse *resolution* (that is computational less intensive) to find the appropiate parameters.

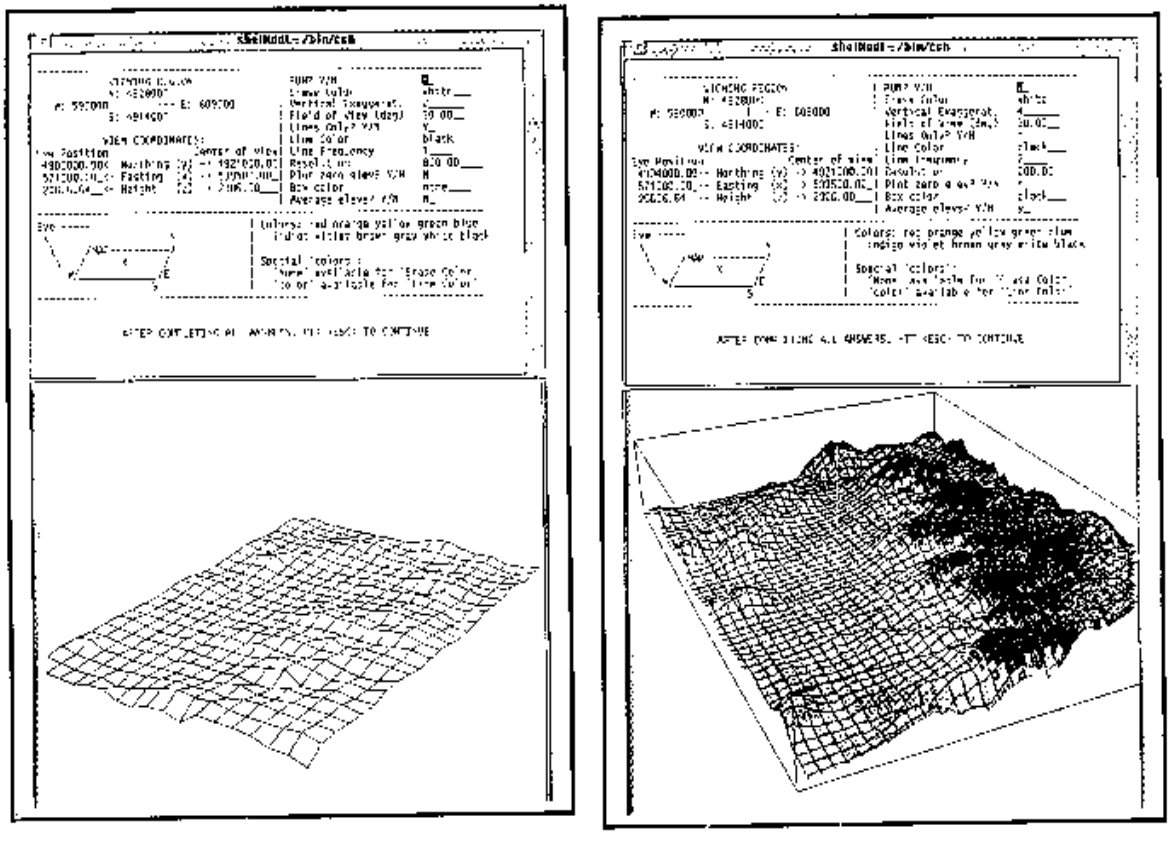


Figure 37 "Examples of *d.3d*"

III.2 HARDCOPY MAP PRODUCTION MODULES

You should be prepared, that the *p.* commands provide you with a limited map production functionality only. If you want to create full-featured professional looking maps, you should refer to the MAPGEN package that is part of GRASS' garden.

The GRASS paint, or map printing, system provides a flexible tool to produce presentation quality outputs from GRASS data using different colors and patterns, multiple *vector* overlays, user-defined scales, *point* data depiction with user-created or default icons, and user-defined labels and legends. The map printing process parallels that for map display. Typically, however, printing of output is done only after all map analysis is completed. The user should design a map's contents, select a paint (device) *driver* to which output will be sent, and then print the map.

After selecting a printer or plotter with *p.select* the user should print a color chart for the printer currently selected. This table printed with *p.chart* allows to adjust screen colors so that they look like the *hardcopy* output. The latter is done with *p.colors*. While the *d.* commands work with percentages of red, green and blue, *p.colors* modifies a *raster map* by assigning colors based on the printer color numbers. *p.icons* is used to create icon files which can be called by d.icons to place them at sites. Similarily, *p.labels* creates *label* files that contain all information about the placement of labels (the text itself, its font, size, color, justification, location, etc.).

The main *p.* program, however, is *p.map*. It allows to graphically combine a *raster map* with any number of *vector* maps, *site* data and text labels. There are 20 parameters that the program will prompt for if you don't specify them in an input file. For a complete description of these, please refer to the technical reference. A few ought to be mentioned though, be it that they are especially important or that they should be treated cautiously. An example for the latter is *d.colortable*. If the *raster map* has many categories (e.g. a *LANDSAT* image) the resulting color table would be very long, possibly several pages! The parameter region places the out*line* of a smaller *geographic region*. In most cases though, the user should prepare a special *file* that is read into *p.map*.

The following commands are used to print or save the output produced by the various GRASS print commands:

- *p.map* - *p.ppm* - *p.screen*

In addition to *hardcopy* output devices (printers), a 'preview' *driver* also exists to send print output to a user's graphics display monitor. As to the time this is published, the following printer/plotter device *driver*s were available:

preview	on-screen display of the map
ppm	map.ppm *file* that can be used as input to the suite of conversion routines in the pbm package (GRASS garden)

Tektronix	Diabolo	
EPSON	Genicom	in contrast to the first two, these are device *driver*s that
Shinko	Versatec	work the way a (*UNIX*) user would expect.
Xerox		

IV DIGITIZING AND OTHER FORMS OF DATA INPUT INTO GRASS

To be used by GRASS programs, data must be digitally stored in a GRASS *database*. Data can be entered into a GRASS data base in one of several ways:

- *Hardcopy* data (like paper maps) may be either digitized or scanned into GRASS's *vector file* format;
- Data already available in another digital format (like ARC/INFO data) may be converted into GRASS's *digital data file* format, and stored in a GRASS *database*;
- GRASS analysis programs can be used to create new GRASS data from data already stored in GRASS *digital data file* format.

The GRASS programs described in the map development section are designed for the input, manipulation, and adjustment of data, primarily obtained from sources outside of GRASS.

IV.1 DIGITIZING

IV.1.1 PREPARING A MAP FOR DIGITIZING

Paper maps can hold a great amount of information. This information can be much more easily manipulated in a computer *database*. To get map data into a GRASS *database*, map data (analog data) can be converted into *digital data* by tracing relevant *map feature*s using a *digitizer*. The *digitizer* converts map data into a GRASS *vector format*.

Frequently, map data cannot readily be digitized directly from original paper source maps. Paper maps will generally contain many types of information. The person that is digitizing must know exactly what information is to be digitized. It is therefore generally necessary for the user to prepare map(s) containing only the information that is to be digitized before beginning to digitize.

The user should have the following items before beginning to digitize:

- a stable medium, i.e., mylar, plastic, etc. on which the features to be digitized appear clearly and accurately
- at least 10 marked *point*s on the medium corresponding to *database* (e.g., *UTM*) coordinates
- a region in *database* coordinates defining the *area* being digitized
- a region in *database* coordinates defining the *area* of the resultant *raster data* file
- a map of the *area* being digitized near the *digitizer* readily available for reference
- masking or drafting tape to secure the medium
- a planned final region and *resolution* for the *raster data* file.

The GRASS *v.digit* program is used to *digitize* data in GRASS *vector format*. Unlike most other GRASS programs, the *v.digit* program does not(!) use the boundaries and *resolution* of

the user's current *geographic region*. Instead, the user establishes map boundaries (the "digitizing region") when entering *v.digit*, and modifies the map's *resolution* by setting the *digitizing threshold* (see the following section on tresholds) in the *v.digit* Customize Menu.

When the user begins to *digitize* a *file* with *v.digit*, the program asks the user to define the boundaries of the *geographic region* being digitized. Once the edges of this region are established, *v.digit* will recognize only data digitized by the user within these boundaries. This region is referred to as the "digitizing region".

Users can set different digitizing regions for each map digitized. Typically, users might elect to set these boundaries equal to those of the default region; however, this is up to the user. The current *geographic region* setting stored in the user's "WIND" *file* does not affect the user's digitizing session. Conversely, digitizing region setting does not affect the user's current *geographic region* setting.

Inside *v.digit*, the user can graphically *zoom* into different areas within the digitizing region to aid digitizing. These *zoom*-in areas are referred to as "working frames". They do not affect which areas can be digitized, but only affect which areas of a map will be seen on the *graphics monitor*. Establishing working frames in *v.digit* does NOT alter the user's digitizing region nor the user's current *geographic region* settings.

Note: The current *geographic region* settings become important after all digitizing is done, when the user converts a digitized *vector* map to *raster format*.

Although all of the commands listed below are used to develop new map data, *v.digit* is mainly used to develop new map data from non-digital sources. *v.digit* is a robust, highly *interactive* data development program used for *vector digitizing*, *editing*, labeling and converting *vector data* to *raster format*. The user can *digitize* data using either a *digitizer* hooked-up to his system, or GRASS's on-screen digitizing capability.

- *v.digit*
- *v.mkcontour*
- *v.mkgrid*
- *v.mkquads*
- *v.support*
- *v.patch*
- *r.random*
- *r.patch*
- *s.menu*

Frequently, (due to the size of digitizing *tablet*s) users *digitize* data in relatively small chunks which must be "patched" together before data are analyzed. The GRASS commands listed below can be used to edit and patch *digital data*.

- *v.patch*
- *v.prune*
- *v.spag*
- *r.patch*
- *r.mapcalc*

A variety of supporting information is needed for each GRASS *map layer*. For data stored in GRASS *vector format*, information on the *location* of *vector*s, the *file topology*, *vector category values*, and *category labels* are stored in separate files. For data stored in GRASS *raster format*, additional files are also stored to hold information on *category* colors, the map's development history, etc. The programs below help the user build necessary GRASS *support files* for raster and *vector data*.

- *r.support*
- *v.support*

Many GRASS commands automatically generate necessary *support files*. Refer to specific GRASS program manual entries to learn when the user must explicitly create *support files*.

IV.1.2 THRESHOLDS

The *g.region* program is used to set the *resolution* of data stored in *raster data* format. However, when digitizing *vector file*s in GRASS, and when converting *vector file*s to other (e.g., raster) data formats, the data "*resolution*" is based on the digitizing (or map) *threshold* and the *node snapping threshold*. These *threshold* levels are explained briefly below.

"Digitizing threshold" - Determines which *point*s are saved during digitizing. Expressed, in map distance, as the width of the corridor surrounding each *point* outside of which digitized *point*s will be saved.

"Map Threshold" - Expresses the *digitizing threshold* in real world units. The map *threshold* is expressed in units on the ground, as the width of the corridor surrounding each *point* outside of which digitized *point*s will be saved.

"Snapping threshold - The radius within which *vector* nodes will be "*snap*ped" and said to be at the same geographic coordinate location. Used during digitizing, when importing *vector file*s to GRASS *vector format*, and when converting *vector file*s to GRASS *raster format*. Generally expressed as a map distance, but may also be expressed as ground distance.

IV.1.2.1 THE DIGITIZING THRESHOLD

Digitizing using the GRASS *v.digit* program may be done in either *point* or stream mode. In *point* mode, a *point* is collected at each *location* specified by the operator. In stream mode, the operator traces a continuous line, and *point*s are collected automatically as the *line* is traced. When digitizing in stream mode, the *digitizer* picks up many *point*s, often more than are needed to adequately depict a line. To limit the total number of *point*s in a data file, a *pruning* of the original *point*s occurs before the *point*s are saved in a data file. The *digitizing threshold* is used to determine which *point*s are saved and which are thrown out.

The *pruning* process occurs during digitizing each time the user accepts a line. It is achieved by setting up a sort of corridor the width of the *digitizing threshold*. The direction of this corridor is determined by the first two *point*s that were collected by the digitizer. All subsequent *point*s are examined to determine whether they fall within the corridor limits. The first *point* which lies outside the corridor effectively establishes a new direction for the line. The program saves the original first *point* of the *line* segment and the last *point* that is found to be inside the corridor. The intervening *point*s are discarded, as they are unnecessary to define the *line* segment. A new corridor is now established between the second saved *point* and the *point* collected immediately after it and the process continues.

As a result of this process, the curved portions of a *line* will contain more *point*s, while a straight portion will have fewer. Theoretically, the user could do what is essentially *point* mode digitizing in stream mode. In cases where the straight *line* segments are short and interspersed with curved lines this could be done. It is not recommended that long straight *line* segments be done in stream mode, however. They are done more efficiently in *point* mode. The default *digitizing threshold* is 0.03 inches, which is equivalent to about 0.75 mm. This level works

quite well when digitizing curved lines of a fairly complex nature. If the user is not satisfied with the default *threshold*, it may be changed using the *v.digit* "Customize" menu. A larger *threshold* (e.g., 0.05 inches) will result in fewer *point*s being saved; a smaller *threshold* (e.g., 0.02 inches) will increase the number of *point*s saved. The default *digitizing threshold* is automatically reset at the beginning of each "v.digit" session.

IV.1.2.2 THE MAP THRESHOLD

The distance indicated by the *map threshold* is the same as that indicated by the *digitizing threshold*. However, it is measured in map units (distance on the ground) and thus takes into account the map's scale. For example, suppose a map is being digitized using the *UTM* coordinate system. Map units would thus be meters. The default *digitizing threshold* of 1 mm is equal to 0.001 meters. If the map *scale* of the map being digitized is 1:25,000, the map *threshold* would be 0.001 multiplied by 25,000, or 25.0 meters. A map *scale* of 1:50,000 would give a default map *threshold* of 50.0 meters. The map *threshold* may not be changed directly. It changes automatically with a change in the *digitizing threshold*.

IV.1.2.3 THE SNAPPING THRESHOLD

Map data are digitized in *vector format*. The edges of *map feature* are traced with a digitizer. These edges consist of a series of *line* segments (*arcs*) connected at their end-*point*s (*nodes*). The geographic coordinates of these *line* segments and *nodes* are stored in a *vector data* file. If the user is digitizing the perimeter of a closed surface (e.g., an installation boundary), all of the *line* segments representing the perimeter must *join* together in the data file. A *line* segment that is attached to a *node* is said to be "*snapped*" to that *node*. To *join* two (or more) *line* segments, the user must "*snap*" these lines to the same *node point*.

A *snapping threshold* determines what digitized *point*s are assigned to the same node. If two *nodes* fall within the *snapping threshold* distance of one another, they will be forced to use the same *node point*. The result will be a single *node* rather than the original two, and the two *line* segments previously connected to two different *nodes* will be joined at the single *node* retained. The *snapping threshold* makes it possible for those digitizing to connect lines without possessing superhuman abilities. The machine will assign any two *point*s falling within the *snapping threshold* to the same geographic coordinate *node point*, even if they were (unintentionally) entered at very slightly different spots.

IV.1.2.4 THE SNAPPING OF NODES

The geographic coordinates associated with digitized *point*s (defining *line* segments and nodes) will be added to a GRASS data *file* in the order in which they are digitized. If two *node point*s fall within the geographic *snapping threshold* of one another, the *node* that appears second in the data *file* will *snap* to the one that is listed first. In GRASS, the default *snapping threshold* is 1 map millimeter (in terms of distance on the map). The *snapping threshold* can also be expressed as actual ground distance in *database* mapping units (e.g., in meters for a *UTM database*). When expressed in *database* mapping units, the number given for the *snapping threshold* will be dependent on the *scale* of the map.

IV.1.2.5 SETTING THE SNAPPING THRESHOLD VALUE

The default *snapping threshold* value should suit most occasions; however, the user can modify the *threshold* if necessary.

Digitizing in GRASS:
When digitizing, the *snapping threshold* may be changed using the *v.digit* "Customize" menu. This method is useful when two nodes on a map being digitized need to be closer together than the *snapping threshold* allows. Lowering the *snapping threshold* should fix this. In such a case, the user should leave the *snapping threshold* at the lower value when the completed digitized *file* is saved.

Importing Data Files into GRASS:
The *snapping threshold* may also need to be modified when importing a *vector file* produced by other software. Since different software handles *snapping* differently, it is possible that *line* segments in the *file* to be imported are not really connected to one another. GRASS requires that any shared nodes have exactly the same geographic coordinates. If the user is aware that nodes in to-be-imported *file* are not yet *snapped*, the user can force *snapping* to occur within a default distance while *v.import* is run.

If the user is unaware of the status of the *file* to be imported, it is recommended that the user first run *v.import* without requesting that nodes be *snapped*. If, after *import* to GRASS, the *file* appears to contain unsnapped *nodes*, the user can run *v.support* to correct this. Here, the user is offered the option of using a *snapping threshold* other than the default value.

Importing Scanned Files into GRASS:
It has been observed that with machine-produced *vector file*s (i.e., those electronically scanned), it is possible to have lines shorter than what might be expected from a manually digitized file. In this case, the *snapping threshold* should be set lower than the default when running *v.support*, to avoid losing such lines.

A NOTE OF CAUTION:
v.support allows the user to assign a *snapping threshold* other than that used during digitizing. Users who assign a higher *snapping threshold* than this should be careful -- a higher *snapping threshold* may *join line* segments that the user did not intend to join, thereby corrupting the data file!!!

See "Cartographic Issues in the Development of a Digital GRASS Database", (RUIZ and MESSERSMITH 1990) from which the information in this section was obtained, for further information on map development issues.

IV.2 DATA CONVERSION IN GRASS

To be used by GRASS programs, data must be stored in GRASS's *digital* format in a GRASS *database*. Often, this process requires that users convert data into the GRASS format they desire.

"Data conversion" may involve the:

(1) conversion of data in ASCII format to/from binary format,
(2) conversion of data among GRASS raster, vector, and sites data formats,
(3) conversion of data in GRASS data formats to/from other data formats,
(4) conversion of data from one *map coordinate system* to another,
(5) conversion of data in one map *projection* to another.

The latter two are dealt with in chapter II. Some GRASS programs operate only on data in *raster format*, some only on *vector data*, and some only on *site* data. Typically, GRASS raster and *vector data* must be in binary format, while *site* data (in site_lists format) must be in ASCII format, to be usable by most GRASS programs. When the user wishes to use a GRASS program on data that are not in the *file* format Demanded by that program, the user must convert the data to the appropriate format using GRASS data conversion (import/export) programs.

IV.2.1 MOVING DATA TO ANOTHER COMPUTER (ASCII/BINARY CONVERSION)

One form of data conversion is the conversion of data in ASCII format to and from binary format. American Standard Code for Information Interchange (ASCII) is a standard seven-bit character code by which information is stored and transmitted in a computer or a data transmission system. The 128 characters in the ASCII character set are produced by combinations of the shift and control keys with the other keys. All its characters are of a fixed, seven-bit length. Files in ASCII format can be read by the user.

In binary format, data are represented as a series of 0's and 1's. A binary digit (0 or 1) is called a "bit" of information. Files are put in binary format to put them in a code that the computer can understand. Binary format is machine-readable. Typically, users cannot read files in binary format.

The commands listed below are useful to importing and exporting data to and from GRASS and when converting data among various *file* formats. They perform ASCII/binary data conversions. Commands with the extension ".in.ascii" produce binary files from ASCII input. Commands with the ".out.ascii" extension produce ASCII output from binary input.

GRASS programs that convert ASCII data to Binary format:	GRASS programs that convert Binary data to ASCII format:
r.in.ascii	*r.out.ascii*
s.in.ascii	*s.out.ascii*
v.in.ascii	*v.out.ascii*
v.import	*v.import*

IV.2.2 RASTER/VECTOR/SITES CONVERSIONS

The commands listed below are used to convert data among GRASS raster, *vector*, and sites data *file* formats. Such conversion is needed when the user wishes to run a GRASS program on data that are not in the format expected by the program.

Raster Conversion Programs:	*Vector* Conversion Programs:	Sites Conversion Programs:
r.line	*v.to.rast*	*s.in.ascii*
r.poly	*v.to.sites*	*s.out.ascii*
r.thin		
r.in.sunrast		

Typically, linear and *polygon* data are entered in *vector format*, and converted to *raster format* for analysis. Often, *point* data are entered in site_lists format and analyzed either in this format or after conversion to *raster format*. Data can be displayed and printed in any of these three *file* formats.

Map data that are *scanned* electronically are usually scanned as *raster data* at a very fine *resolution* (around 100 microns). The *raster data* are then converted to GRASS *vector file* format. When done properly, the degree to which the *digital data* are different from the original lines is indiscernible. Such files must be checked carefully, however, for errors introduced because of mislabeled *polygons*, unsnapped *nodes* or some misunderstanding on the part of the persons doing the *scanning* as to what exactly was required.

Once made digital, *vector data* must be run through the GRASS program *v.to.rast* in order to be converted to *raster format*. Any *vector* to raster conversion will result in the zigzag lines characteristic of *raster data* (see figure 38).

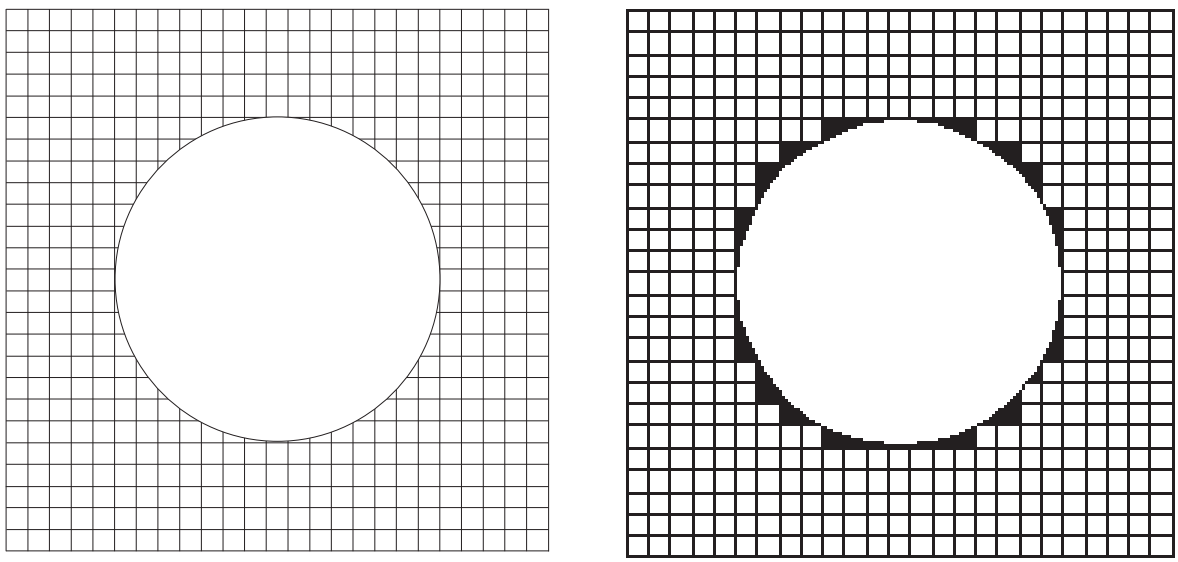


figure 38 "Comparison of *Vector* Map with its Raster-Converted Equivalent"

IV.2.3 GRASS TO/FROM OTHER FILE FORMAT CONVERSIONS

Increasingly, digital spatial data is becoming available in a wide variety of data formats. It is generally much more efficient to convert existing *digital data* to the format needed by your system, than to re-*digitize* such data.

The commands listed below are useful in the import/export process. These commands convert GRASS files to and from such other *file* formats as ARC/INFO, USGS DLG-3 Optional,

DXF, and SUN *raster format*. Command name extensions take the form ".in.format" and "out.format", where: ".in." designates that data is being imported to GRASS from another format; ".out." designates that data is being exported from GRASS to another format; and "format" designates the other (non-GRASS) data format conversion type.

Importing other data to GRASS *vector format*:	Exporting GRASS data to other systems' formats:
v.in.arc	*v.out.arc*
v.in.dlg	*v.out.dlg*
v.in.dxf	*v.out.dxf*
r.in.sunrast	
Gen.Maps	

If the user wishes to export data in GRASS raster or site_lists data formats to another system's format that usually works with *vector*s (such as ARC/INFO), these files must be converted to GRASS *vector format* before export.

IV.3 EXTRACTING DATA FROM MAGENTIC TAPES

IV.3.1 IMAGERY DATA

(see also chapter VII "*Image* Processing")

Several GRASS commands are available to extract specific forms of satellite *imagery* data from magnetic tape. They extract *LANDSAT M*ulti-Spectral Scanner (*MSS*) data, Thematic Mapper (*TM*) data and other (e.g., *SPOT*) satellite data, as well as header information, from tape. Procedures for extracting these data from tape and importing them into a GRASS data base are described in the *"GRASS Imagery Tutorial"* (Harmon and Shapiro, 1990) available via Internet from the moon.cecer.army.mil server.

Remotely-sensed *images*, such as satellite data and aerial photographs, are often available in a digital format. The data in these images can be interpreted using *image processing* techniques that classify *image elements* based on the spectral reflectance values of individual *image* pixels or on user-identified pixel *grouping*s.

GRASS integrates a *GIS* with *image* processing capabilities. This allows *image* data to augment a GRASS *database*, and allows *image* classifications to be guided by known *elements* in other maps in the *GIS database*.

Processing of remotely-sensed images typically involves:

1. extracting data from their magnetic storage media,
2. entering extracted data into a GRASS data base,
3. classifying pixels and *elements* in the image,
4. rectifying the *image* to a *map coordinate system* or projection,
5. correcting and enhancing the image.

GRASS commands commonly used to perform *image* processing are discussed in chapter VI. Here, only those GRASS imagery commands used to extract imagery data from magnetic tape media are discussed.

Once imagery data are obtained, they must be extracted from the magnetic tapes on which they are stored.

Specific procedures for extracting this data from tape and importing it into a GRASS *database* are described in the document *"GRASS Imagery Tutorial"*.

When ordering *digital data*, the user must specify the format of the tape and data he wishes to receive. Issues are briefly reviewed here. Refer to company data ordering instructions for specific information.

IV.3.1.1 DATA CORRECTIONS, MAP PROJECTIONS AND DATA RESAMPLING

Satellite data can be geometrically and radiometrically corrected and resampled, and put in a standard, distortion-free map projection. Radiometric corrections compensate for variations in satellite sensor gain (affecting *image* brightness). Geometric corrections are made to compensate for the earth's rotation, spacecraft altitude and sensor variations.

Companies making available *digital imagery* and other data supply free indexes showing areas of the earth for which data are available. The user must balance data acquisition and storage costs against the extent (ground coverage) and spatial and spectral *resolution* of the data desired. Users might request that companies shift standard scenes to include their data, generally for an added fee.

- *LANDSAT TM* data have roughly a 30 meter *resolution*.
- *LANDSAT MSS* data have roughly an 80 meter *resolution*.
- *SPOT* 1-*band* panchromatic data have roughly a 10 meter *resolution*.
- *SPOT* 3-*band* data have roughly a 20 meter *resolution*.

Generally, it is advisable to order data with no more than 10% cloud cover. Scenes can generally be reviewed before ordering to ensure that clouds do not obscure vital portions of your image. In most regions, the season of the year at which imagery has been taken will have an drastic impact on landcover and therefore on spectral responses of ground (earth surface). Consider the application for which your imagery will be used when selecting a date for imagery. If you wish to detect landscape changes occurring over multiple years, attempt to order satellite data from the same time (month) of each year.

Different satellites detect different spectral bands. Different *image* bands are useful for different applications. Panchromatic images are black/white images with 1 *image band*. Multispectral images are color images with multiple *image bands*.

- *SPOT* panchromatic data contain 1 *image* band.
- *SPOT* multi-spectral data have 3 *image* bands.
- *LANDSAT* panchromatic images contain 1 *image* band.
- *LANDSAT* Thematic Mapper (*TM*) data have 7 *image* bands.
- *LANDSAT* Multi-Spectral Satellite (*MSS*) data have 4 *image* bands.

IV.3.1.2 TECHNICAL INTRICASIES

Multi-*band* data can be ordered in either *band* sequential (BSQ) or band-interleaved (BIL) formats, at either 1600 bpi or 6250 bpi. In BIL format, the different *image bands* are stored together (interwoven) for each pixel in the image. In BSQ format, *band* data are separated: data for all *image* pixels in one *image band* are stored together, followed by all data pertaining to a second *band*, etc. GRASS can handle either BIL or BSQ data. See the *"GRASS Imagery Tutorial"* or the programs listed under the Image Processing menu for further information.

The *UNIX* commands "dd" and "tar" are useful for extracting data from other storage media (e.g., CD-ROM, floppy disk, streamer tapes). These commands can also be used to load GRASS (and other) data onto other storage media for export to other systems.

"dd" copies a user-named input *file* to a user-named output location. The user can copy files from (or to) tape by specifying a tape device as the input (or output) *file* name. The user can specify input and output block sizes, and perform *file* conversions between ASCII and EBDIC formats, fixed and variable length records, and upper and lower case, and perform other functions.

The *UNIX* "tar" command archives and extracts multiple files onto a single tar-*file* archive, called a tarfile. The user specifies the *file* or directory names that specify which files are to be archived or extracted.

If the user specifies a directory name, tar recursively extracts or archives all of the files and subdirectories of that directory. The user can specify a specific blocking factor to use when extracting or archiving data. "tar" will automatically determine the tape blocking factor when extracting data. "tar" can also follow symbolic links. One useful function performed by tar is to list the contents of a tarfile.

Users can read about the specific syntax of "dd" and "tar" in the "*UNIX* Reference Manual". On most systems, the manual is available on-line. Users can read about a specific *UNIX* command by typing "man command_name", where "command_name" is the command the user wishes to read about. For example, the user can read about the *UNIX* "dd" command by typing "man dd".

V ANALYZING DATA IN GRASS

V.1 OVERVIEW OF THE ANALYZING CAPABILITIES OF GRASS

V.1.1 ANALYZING SITE DATA

Site data can be analyzed in GRASS either in their site_lists format or after conversion to *raster data* format. The following GRASS programs are used to analyze site_list data:

s.db.rim	Interface to the RIM *database* management system
s.in.ascii	Imports *site* data in ASCII format
s.menu	accesses and manipulates sites data
s.out.ascii	Exports *site* data into an ASCII file
s.surf.idw	Interpolates a surface from *site* locations

Most GRASS analysis programs operate on *raster data*. GRASS programs exist that convert GRASS data between raste*r, vector, and site formats*.

V.1.2 ANALYZING VECTOR DATA

Many GRASS programs operate on data stored in *vector format*. These *vector* programs perform data development, data import/export, and data conversion functions. However, most GRASS analysis programs operate on data stored in *raster data* format. Users must convert *vector data* to *raster format* to use these raster analysis programs. Refer to chapter IV for more information on data conversion between GRASS *raster, vector, and sites formats*.

V.1.3 ANALYZING RASTER DATA

GRASS was designed as an *image* processing and map analysis system. Most GRASS analysis programs operate on *raster data*. GRASS *raster data* analysis functions can be employed for such tasks as:

1 two- and three-dimensional manipulations,
2 single *cell* map reporting functions,
3 multi-*cell* coincidence tabulations and comparisons,
4 *neighborhood analysis* and filtering,
5 regional analysis and filtering,
6 proximity analysis,
7 Boolean *overlay*functions, and
8 terrain analysis functions.

V.1.3.1 2-D AND 3-D MANIPULATIONS

GRASS programs can be used to perform two- and three-dimensional data manipulations on *raster* data. These functions are useful for modeling iterative interactions across a landscape.

r.basins.fill	creates a *raster map layer* of watershed subbasins
r.drain	traces a flow through an elevation modelon a raster *map layer*
r.surf.contour	surface generation program
r.watershed	watershed basin analysis program
r.surf.idw	surface interpolation program
r.surf.idw2	yet another surface interpolation program

V.1.3.2 SINGLE CELL MAP REPORTING FUNCTIONS

In their *raster format*, data are stored on a grid having rectangular *cells* of uniform size. Each *cell* within the grid can be assigned a data value.

The programs listed below report information about a single cell, in one or more raster *map layer*s. They are useful for reporting the type of data occurring at a particular geographic location. These programs may also serve other functions; some also provide information on surrounding cells.

d.what.rast	allows *interactive* query of the *category* contents of multiple maps
r.stats	generates *area* statistics for raster *map layer*s
r.describe	prints short list of *category values* found in a raster *map layer*
r.what	queries raster *map layer*s on their *category* values and labels
r.info	outputs basic information about a raster *map layer*
r.volume	calculates the volume of data "clumps"
r.report	reports statistics for raster *map layer*s

V.1.3.3 MULTI-CELL COINCIDENCE TABULATIONS AND COMPARISONS

The programs listed below report back information about multiple cells in one or more raster *map layer*s, in a way useful for making correlations among data. These programs may also serve other functions.

d.what.rast	allows *interactive* query of the *category* contents of multiple maps
r.stats	generates *area* statistics for raster *map layer*s
r.coin	tabulates mutual occurence of categories for 2 *raster map*s
r.what	queries raster *map layer*s on their *category values* and *labels*
r.report	reports statistics for raster *map layer*s

V.1.3.4 NEIGHBORHOOD ANALYSIS AND FILTERS

The user may wish to model interactions across a landscape, or to remove systematic distortions occurring across a map or image. The user may also wish to eliminate "noise" from map data in order to make the basic results more visually apparent. Such applications can be approached by filtering data. The user can build custom filters to make the value of one *cell* a (user-defined) function of the values in the surrounding cells.

The programs below are used in *neighborhood analysis* and filtering.

r.basins.fill	creates a *raster map layer* of watershed subbasins
r.mapcalc	a very powerful "map calculator", see also chapter VI.3
r.buffer	creates buffer zones around cells with non-zero values
r.neighbors	makes each *cell* value a function of the values of its neighbors
r.cost	creates a map of cumulated costs for moving between cells
r.slope.aspect	generates slope and *aspect* layers from an elevation layer
r.clump	recategorizes a *raster map layer* by *grouping* cells
r.thin	thins line-like features to be used as input for *r.line*
r.drain	traces a flow through an elevation model on a raster *map layer*
r.watershed	watershed basin analysis program
r.grow	creates a *layer* where contiguous areas are grown by one cell

V.1.3.5 REGIONAL ANALYSIS AND FILTERS

GRASS can also be used to analyze and filter data "regions". Neighborhood filters modify each *cell value* as a function of the *cell values* found in its (square) neighborhood. Regional filters make each *cell value* a function of the values found within its (user-defined) region/clump of cells. A region can be defined as a contiguous group of cells having the same *category value*. Such filters are useful for *image* analysis and other applications.

Listed below are GRASS programs which can be used for regional analysis and filtering.

i.zc	zero-crossing edge detection function for *image* processing
r.clump	recategorizes a *raster map layer* by *grouping* cells
r.average	finds the average of values in a values map within user-defined areas
r.volume	calculates the volume of data "clumps"

V.1.3.6 PROXIMITY ANALYSIS

GRASS can be used to examine the proximity of certain map categories from others. Often, this is useful when determining which geographic areas warrant further analysis. For example, if investigating potential nesting sites of species X, the user might restrict his analysis to areas within a 300-meter radius of cells containing relevant species habitat.

The *r.buffer* program allows the user to establish buffer zones around cells with non-zero *category values*. The *r.reclass* program can be run prior to *r.buffer* to identify particular *cell values* of interest.

r.buffer	creates buffer zones around cells with non-zero values
r.reclass	creates a new map *layer* according tothe user' re*classification* of another map

V.1.3.7 BOOLEAN OVERLAY FUNCTIONS

GRASS can perform basic Boolean *overlay* functions, involving data union ("or" operations), intersection ("and" operations), and exclusion ("not" operations). The GRASS commands listed below are used to do this (although they may also serve other functions).

r.combine	allows *category* values from several *map layer*s to be combined
r.cross	creates a cross product of *category* values from multiple *map layer*s
r.infer	rule-based derivation of *category* values
r.mapcalc	a very powerful "map calculator", see also chapter VI.3
r.weight	a *raster map overlay*program
r.weight.new	as above, but using a different algorithm

V.1.3.8 TERRAIN ANALYSIS

Elevation data are crucial to any *GIS*. A number of GRASS commands were designed specifically to address terrain analysis issues related to water movement, development of ballistic trajectories, and computation of slope and *aspect*.

Although the GRASS commands listed below were designed to analyze terrain data, they can be used to analyze any other theme.

r.basins.fill	creates a *raster map layer* of watershed subbasins
r.slope.aspect	generates slope and *aspect* layers from an elevation layer
r.cost	creates a map of cumulated costs for moving between cells
r.traj	calculates the way of a trajectory
r.drain	traces a flow through an elevation modelon a raster *map layer*
r.volume	calculates the volume of data "clumps"
r.los	line-of-sight analysis program
r.watershed	watershed basin analysis program
r.surf.idw	surface interpolation program
r.surf.idw2	yet another surface interpolation program
r.surf.contour	surface generation program
d.rast.arrow	places little arrows designating the direction of flow of an elevation layer

See also: *d.3d*, *r.mapcalc*, *r.clump*

V.2 MACRO DEVELOPMENT

GRASS has often been referred to as a "tool chest" of commands. In this scenario, the user supplies the data and the notion of which functions to perform (i.e., a model), while GRASS supplies the commands (functions) necessary to run the model.

The GRASS commands listed below are actually *UNIX* shell scripts running a series of GRASS commands. They exist both to perform a specific function and to exhibit how such macros can be constructed. All (but *r.average*) are stored under the directory "$GISBASE/scripts". User sites are encouraged to examine these shell scripts and to develop their own macros based on these examples.

- 3d.view.sh
- show.color.sh
- dcorrelate.sh
- split.sh
- old.cmd.sh
- Gen.trac*TM*ap
- shade.rel.sh
- bug.report.sh
- slide.show.sh
- hsv.rgb.sh
- Gen.Maps
- rgb.hsv.sh
- blend.sh
- show.fonts.sh
- grass.logo.sh
- start.man.sh
- r.average

V.2.1 BASIC DATA MANIPULATION FUNCTIONS

V.2.1.1 R.RECLASS

This program creates a new *raster map layer* whose *category* values are based upon the user's re*classification* of categories in an existing raster *map layer*.

Format: *r.reclass* input=mapname output=mapname [title=name] [< *file*]

The command line version of *r.reclass* is identical in its function to the *interactive r.reclass*, it is "batch" oriented and its syntax often makes it quicker to use. The basic rule structure is:

category = reclassed category [optional *label*]

Example:

```
1 = 3 poorly drained soils
2 = 3                              [ a label is only]
3 = 3                              [ needed once for ]
4 = 1 well-drained soils           [ each new ]
5 = 3                              [ category ]
6 = 3
7 = 1
8 = 2 moderately drained soils
9 = 2
end
```

This could also be written:

```
1 thru 3 = 3 poorly drained soils
5 6 = 3
4 7 = 1 well-drained soils
8 9 = 2 moderately drained soils
end
```

Subsequent entries "override" earlier ones, thus we could write:

```
1 thru 6 = 3 poorly drained soils
4 7 = 1 well-drained soils
8 9 = 2 moderately drained soils
end
```

All original categories not listed in the *reclass* rules will be set to zero.

V.2.1.2 R.RESCALE

This program rescales the range of *category* values in a raster *map layer*.

Format: *r.rescale* input=name [from=min, max]output=name [to=min, max] [title="phrase"]

Example: Original map range: 900 to 4000;
Desired range to be rescaled: 1000 to 3500
New range (perhaps for color *hardcopy*): 1 to 25
Values not included in the old range will be reset to zero. In this example any values below 1000 or above 3500 would become zero.
r.rescale input=elev output=elev.25 from=1000,3500 to=1,25

Categories in original map	new categories
900 to 1000	0
1000 to 1100	1
1100 to 1200	2
...	...
3400 to 3500	25
3500 to 4000	0

V.2.1.3 COMPUTING MAP STATISTICS - MEAN, VARIANCE, STANDARD DEVIATION OF MAP CATEGORY AREAS

The following script calculates basic map statistics for a raster *map layer*, specified by the user. It first prompts for the name of that map and checks whether it exists. If it exists, it proceeds with running *r.stats.* The output of *r.stats* is then interpreted by an awk script that assigns values calculated by *r.stats* to temporary variables. These are then used for the calculation of aggregated areas.

```
g.ask element=cell type=old prompt="Raster map to be analyzed" unixfile=/tmp/$$
./tmp/$$
rm -f /tmp/$$
if ["$name" = ""]
then
  exit
fi
input="$fullname"
r.stats -za input="$input" | awk 'BEGIN {sum = 0.0; sum2 = 0.0}
NR == 1 {min = $2; max = $2; mincat = $1; maxcat = $1}
        {sum += $2; sum2 += $2*$2; N++}
        {
          if ($2 > max) {max = $2; maxcat = $1}
          if ($2 < min) {min = $2; mincat = $1}
        }
END {
print "Smallest area (clump) size is : ", min, "  Category value is:", mincat
print "Largest area (clump) size is : ", miax, "  Category value is:", maxcat
print "Mean area (clump) size       :", sum/N
print "Variance in area is          :", (sum2 - sum*sum/N)/N
print "Standard Dev of areas is     :", sqrt((sum2 - sum*sum/N)/N)
}'
```

V.2.1.4 COMPUTING STATISTICS ON MAP CATEGORY VALUES (E.G. ELEVATION SURFACES)

Whereas the above script calculated general map statistics, the following script deals with the values of the cells themselves. The method is very similar. The only differences are the assignment of variables within the awk script and the calculations performed with them.

```
input="$fullname"
r.stats -za input="$input" | awk 'BEGIN {sum = 0.0; sum2 = 0.0}
NR == 1 {min = $2; max = $1}
        {sum += $1; sum2 += $1*$1; N++}
        {
          if ($1 > max) {max = $1}
          if ($1 < min) {min = $1}
        }
END {
print "Smallest category value is : ", min,
print "Largest category value is  : ", max
print "Mean category value is     :", sum/N
print "Variance in categories is  :", (sum2 - sum*sum/N)/N
print "Standard Dev of cats is    :", sqrt((sum2 - sum*sum/N)/N)
}'
```

V.2.1.5 COMPUTING SPATIAL STATISTICAL SURFACES

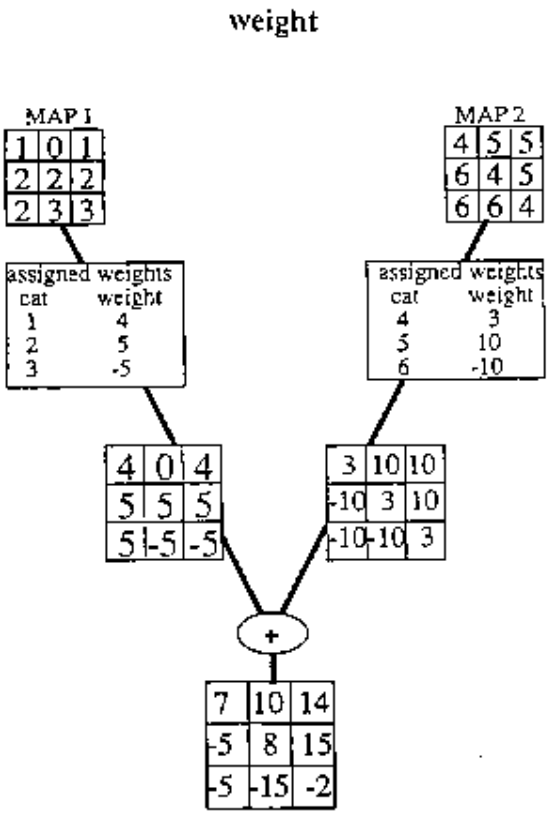
```
g.ask element=cell type=old prompt="Raster map to be analyzed" unixfile=/tmp/$$
./tmp/$$
rm -f /tmp/$$
```

```
if ["$name" = ""]
then
   exit
fi
input="$fullname"
g.ask element=cell type=new prompt="Name for results map" unixfile=/tmp/$$
./tmp/$$
rm -f /tmp/$$
if ["$name" = ""]
then
   exit
fi
input="$fullname"
r.stats -za input="$input"  |  awk 'BEGIN {sum = 0.0; sum2 = 0.0}
          {sum += $1; sum2 += $1*$1; N++}
END {print sum/N, sqrt((sum2 - sum*sum/N)/N)}'  > /TMp/$$
r.stats input="$input" >> /tmp/$$
awk 'NR == 1 {mean = $1; std = $2} {printf ""%d\n", $1, ($1 - mean) *10/std}' <
                                    /tmp/$$  |  r.reclass input="$input" output="$output"
rm -f /tmp/$$
```

V.2.2 ADVANCED DATA MANIPULATION FUNCTIONS

V.2.2.1 MAP OVERLAYWITH R.WEIGHT

In order for *r.weight* to generate maps useful for analysis, the user must make reasonable and defensible request. While much more powerful than *r.combine*, *r.weight* is also more dangerous. The user provides the necessary value judgements which are registered as weights. Only well-conceived value judgements will result in defensible results.



Step 1: BEFORE RUNNING WEIGHT

a) Define the question to be answered, e.g., "Locate sites suitable for logging"

b) Determine what mapped information is useful for answering the question, e.g. geology, soils, landuse, flood potential

c) Based on professional judgement, statistical inference and engineering principals, assign weights to the categories in the chosen maps.

Step 2: CHOOSE MAPS

In weight, use the command *choose* to identify up to six *map layer*s of interest

Step 3: ASSIGN WEIGHTS

Using the weight command *assign*, assign specific weights to map categories.

Step 4: SAVE ANALYSIS

Use the *save* command to save a copy of the analysis requested for later use.

Step 5: RUN ANALYSIS

Use *execute* to run analysis

Step 6: EVALUATE RESULTS

To modify an existing analysis request, use the *recover* command to retrieve the analysis and then go to step 3.

figure 39: *r.weight* Example

V.2.2.2 SAMPLE SCRIPT TO ELIMINATE CLUMPS BELOW A SPECIFIED SIZE

```
g.ask element=cell type =old prompt="Raster map to be analyzed" unixfile=/tmp/$$
./tmp/$$
# g.ask outputs a UNIX text file with the following structure:
# name='some_name'
# fullname='some_mapset'
# file='some_fullpath'
# the /tmp/$$ writes this file to a tmp file with the process id as the name

# the ./tmp command casts the variables into the environment so they can be used

rm -f /tmp/$$
if ["$name" = ""]
then
  exit
fi

input = "$fullname"
g.ask element=cell type=new prompt="Name for results map" unixfile=/tmp/$$
./tmp/$$
if ["name"=""]
then
  exit
fi
output="$name"

echo -n "Enter minimum clump size (in meters):"
read meters

set -x

# in the following line all the values in the input map are output to an awk script
# if the area is greater than the $meters then the category is printed as input to the reclass,
# otherwise it is not, and thus eliminated

 r.stats -a "$input" | awk "\$2 > $meters {print \$1 \"=\"\$1}" | r.reclass input="$input" output="$output"
```

The following figure shows the original and resulting *raster map* after running this shell script, accompanied by describing histograms

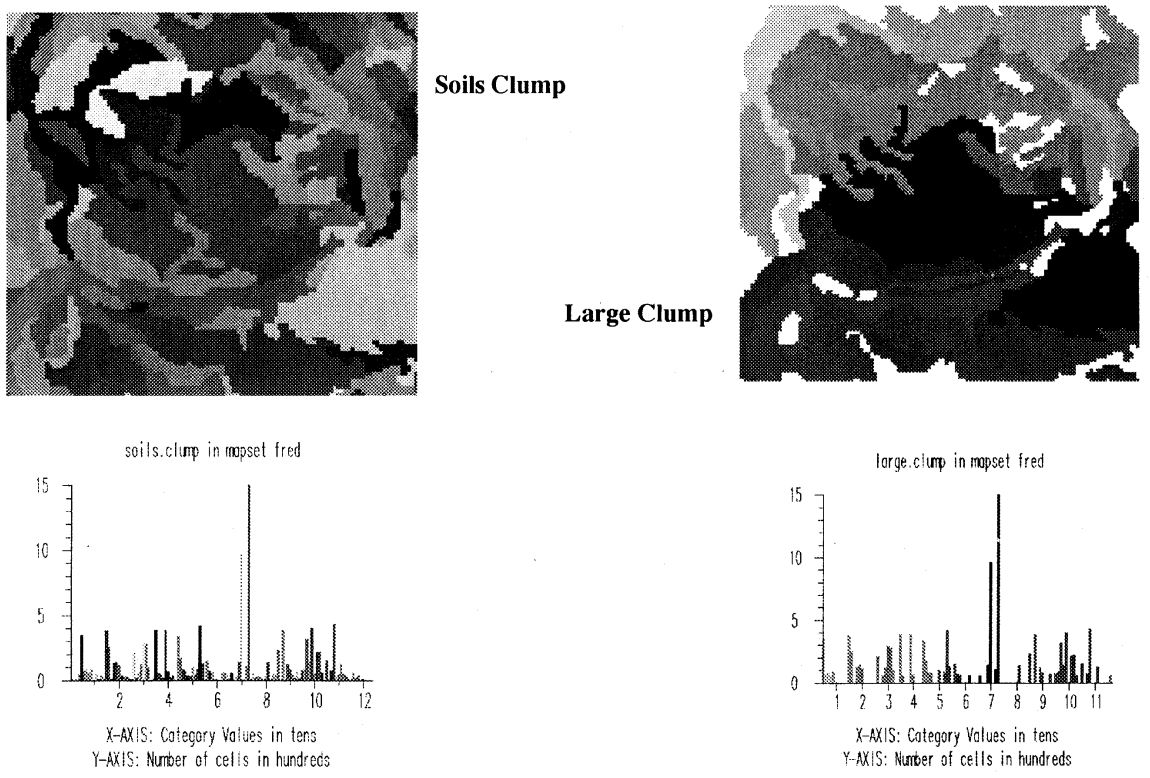


figure 40: Comparison of *Map Layer*s Before and After Running a Clump Script

V.2.2.3 SAMPLE SCRIPT TO ASSIGN CATEGORY VALUES TO CLUMPS WHICH REFLECT THEIR SIZE ORDER (SMALLEST IS 1, LARGEST IS .. MAX NUMBER)

```
g.ask element=cell type=old prompt="Raster map to be analyzed" unixfile=/tmp/$$
./tmp/$$
rm -f /tmp/$$
if ["$name" = ""]
then
  exit
fi

input="$fullname"
g.ask element=cell type =new prompt="Name for resulting map" unixfile=/tmp/$$
./tmp/$$
if ["name" = ""]
then
  exit
fi
output="$name"

r.stats -a "$input" | sort +1n | awk '$1!=0 {print $1, "=", i++}' | r.reclass input=$input" output="$output"
```

V.2.2.3 SPATIAL SAMPLING

Stratification of the sample is essential to good sample design. Introductionary text books usually deal with simple mappable characteristics, e.g. soil type x, floodplain, etc. The problem with this approach is that spatial units are multidimensional by their nature. Therefore, multidimensional sample strata have to be created. This is done in four steps:

Identify parameters of interest
Create maps of the same
Create permutation maps using *r.cross*
Select sample locations from each permutation according to sample fraction

Example: Vegetation mapping consisting of parameters soils.ph, slope and aspect

Step 1

reclass aspect map to eliminate chatter
r.mapcalc new.*aspect* = if(slope > 4, aspect, 25)

Step 2

reclass new *aspect* to fewer categories

```
r.reclass input=new.aspect output=aspect.5 <<EOF
1 thru 4 = 2 east
23 24 = 2
5 thru 10 = 1 north
11 thru 16 = 4 west
17 thru 22 = 3 south
25 = 5 flat
end
EOF
```

Step 3

reclass slope map to fewer categories

```
r.reclass input=slope output=slope.5  <<EOF
1 thru 4 = 1        cats 0 thru 4
5 thru 10 = 2       cats 5 thru 10
11 thru 16 = 3      cats 11 thru 16
17 thru 25 = 4      cats 17 thru 25
26 thru 30 = 5      cats 26 thru 30
end
EOF
```

Step 4

run *r.cross*

```
r.cross maps=aspect.5, slope.5, soils.ph output =permutations
```

Part of the report on the map "permutations"

```
|46|soils.ph(4) slope.5(3) aspect.5(3).....| 556|
|47|soils.ph(4) slope.5(3) aspect.5(4).....| 903|
|48|soils.ph(4) slope.5(4) aspect.5(1).....|  79|
|49|soils.ph(4) slope.5(4) aspect.5(2).....| 127|
```

Step 5

Create clumps of contiguous units for sample

Step 6

Remove small clumps - use remove.clump script

Step 7

reclass map to a single clump (example)

Step 8

r.random on clump with specified sample size

r.random input=perm.clump.148 nsites=1.0%
sites_output=perm.148

Sites list created by *r.random*

desc | Random sites from [perm.clump.148 in bizerte]
601753.41880342 | 4926865 | #1
601783.33333333 | 4926625 | #1
601454.27350427 | 4926595 | #1

V.2.2.5 LEAST COST PATHS, MODELING ALTERNATIVES

Traditional way:

1. Create a friction surface slope*slope
2. Run *r.cost* over friction surface from starting *point* (A)
   r.cost input=friction output=cost.a coor=(coordinates of A)
3. Run *r.drain* from *point* B back to starting *point* (A)
   r.drain input=cost.a output=drain.a coor=(coordinates of B)

This traditional approach has only a single "best" drain; therefore the

Alternative way:

1. Compute a cost surface from B as well
   r.cost input=friction output=cost.b coor=(coordinates of B)
2. Add the two cost surfaces together
   r.mapcalc both=cost.a + cost.b
3. Reclas*s* the result into "equivalent potential" or "fuzzy" paths

V.2.2.6 IDENTIFYING SPATIAL DISCONTINUITIES

Spatial analysis very often is more aimed at the detection of change in spatial distributions than in the distribution itself.

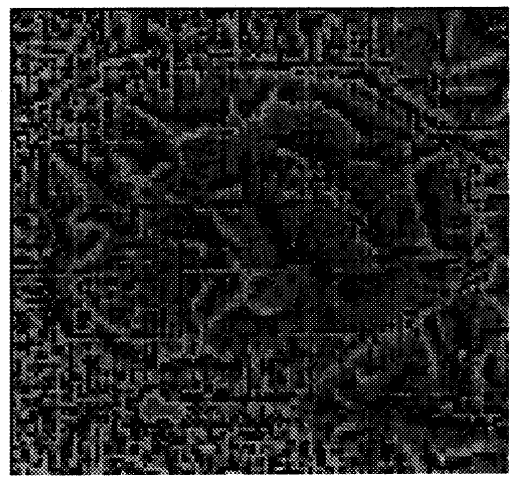
Computing a moving standard deviation

(compare this with section 2.1.5 "Computing spatial statistical surfaces")

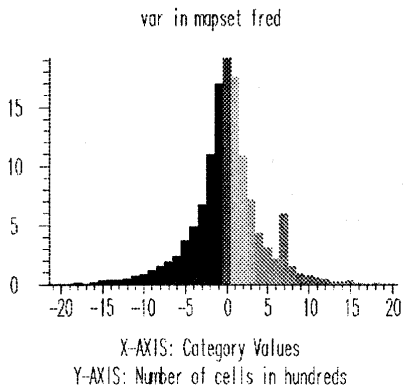
```
var = eval (sum = elev.DEM[-1,-1] + elev.DEM[-1,0] + elev.DEM[-1,1] + \
                  elev.DEM[0,-1]  + elev.DEM[0,0]  + elev.DEM[0,1  ] + \
                  elev.DEM[1,-1 ] + elev.DEM[1,0]  + elev.DEM[1,1],    \
sum.square = elev.DEM[-1,-1] * elev.DEM[-1,-1] + elev.DEM[-1,0] * elev.DEM[-1,0] + \
             elev.DEM[-1,1]  * elev.DEM[-1,1] + elev.DEM[0,-1] * elev.DEM[0,-1] +\
             elev.DEM[1,-1]  * elev.DEM[1,-1] + elev.DEM[1,0] * elev.DEM[1,0] +\
             elev.DEM[1,1]   * elev[1,1],                                       \
ave = sum /9.0,                                                                 \
var = (sum.square - 2.0 * ave * sum + 9.0 * ave * ave) / 8.0,                   \
std = sqrt(var),                                                                \
t.dif = elev.DEM - ave,                                                         \
t.dif * 10 /std)
```

Original elevation map

Map of local deviation {3x3}



Histogram of deviations



figrue 41: Comparison of the Original Elevation Map With a Map of Local Deviation

Cliffs can be identified on an elevation map *layer* by running two consecutive neighborhood operations, first with min then with max.

r.neighbors input=elev.DEM output=elev.min method=minimum size=3

r.neighbors input=elev.DEM output=elev.max method=maximum size=3

r.mapcalc dif = map.max - map.min

V.2.2.7 DYNAMIC MODELS (CELLULAR AUTOMATS)

The following very general structure for an iterative model displays a *snap*shot on the screen at every iteration and (could) save a map every 10th iteration:

```
#!/bin/sh
g.copy rast=t.start,  t
d.rast  t
i=1
while [$i != 10] do
   n=1
   while  [$n != 10] do
        r.mapcalc  <  spread.$1        a text file with t-equations like those on next page
        d.rast  t
        n = 'expr $n + 1'
   done
#g.copy rast=t,  t.$i
i = 'expr $i + 1'
done
```

The basic structure of a dynamic model is similar to the approach used in the last section.

[-1,-1]	[-1,0]	[-1,1]
[0,-1]	[0,0]	[0,1]
[1,-1]	[1,0]	[1,1]

t = if (t[-1,-1],   t[-1,-1] -1,   t)

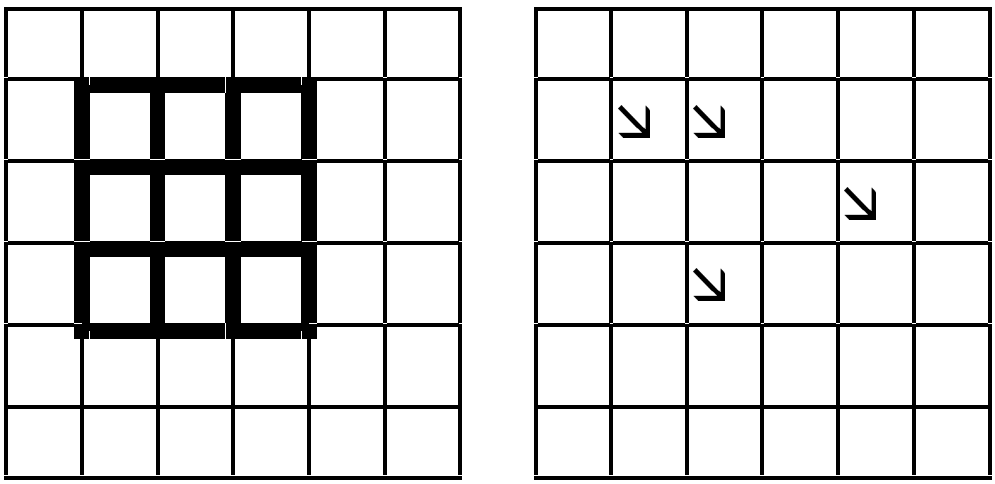


figure 42: Basic Structure of a Dynamic Model

A somewhat more realistic version of such a spreading model can be represented as:

t = if (t[-1,-1] > 0, t[-1,-1] - 1, if(t[-2,-1] > 0, t[-2,-1] - 3, if(t[-1,-2] > 0, t[-1,-2] - 3, t)))

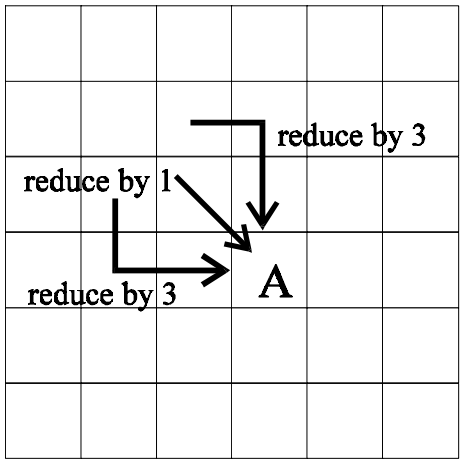


figure 43: Expanding the Basic Model

If the *cell* to the northwest is greater than zero then take its value, reduce it by 1 and put it in the current cell.
If the *cell* to the northwest is not greater than zero check to see if the *cell* two rows up and one column back is greater than zero.
If it is, reduce it by 3 and put it in the current cell.
If neither of these are true check the *cell* one row up and two columns back.
If it is non-zero reduce it by 3 and put it in the current cell.

Hindrance by mountains could be simulated by a mapcalc input *file* containing the line:
t = if (t[-2,-1] &&elev.DEM < 1325,1,if([-1,-2] && elev.DEM < 1325,1,t))

With a map *layer* describing the density of forest tree we could model the viewshed in a wood:

```
t = if (t[-1,-1] > 0,t[-1,-1] - density.2,  if (t[-1,0] > 0, t[-1,0] - (3*density.2),
      if (t[0,-1] > 0, t[0,-1] - (3*density.2), t)))
```

V.2.2.8 VIEWSHED MODELING

is usually done with a singular call to *r.los*. The techniques used in the previous section, however, could also be applied here to model the attenuation caused by smog, sparse trees or other factors. Assuming a constant attenuation through vertical column (which is not always true), we would use *r.cost* to:

create an "attenuation" surface = friction
run r.cost to sum attenuations from view*point*
determine maximum attenuation that would block view

V.3 MAPCALC INTRO

r.mapcalc is one of the most advanced GRASS programs. It was developed as an operationalization of Tomlin's (1990) "Map Algebra". The program uses mathematical operators and functions to modify maps and images; e.g. *sum=map1+map2*. The following examples were drawn from a recent tutorial by Mike Shapiro and Jim Westervelt from CERL. For more detailed information on the technical aspects of *r.mapcalc*, you are refered to their paper, which is available from the ftp server moon.cecer.army.mil. Here, only applications are shown that express the power of this "programing language". The examples fall into both, the fields of *GIS* and *image processing*.

V.3.1.1 MAP RECODING AND SELECTIONS

One of the most basic *GIS* operations is recoding of map values. From a digital elevation model (*DEM*) map file, we can extract a certain height slice by creating a new map

upland = if (elevation >= 1200 && elevation <= 1375, elevation).

While the above example creates a new map that contains all existing values between 1200 and 1375, with all others set to null, the next expression results in a binary map:

upland = elevation >= 1200 elevation <= 1375

Selections may involve any number of maps while map recoding works only with one input map at a time only.

V.3.1.2 SLOPE AND ASPECT

A simple elevation map *layer* can be used to calculate slope and aspect. Given a map "elev" with a *cell resolution* of 30 units, slope and *aspect* can be calculated as follows:

```
slope = eval( x= (elev[-1,-1] + 2*elev[0,-1] + elev[1,-1] - elev[-1,1] - 2*elev[0,1] - elev[1,1]) / (8.0 * 30.0), \
              y= (elev[-1,-1] + 2*elev[-1,0] + elev[-1,1] - elev[1,-1] - 2*elev[1,0] - elev[1,1]) / (8.0 * 30.0), \
              atan(sqrt(x*y + y*y)))
```

```
aspect= eval( x = (elev[-1,-1] + 2*elev[0,-1] + elev[1,-1] - elev[-1,1] - 2*elev[0,1] - elev[1,1]) / (8.0 * 30.0), \
              y = (elev[-1,-1] + 2*elev[-1,0] + elev[-1,1] - elev[1,-1] - 2*elev[1,0] - elev[1,1]) / (8.0 * 30.0), \
              a = round(atan(x,y)),                                                                                 \
              if (x||y, if(a,a,360)))
```

This example employs temporary variables a, x, y. The last two lines for the *aspect* expression take care of a possible "division by zero" error in flat terrain that has null slope.

V.3.1.3 HYDROLOGIC SIMULATION

Similar to the "dynamic modeling" example in section 2.2.7, *r.mapcalc* can be used to model water flow across a landscape. Assuming a constant amount of rain falling on each *cell*, the drainage to each of its neighboring *cells* works the same way as the cellular automate from above. In the following expression, diagonal *cells* gain 10% from their neighbors, while those in cardinal directions (2^{-2} units nearer) get 15% each. The resulting map *layer* "water" represents a single iteration of a dynamic model that has to be repeatedly run (preferably in a shell script).

```
water = water + eval(x = elev + water,                                                    \
        if ( x > (y = elev[-1,0] + water[-1,0]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[-1,0]>x, water[-1,0], y-x)) + \
        if ( x > (y = elev[1,0] + water[1,0]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[1,0]>x, water[1,0], y-x)) +     \
        if ( x > (y = elev[0,-1] + water[0,-1]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[0,-1]>x, water[0,-1], y-x)) + \
        if ( x > (y = elev[0,1] + water[0,1]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[0,1]>x, water[0,1], y-x)) +     \
        if ( x > (y = elev[-1,1] + water[-1,1]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[-1,1]>x, water[-1,1], y-x)) + \
        if ( x > (y = elev[1,1] + water[1,1), -0.15*if(elev>y, water, x-y), 0.15*if(elev[1,1]>x, water[1,1], y-x)) +      \
       if ( x > (y = elev[1,-1] + water[1,-1]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[1,-1]>x, water[1,-1], y-x)) + \
       if ( x > (y = elev[-1,-1] + water[-1,-1]), -0.15*if(elev>y, water, x-y), 0.15*if(elev[-1,-1]>x, water[-1,-1], y-x)))
```

You should keep in mind, that this is a very simple model. Nevertheless, with the similar methods applied, factors like evapotranspiration, ground saturation and flow impedance could be modeled as well.

V.3.1.4 ROUTING AND NETWORK ANALYSIS

Dana Tomlin showed in his "GIS and Cartographic Modeling" that *elements* of a raster (vector) network can be characterized by 47 conditions of adjacency based on whether the unit is a straight strech, a turn, an intersection or the beginning (ending) of a segment. These 47 conditions are shown on figure 45 (note that they assume a "thinned data structure with no "blocky" situations. Each of the adjacency conditions can be determined with an appropriate *r.mapcalc* syntax:

```
map46 = (if
            (\
      vect[-1,-1]       && vect[-1,0] == 0  && vect[-1,1]       &&\
      vect[0,-1] == 0 && vect[0,0]          && vect[0,1] == 0 &&\
      vect[1,-1]        && vect[1,0] == 0  && vect[1,1]              \
            ), 46)
```

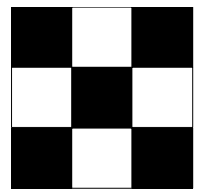


figure 44: MapCalc Example for Detection of an Intersection

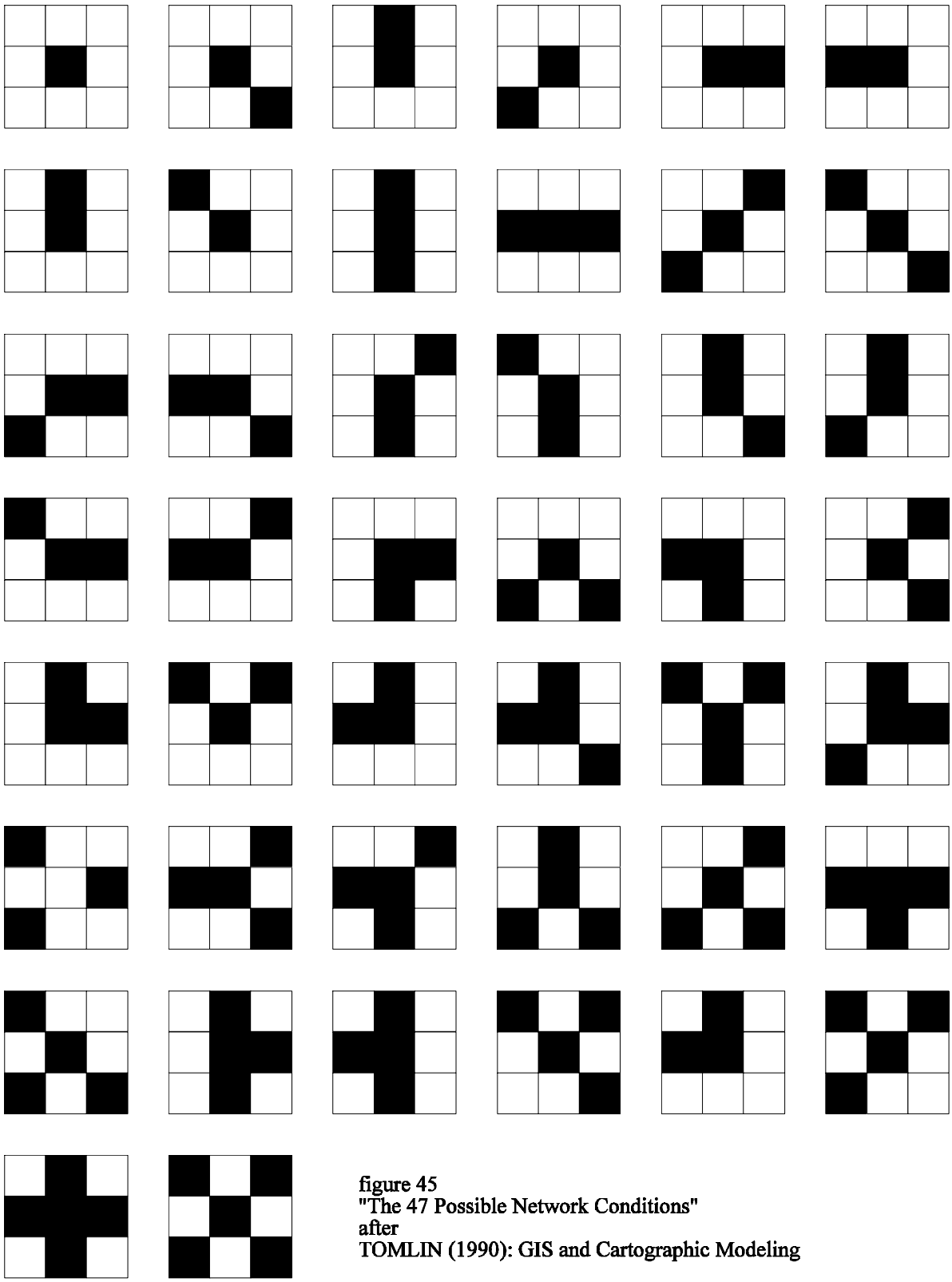


figure 45
"The 47 Possible Network Conditions"
after
TOMLIN (1990): GIS and Cartographic Modeling

```
map9 = (if
        (\
    vect[-1,-1] == 0  && vect[-1,0]   && vect[-1,1] == 0  &&\
    vect[0,-1]  == 0 && vect[0,0]     && vect[0,1]   == 0 &&\
    vect[1,-1] == 0  && vect[1,0]     && vect[1,1]   == 0       \
        ), 9)
```

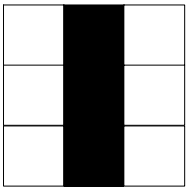


figure 46: MapCalc Example for Detection of a Vertical Line

Once the above structure has been applied to a map *layer* (e.g. roads) the following operations are possible:

Determining road segment length:

1. Reclas*s* all *vector* segments 30 and above to 0 (numbers according to fig. 39)
   The conditions 30 and above are intersections
2. Run *r.clump* on the remaining cells
3. Run *r.stats* (or *r.report* for manual purposes) on the clump map
4. Multiply (via an awk script) the number of cells in the clump times the *resolution* and assign the result to the clump
5. To be more precise compute different lengths for those conditions that are diagonals (use the square root of 2 times *resolution* for these)

To do shortest path computations with "impedances":

1. Assign speeds to all conditions based on shape and other factors
   for example --
   each clump (from above) would have a "speed limit"
   compute time to traverse the *cell* based on speed and *resolution*
   you can adjust based on shape -- e.g. sharp rights are slower than straights
2. Run *r.cost* from starting *point*

V.3.2.1 SPECTRAL RATIOS

Ratios between spetral bands are used to eliminate multiplicative effects, whereas ratios of between-bands differences eliminate additive effects. Such the normalized difference vegetation index (NDVI) for an AVHRR *image* may be computed as

```
ndvi = 255.0/90.0 * atan(float((avhrr.2 - avhrr1)) / float((avhrr.2 + avhrr.1)))
```

To retain results in the easily visualized range 0 - 255 (GRASS terminals display 256 colors) , the usual NDVI has been normalized with 255/90 * atan(expression).

V.3.2.2 MERGING PANCHROMATIC DATA WITH MULTISPECTRAL DATA

Lower *resolution* spectral bands may be merged with an high *resolution image* if the spectral data have been registered and resampled to the panchromatic data *cell* width. A simple solution would create three new *map layer*s, such as

```
merge.1 = (landsat.TM.1 + spot.pan) / 2
merge.2 = (landsat.TM.2 + spot.pan) / 2
merge.3 = (landsat.TM.3 + spot.pan) / 2
```

Another often used method would *transform* three selected bands from a red-green-blue color space to an intensity-hue-saturation color space. The intensity part would then be replaced by the panchromatic *image* and the three layers be transformed back into red, green and blue. While this could also be programmed in *r.mapcalc*, the functions *i.rgb.his* and *r.his.rgb* perform this job just as well.

VI IMAGE PROCESSING

GRASS integrates a geographic information system *(GIS)* with an *image processing* system. A *GIS* has four components: (1) data input, (2) data storage and retrieval, (3) data manipulation and analysis, and (4) data reporting. Data inputs may include spatial and thematic data derived from a combination of existing maps, aerial photographs and manual interpretations of remotely sensed images.

Remotely sensed images are acquired at some distance using a camera or a sensor. Within a *GIS*, an analyst may perform spatial procedures to generate new information such as the best *location* to build a road, preserve a wildlife habitat or harvest timber.

An *image* processing system has at least five *elements*: *image* input, *image* storage, *image* analysis, accuracy assessment, and information reporting. Note the similarity of these two systems.

When geographic information systems and *image* processing systems are integrated, not only can the *remote sensing* data be used to update the *GIS*, but the *GIS* thematic data and *attributes* can be used to guide *image classification*.

VI.1 TAPE EXTRACTION

To load or read *digital imagery* into GRASS, the commands listed below handle several different types of *digital imagery*: *MSS*, *TM*, *SPOT*, NAHP, etc.

i.tape.mss.h	extracts header information from *LANDSAT MSS* data
i.tape.mss	extracts Multispectral Scanner (*MSS*) data from tape
i.tape.TM	extracts *LANDSAT* Thematic Mapper (*TM*) images
i.tape.other	extracts scanned aerial imagery, *SPOT* and *TM* images
i.target	"targets" an imagery group *file* to a GRASS *LOCATION* with a geographic coordinate system

The GRASS tape extraction programs *import* elevation, land use and land cover, satellite imagery, and other data stored on magnetic tape into a GRASS *database*. These programs use the user's current *geographic region* settings for data extraction. The user is urged to check the boundaries and *resolution* of the current region setting before(!) extracting data, since GRASS tape extraction programs will only extract the data that fall within these boundaries and use only the set *cell resolution* during extraction. These settings can be modified with the *g.region* command.

VI.2 MANAGEMENT OF IMAGE FILES

In order to identify collections of imagery *band files* for *classification* and/or *rectification*, *image* files should be grouped together. Many imagery programs need to know which 'imagery group' to use when performing their particular task. The '*i.group*' command performs this function.

In addition, before running several *image* programs on an *image* group, the group must be 'targeted' to a GRASS *database* standard *LOCATION*, such as *UTM*. *i.target* accomplishes this task.

The commands associated with *image file* management are:

i.build.blk *i.group* *i.target*

VI.3 IMAGE CLASSIFICATION

One of the methods of deriving information from *digital imagery* is multispectral classification. Multispectral *classification* analyzes the spectral reflectance values of pixels in an image. The word "pixel" is usually used in reference to a *cell* when it is displayed on a computer monitor, but it is also often used to refer to the *cell* in a stored digital *image* that we cannot see until it is displayed. In raw spectral *digital imagery* each *cell* can have a digital number (DN) or spectral reflectance value ranging from 0 to 255. These are grouped into clusters or *categories* during *image classification* that can be interpreted to represent features on the surface of the earth.

There are many methods of multispectral classification, which fall into two main groups:

- supervised classification - unsupervised classification

GRASS 4.0 supports both kinds of classification. Both types of *classification* require a two-step procedure. The two GRASS programs that create a supervised classified *image* are:

- *i.class* - *i.maxlik*

Both programs must be run to complete a supervised classification.

The two programs that create an unsupervised *image* are:

- *i.cluster* - *i.maxlik*

As with a supervised classification, both programs must be run to complete an unsupervised classification.

Other GRASS programs which can be used in the *image classification* process are:

- *i.cca* canonical components analysis
- *i.pca* principal components analysis
- *i.zc* zero-crossing "edge detection" raster function
- *r.mapcalc* a very powerful "map calculator", see also chapter VI.3

VI.4 GEOMETRIC RECTIFICATION

Rectification is the mapping of an *image* from one coordinate system to another. The geometry of an *image* extracted from tape into a GRASS *LOCATION* having an x,y coordinate system is not planimetric. To make an *image* planimetric, that is, to convert the x,y coordinate system into a standard *map coordinate system* (e.g., the *UTM* coordinate system), *point*s from a map having the standard coordinates must be associated with the same *point*s on an *image* to be rectified. This is done by marking *point*s on an *image* and then inputting standard coordinates from a map. A squares regression is then calculated using the two types of coordinates for each marked *point*. A matrix containing transformation coefficients is the result. The coefficients are then used to calculate a standard coordinate for each x,y pixel in the image. The rectified planimetric *image* can then be deposited in a *target database*.

The GRASS programs that are used in *image rectification* are:

i.points	enables the user to mark coordinate system *point*s and enter the cordinates for each *point* to create a transformation matrix
i.rectify	computes the transformation for each pixel using the coefficient from *i.points*
i.build.blk	
i.rectify.blk	orthorectifies an *image group* file

VI.5 RADIOMETRIC CORRECTION

Commonly, when analyzing multiple images, it is necessary to correct *image* distortions attributable to such sources as the earth's radiation. Such distortions can result in different spectral reflectance values being collected for the same *image* pixel, based on the time of day, month, year, etc., at which the *image* was taken. GRASS programs exist which help the user to identify and eliminate such distortions from images, allowing valid comparisons to be made among images.

GRASS commands that can be used for *image* correction are:

i.fft	fast fourier transformation
i.ifft	inverse fast fourier transformation
i.zc	zero-crossing "edge detection"
r.mapcalc	a very powerful "map calculator", see also chapter VI.3

VI.6 IMAGE ENHANCEMENT

GRASS programs can be used to enhance or subdue certain data values in order to make more evident certain features or qualities of *digital imagery*. Following is a list of GRASS programs which can be used in the *image* enhancement process:

i.colors	sets colors for *image group files*
i.composite	creates a color compos*ite* from three *image bands*
i.grey.scale	interactively assigns a histogram contrast strech grey *scale* color table
i.his.rgb	color transformation from hue-intensity-saturation to red-green-blue
i.median	interactively creates a *raster map layer* whose color table is based on the red, green and blue color values present in existing *image group files*
i.rgb.his	color transformation from red-green-blue to hue-intensity-saturation
r.mapcalc	a very powerful "map calculator", see also chapter VI.3

BUILDING CUSTOM FILTERS

The user may wish to model interactions across a landscape or remove systematic distortion occurring across a map or image. Both applications can be accomplished by filtering data. Custom filters can be built which make the value of one pixel a user-defined function of the values in the surrounding *cells*.

The GRASS programs that can be used to build custom filters are:

- *i.zc* zero-crossing "edge detection"
- *r.mfilter* a matrixfilter
- *r.neighbors* makes each *cell* value a function of the values of its neighbors
- *r.mapcalc* a very powerful "map calculator", see also chapter VI.3

VII REPORT GENERATION

VII.1 GRASS COMMANDS FOR REPORT GENERATION

Several commands exist in GRASS to report information about a map's contents, or the results of an analysis on one or more maps, back to the user. Users can also develop macros specific to their needs to prepare other types of reports (see, for example, past issues of the newsletter "GRASSClippings").

The following GRASS commands report on the contents of one or more *map layer*s.

- *d.geodesic*
- *d.histogram*
- *d.legend*
- *d.measure*
- *d.profile*
- *d.rhumbline*
- *d.what.rast*
- *d.what.vect*
- *d.where*
- *d.rast.arrow*
- *d.rast.num*
- *r.average*
- *r.cats*
- *r.coin*
- *r.covar*
- *r.describe*
- *r.info*
- *r.profile*
- *r.report*
- *r.stats*
- *r.volume*
- *r.what*
- *v.stats*
- *s.menu*
- *s.db.rim*

VII.2 CUSTOMIZING REPORTS

GRASS programs like *r.report*, *r.coin*, and *s.menu* output their results in the form of reports. New options made available in GRASS 4.0 allow the user added flexibility in designing the form of these reports.

Users can also use GRASS programs in combination with *UNIX* programs like "awk", "sed", and "sort" to completely customize the formats of reports.

One sample use of "awk" with GRASS programs appears in the Winter/1989 issue of the newsletter *"GRASSClippings"* (article by Fred Limp, Vol 3, No 1, pp 14-16). This script uses output from the *r.stats* program to calculate a region's expected total annual crop yield based on a *raster map layer* of estimated crop yield per soil type. It is reproduced below using GRASS 4.0 command syntax.

```
r.stats -aq soil.yield.map | awk '{sum+=$1*($2/100000)*2.47} END {print"Total Yield:",sum}'
```

When piping GRASS program outputs through other commands like "awk", users should include the option to run GRASS commands quietly (i.e., non-verbosely). Frequently, this will be a -q option as in *r.stats -q* above.

VIII INTERFACES BETWEEN GRASS AND OTHER SOFTWARE

One useful function performed by GRASS is to output data in a form suitable for input to other existing computer models. GRASS programs allow GRASS data to be interfaced with those of the BNOISE (blast noise simulation), ARMSED (watershed and sedimentation modeling), ELAS (*image* processing), UW-RIM (University of Washington's version of the RIM relational data base management system), the "S" statistical package, and ISM (*interactive* surface modeling) programs.

These programs are distributed with GRASS, but the user must compile them separately. Compilation instructions are included in the "GRASS 4.0 Installation Guide" distributed with the "GRASS User's Reference Manual".

In some cases (e.g. to compile RIM and the GRASS programs interfacing RIM), the user needs a FORTRAN compiler to compile this software. A FORTRAN compiler is not distributed with GRASS.

The user can use GRASS programs in combination with *UNIX* functions and other programs to build macros that convert data into a form suitable for *import* and export to/from programs other than those listed above.

APPENDIX A: SOURCES OF DIGITAL DATA

In this section, major sources of national and/or global *digital* landscape data, and status, coverage and acquisition information about those sources, are identified. Information is provided on data: format, scale/*resolution*, coverage, media, costs, and textural descriptions and comments.

This information is needed to assess the cost and time requirements of building a *digital database* for any specific *geographic region*. Use of existing *digital data* sources, rather than construction of new *digital data*, can result in tremendous savings in both time and money. Also, use of standard sources insures minimum standards for accuracy and quality.

More sources may be available at state agencies such as transportion, environmental, and natural resource management departments. Local planning offices, university geography, urban planning, and civil engineering departments may also provide data.

Data acquisition decisions deal with projections, *scale* and possible sources:

	Soils	Wetlands	Elevation	Hydro-graphy	Trans-portation	Political Bounda-ries	Imagery Satellite Coverages	Socio-Economic Data
Source	SCS, FAO		USGS				Landsat *MSS+ TM* *SPOT* AVHRR	Census
Scale	1:15,000 - 1:1,000,000		1:25,000 - 1:1,000,00				10m 30m 1000m	1:100,000
Type	Digital and non-digital		Digital				Digital	Digital and non-digital
Cost	Varies with scale							
Info								

EARTH OBSERVATION SATELLITE CO. (EOSAT)

EOSAT: Multispectral - Satellite

Data Source:

Earth Observation Satellite Co.(EOSAT)
4300 Forbes Blvd.
Landham, MD. 20706
Phone: 1-800-367-2801

Data Format:

MSS: BIP (until January, 1979)
BSQ or BIL corrected (1/79 - 5/81)
BSQ or BIL corrected or uncorrected (6/81 - present)
TM: BSQ or BIL corrected or uncorrected (4/84 - present)
RBV: SSQ corrected (9/80 - present)

Scale/*Resolution*:

MSS ~ 80m,
TM ~ 30m,
RBV ~ 80m

Data Coverage:

Worldwide. Historical data may be obtained through EROS Data Center.

Data Description:

Standard 4 *band MSS*, 7 *band TM* data, or RBV data. Since the *LANDSAT* system has been commercialized, users must sign a form stating that they will not copy or distribute the data without prior authorization from EOSAT.

Media:

9-track tape, 6250 or 1600 bpi.

Acquisition Costs:

Product	Cost	Product	Cost
MSS	$ 660	*TM* Full Scene	$3300
MSS copies	$ 120 +	*TM* Quarter Scene	$1650 *
RBV Sub-scene	$ 660	*TM* copies	$ 720 +
RBV 4 Sub-scenes	$1320		

Comments :

* Quarter scenes are only available from stock in-house at EROS Data Center.
+ Copies are available only at the time of the original order and no more than 10 copies are allowed.

NGDC: Terrestrial & Marine Geophysical Data

Data Source:

National Geophysical Data Center
325 Broadway
Boulder, CO 80303
Phone: 303-497-6474; FTS 320-6376
Attn: Christine Schomaker

Data Format: Varied.

Scale: Varied.

Data Coverage:

Mostly worldwide, some data specific to U.S. coastal waters.

Data Description:

Aero & Geo magnetic data; oil and gas lease data; ocean core sample locations; ocean bottom characteristics; dangers to navigation; coastal, deepwater, or gridded bathymetry; land and marine seismic data; geothermal data; land and marine geology; satellite data (Aurora, GEOS/NOAA, LANDSAT, MAGSAT,..); solar data; land and marine topography.
All data is latitude/longitude referenced.

Media:

9-track tape (ASCII or EBCIDIC), computer listing, or plots.

Acquisition Costs:

Varied; call/write for quotes.

Comments:

NGDC is part of the National Environmental Satellite Data & Information Service (NESDIS).

National Technical Information Service (NTIS)

NTIS: Small Scale World Boundaries, Major Rivers

Data Source:

National Technical Information Service
U.S. Dept. of Commerce
5285 Port Royal Rd.
Springfield, VA 22161
Phone: (703)487-4650, FTS 737-4650

Data Format: Vector (points, lines)

Scale/*Resolution*: 1:12,000,000; 1:3,000,000; & larger

Data Coverage: Worldwide

Data Description:

World Data Bank I - Coast lines, national boundaries, and place names at 1:12,000,000 scale.
World Data Bank II - Coast lines, major rivers, national boundaries, state/province boundaries, and place names at 1:3,000,000 or larger.
Data is latitude/longitude referenced.

NOTE:

According to sources in the CIA the copy of World Data Bank II distributed by NTIS is not as complete as that of the CIA's.

Media:

9-track, 1600 bpi, odd parity, EBCIDIC magnetic tape

Acquisition Costs:

Data is available free of charge from the CIA to federal agencies (see Comments section). All that is required is a letter of request on official letterhead designating *area* of interest and enough 1/2 inch tapes to contain the data (the whole *database* will take 10 tapes).
Others may obtain current price information by calling NTIS.

Comments:

Data originally created by the CIA Office of Geographic & Cartographic Research.

Système Probatoire pour l'Observation de la Terre (*SPOT*)

SPOT: Multispectral - Satellite

Data Source:

SPOT Image Corp.
1897 Preston White Dr.
Reston, VA 22091-4326
Phone: (703) 620-2200

Data Format: Raster

Scale/*Resolution*: 10-meter panchromatic or 20-meter multispectral

Data Coverage: Worldwide, though historical data is not yet extensive.

Data Description:

The *SPOT* satellite multispectral data is essentially the same as those used by bands 2, 3, & 4 of the Landsat-5 Thematic Mapper. However, *SPOT*'s multispectral data has a *resolution* of 20m versus Landsat's 30m. *SPOT* also has a panchromatic *band* with a *resolution* of 10m and stereo *image* aquisition capability.

Media: 9-track 6250/1600 bpi tape

Acquisition Costs:

	6250bpi	1600bpi
Unreferenced (Level 1)	$1475	$1600
Georeferenced (Level 2)	$2425	$2550

Comments:

Level 2 *SPOT* data can be bought already referenced to several projections (e.g., Lambert Conformal, *UTM*, Polyconic,..) or it can be referenced to another *SPOT* scene (Level S). If no maps of the *area* are available the customer will be asked to supply maps at a *scale* of 1:100000 or larger for ground *control point*s.

The following non-standard options are available for an additional $100.- per item:

Band Sequential Format (BSQ) rather than *Band* Interleaved (BIL)
EBCIDIC rather than ASCII
Split record size with maximum length of 3960 bytes
(only available for 1600 bpi tapes)

Appendix A SOURCES OF DIGITAL DATA

APPENDIX B: DIGITIZER EXAMPLE

SUPPORTED DIGITIZERS

Currently, USACERL supports software linking GRASS with the following digitizers:

Altek	CalComp	Kurta
Numonics	Geographics	Hitachi
none -- (for on-screen digitizing)		

Some users have written their own software that enable GRASS to be used with other digitizers. Such software is referred to as a *device driver*. The GRASS Information Center at USACERL maintains a partial listing of digitizers and other peripherals being used by other GRASS user sites. This list is available upon request. Users can also write their own device *driver*s. Some help for doing this is available on-*line* in the directories under $GISBASE/src/mapdev/digitizers and in the "GRASS Programmer's Manual" (Shapiro et al., 1989).

CalComp Digitizers
Manufacturer: CalComp Inc.
2411 West LaPalma Avenue
Anaheim, CA 92801
(714) 821-2872

Specifications:

Accuracy	-	+/- .010"
Connection	-	RS232 Serial Interface
Resolution	-	1,279 Lines per Inch

Item #	Description	List Price
91480	36"x48" Surface w/ Control Electronics	3,146
91004	RS-232 Single Port DTE	241
91036	16-Button Cursor	283
91053	120V Power Supply	189
91082	Cursor Holder	51
91070	Power Base	951
	TOTAL	$4,861

Instructions for installing these *digitizer* device *driver*s on your system is included in the "GRASS 4.0 Installation Guide" distributed with the "GRASS 4.0 User's Reference Manual". Instructions are also on-line in individual digitizer directories under $GISBASE/src/mapdev/digitizers. These instructions detail specific settings and cabling requirements for different systems listed above.

EXAMPLE: Installing a Calcomp Digitizer

This *file* details how to set up the hardware for a Calcomp 9100 series digitizer. The cabling and switching settings are suggestions only. A different Masscomp or Sun may need a different cable configuration.

NOTE: That the table size has to be set by the SB1 dip switches. Consult your Calcomp manual for your particular table settings.
These switch settings and cable configurations are for Port A/C. There is information about Port B/D mentioned later in this file.

Switch Settings
There are two banks of eight switches (SW1 and SW2) on the Communication Interface Board. There is one bank of eight switches (SB1) on the Digitizing Processor.

SW1		SW2	
1 - closed		1 - open	B/D on
2 - closed	parity disabled	2 - open	
3 - closed		3 - closed	LF port A/C off
4 - closed	1 stop parity	4 - open	Port A/C on
5 - open	8 data bits	5 - closed	menu off
6 - closed		6 - open	cursor on
7 - closed	9600 baud	7 - closed	echo off
8 - open		8 - closed	LF port B/D on

SB1	
1 - open	
2 - open	operating mode (run)
3 - closed	
4 - open	output format 2
5 - closed	
6 - close	*resolution* 1000 LPI
7 - open	
8 - close	91480, 36 X 48 inches (table size)

If the *v.digit* program isn't working, make sure the $GISBASE/etc/'digitcap' *file* is correct for your system configuration, and the switch settings on the *digitizer* are correct. There are some diagnostic programs under $GISBASE/src/mapdev/digitizers/calcomp/Diagnostics. If none of the programs work, then there is something wrong with either the cabling or the digitizer.

GLOSSARY

Listed below is a compilation of the most commonly used *GIS* and GRASS terms.

arc - GRASS *vector maps* are stored in an *arc-node* representation consisting of non-intersecting curves called *arcs*. An *arc* (also known as *vector*) is stored as a series of x,y coordinate pairs. The two end-*point*s of an *arc* are called *nodes*. Two consecutive x,y pairs define an *arc* segment (also known as *line* segment). *Arcs*, either singly or in combination with other *arcs*, form higher-level *map features*: *line* features (e.g., roads or streams) and *area* features (e.g., soil types or timber cutting units). Areas are also called *polygons*. GRASS *vector format* does not store the polygons explicitly; they are reconstructed by finding the particular *arcs* which form the *polygon* perimeter. *Arcs* that form linear features are sometimes called *lines* and *arcs* that out*line* areas are called *area* edges or *area* lines.

arc-node - Graphic data (e.g., maps) can be stored digitally in either *raster* or *vector format*. In GRASS, *vector data* are stored in an *arc-node* representation, consisting of non-intersecting curves called *arcs* or *vectors* and the *nodes* at their end-*point*s.

area - A homogeneous region containing data having the same *category value*. An *area* (also known as *polygon)* is formed by the series of *arcs* outlining its perimeter. GRASS *vector format* does not store polygons explicitly; when used to describe *vector data*, therefore, areas refer to *area* edges. More generally, *area* can refer to a geographically defined region (rather than to a region defined by a *category* value).

aspect - The direction in which a slope faces, commonly expressed in degrees clockwise from north. A directly west-facing slope, for example, would have an *aspect* of 270 degrees. Maps of slope and *aspect* can be automatically calculated in GRASS from a map of elevation.

attribute - In GRASS, data are categorized by their distinguishing characteristics (also known as attributes or categories), such as depth to water table, soil moisture, etc. In a *map layer*, each unique *attribute* type is tagged with a unique integer value called a *category value* and a *category label* describing the *category* contents; each map can contain *category* values within the range +/- 2^{31}. *Attribute* values may be assigned to *point*s (also known as sites), linear features, and *area* features.

band - Data falling within a specific range of wavelengths on the electromagnetic spectrum that depict an *image* in terms of spectral reflectance values observed through a *remote sensing* device. Images may be obtained from satellite, aircraft or *scanned* photography.

band file - A digital *file* depicting a remotely-sensed *image* in terms of spectral reflectance values within some specific *band* on the electromagnetic spectrum. *LANDSAT* Thematic Mapper (*TM*) images typically represent data in each of the seven bands, while *LANDSAT* Multispectral Scanner (*MSS*) images contain four bands of data and *SPOT* color images offer three bands of data. Panchromatic images represent data in terms of a single band.

Boolean operations - Mathematical operations which perform logical combinations of data, involving union, intersection, complement and exclusion. GRASS analysis programs like *r.weight*, *r.combine*, *r.infer*, and *r.mapcalc* can be used to perform *Boolean operation*s on one or more maps.

Cartesian coordinate system - A coordinate system in which the locations of *point*s in space are expressed by their reference to three planes, called the "coordinate planes" (x,y, and z) that are all perpendicular to each other.

category - In GRASS, data are categorized by their distinguishing characteristics (also known as, attributes or categories), such as depth to water table, soil moisture, etc. In a *map layer*, each unique *attribute* type is tagged with a unique integer value called a *category value* and a *category label* describing the *category* contents; each map can contain *category* values falling in the range +/- 2^{31}. *Attribute* values may be assigned to *point*s (also known as, sites), linear features, and *area* features.

category label - Alpha-numeric text that describes a category's content (e.g., "Areas of 0-2% slope"). Maps in *vector format* store *category* labels underneath the 'dig_cats' directory, while maps in *raster format* store these labels under the 'cats' directory.

category value - In GRASS, the numeric tag associated with an *attribute* (also known as, category). *Category* values must be integers; each map may contain *category* values falling in the range +/- 2^{31}. Maps in *vector format* store *category* values beneath the 'dig_att' directory, while maps in *raster format* store these values under the 'cats' directory.

cell - The fundamental unit of information in a *raster* map. In their *raster format*, a grid containing rectangular cells of uniform size is superimposed over map data. Each *cell* is assigned a value (also known as *category value*) and *label* (also known as *category* label) describing the type of data found there. Each *cell* is also associated with a specific geographic x,y, coordinate location. In GRASS, the user specifies *cell* dimensions in the x (east-west) and y (north-south) directions, for each map; these may differ. Grid *cell* size is linked to map *scale* and *resolution*. Smaller grid cells enable the assignment of more values to a given area, and are therefore associated with a larger *scale* and a finer *resolution*, than are larger grid cells.

cell file - See *raster file*.

centroid - The gravitational center *point* of a *polygon* or the mid*point* of a *line*. The centroid's *location* is described as an (x,y) coordinate pair and is used by GRASS to locate *category* labels assigned to polygons (*area* features) and lines (linear features).

class - In GRASS, an *image processing* term, used to denote the *category* (i.e., classification) into which *image* data in a remotely-sensed *image* can be placed.

classification - The categorization of features in a remotely-sensed *image*, based on similarities in their spectral reflectance. The process of determining a set of defined features from *image* data.

control points - Maps in a *GIS* must be re*GIS*tered to some coordinate system. If *GIS* analysis and display functions are to be performed on multiple maps, these maps must be re*GIS*tered to the same coordinate system. On each map, the user re*GIS*ters the locations of between 4-10 *point*s ("*control point*s") having known geographic coordinates, and inputs this information to GRASS. Given these *control point*s, and a coordinate system (e.g., *UTM*), GRASS calculates the positions of all other *point*s on the map.

Cathode Ray Tube (CRT) - A device with an electronic tube for displaying graphics and other information. Also known as a visual display device (VDD).

database - A collection of data sets stored together and controlled by a specific schema. A consistent and specified set of procedures is used in adding data to a *database* and in changing or retrieving existing data from it. A *GIS database* includes data about the positions and the attributes associated with geographic features that have been coded as *point*s, lines or areas.
In GRASS, the term *database* refers to the *UNIX* directory in which the user's GRASS data are stored on a computer. The variable $*GIS*DBASE is also used to refer to this directory.

database element - An entity within a *database*. In GRASS, *database elements* are the names of the *UNIX* directories located under a GRASS *mapset* in which GRASS data and *support files* are stored. These *elements* include: *cell* (*cell maps*), dig (*vector maps*), site_lists (*site maps*), icons (p.map icon files), paint (p.map *label* files), window (*window* files), group (imagery group files), etc.

Data Base Management System (DBMS) - A systematic approach to maintaining, accessing and manipulating *files* stored in a *database*. A *DBMS* may consist of a single program or a collection of task-specific programs. GRASS contains programs which allow it to be interfaced with several existing *DBMS*. Typically, *DBMS* allow the user to manipulate and store multiple pieces of information (data) for a single *point*.

Digital Elevation Model (DEM) - A *file* with terrain elevations recorded at regular intervals on a fine-grained grid; the *digital* equivalent of the elevation data on a *topographic map*. 1-degree *DEM*s have a ground (horizontal) *resolution* of roughly 100 meters, while 7.5 minute *DEM*s have a ground *resolution* of roughly 30 meters.

digital data - Data which are represented in discrete, quantized units or digits, commonly for computer manipulation.

digital imagery - Any imagery can be represented in *digital* form so that the pattern of the *image* brightnesses forms an array of numeric values. In their digital form, these brightnesses (also known as spectral reflectance values) can then be treated as numeric values so that they can conveniently be added, subtracted, multiplied, divided, and in general statistically manipulated in ways that are difficult or impossible were the *image* presented in a pictorial format.

digitize - To make *digital*; the process of converting an analog *image* or map into a digital format usable by computer software.

digitizer - A device used to convert *point* locations on a graphic *image* (e.g., a paper map) into planar (x,y) coordinates for *digital* processing.

digitizing threshold - The *digitizing threshold* determines which *point*s are saved during *digitizing*. Expressed, in map distance, as the width of the corridor surrounding each *point* outside of which digitized *point*s will be saved.

driver - A program which directly interfaces a computer's software with a hardware device (like a *graphics monitor*, printer, or *digitizer*) and handles requests made to that hardware device by the software. GRASS provides *driver*s to enable specific *graphics monitor*s, *raster* output devices, *digitizing tablet*s and other peripherals to be used for GRASS input and output. Users can also write their own *driver*s to allow other peripheral devices to be used for GRASS input and output.

display coordinate system - An x,y coordinate system based on the user's *graphics monitor* (display device). Some GRASS programs use this coordinate system, rather than a *map coordinate system*, to specify where on the *graphics monitor* objects will be displayed. Typically, x and y coordinate values are stated as percentages of the display frame's (horizontal) width and (vertical) height, measured from the frame's lower left corner.

easting - The x coordinate of a geographic *point* location. Frequently used to describe the easternmost and westernmost boundaries of a map, especially when stated in *UTM* coordinates.

editing - To remove errors from, or to modify, a *digitized* map or data file.

element - A fundamental geographic unit of information, such as a *point*, linear feature, *area* edge, *cell* or pixel. In GRASS, the term element is also used to refer to *database elements*, which are *UNIX* directories located under each GRASS *mapset*.

file - A storage unit for computerized information that can be accessed by a (relatively) unique name. Files may be stored on tapes or (hard)disks.

geocoding - The activity of defining the position of geographic objects relative to a standard reference grid. The assigning of specific *area*, linear, and *point map feature*s to their associated *digital* geographic coordinates so as to enable computer manipulation.

geographic region - A rectangular *area* (region), stated in the *map coordinate system* of the GRASS *location* under which the region is stored, whose sides are parallel to the coordinate axes. A *geographic region* also has an east-west *resolution* and a north-south *resolution* associated with it, which specify the dimensions used by individual *cells* falling within the region's boundaries.

Geographic Information System (GIS) - A system of computer hardware, software, and procedures designed to support the capture, management, analysis, modeling, and display of spatially referenced data for solving complex planning and management issues.

graphics display frame - A rectangular *area* on a *graphics monitor* to which GRASS display output can be sent.

graphics monitor - A *CRT* capable of displaying maps and other graphic outputs in color.

grid format - A data structure in which data are encoded and stored as a regular, rectangular unit of area, called grids or *cells*.

grid map - Another name for *raster map*. A map in which information is stored in the form of grid *cells*.

grouping - The process of linking a set of imagery *band files*.

hardcopy - A physical representation of a map or other data; for example, a paper map produced on a *line* printer or pen plotter.

i.group - *i.group* is an imagery program that allows the user to group any of the raster imagery files that are present in the current *location* and currently selected *mapset* into:

1. a *subgroup* of the first group created during the tape extraction,
2. a new imagery group, and subsequently, a *subgroup* of a new imagery group,
3. an already existing group in the current *location* and *mapset*, and subsequently, a *subgroup* of an already existing group.

i.group also allows the user to remove raster imagery files from a group in his or her *mapset*. Raster files, however, cannot be removed from, or added to, existing *subgroup*s. If a new *subgroup* is needed, the procedure recommended is to create a new *subgroup*. *i.group* is able to offer its options to all of the raster imagery files in the *mapset*s that were selected using the GRASS command *g.mapsets*. *i.group*, however, is restricted to the files and selected *mapset*s in the user's current GRASS location.

image - In GRASS usage, *image* generally refers to a remotely-sensed representation of a geographic area, stated in terms of spectral reflectance values. A series of one or more *band files* together constitute this image.

image processing - The acquisition, import, and manipulation of remotely-sensed data. In GRASS, *image* processing includes such steps as data import, classification, and *rectification*.

import - The process of bringing data or software from one computer system into another.

interactive - Refers to a system allowing two-way communication between user and the software. In GRASS, some programs are *interactive* and "prompt" the user for input, while others rely on the user's knowledge of a command's specific syntax and usage statement structure.

island - In *GIS* usage, the term *island* refers to a *polygon* which lies completely within another polygon. The two polygons contain data with different *category values*.

join - Typically refers to the connection of two or more digital maps containing data for different (often adjacent) geographic *regions*. Such maps may originally have been *digitized* separately due to *digitizer* size limits. Sometimes, *join* refers to the joining of *digital data* (e.g., two *line* segments) within a map.

label - A name, or other nominal code, assigned to a graphic representation of a feature on a map. Labels are comprised of small amounts of *attribute* data (in alpha-numeric from) descriptive of and assigned to the graphic representation of a *map feature*.

LANDSAT - A series of *remote sensing* satellites, launched since 1972, which produce images of the earth.

layer - A separation of mapped information according to theme. Many *geographic information systems* allow the user to manipulate the contents of a single map *layer* or any combination of layers at a time.

legend - A key; a basic element of map design, explaining the meanings of the symbols used to graphically designate any depicted *elements* in a map.

line - A basic geographic element, defined by an ordered set of *point*s, that describes the position and shape of a linear feature on a map. In GRASS usage, the term *line* does not necessarily refer to a straight line, but merely to a series of ***arcs*** (*vector*s) forming the perimeter of a linear feature (e.g., stream or railroad). *Arcs* that form linear features are sometimes called lines, and *arcs* that out*line areas* are called *area* edges or *area* lines.

location - A directory containing the GRASS *mapset*s accessible by a user within a given GRASS session. All *mapset*s stored under the same GRASS *location* must use the same *map coordinate system*. Typically, each GRASS *location* stores *mapset*s whose data are referenced to the same *geographic region* on the earth's surface. Each *location* directory contains the special *mapset* "PERMANENT" and other user *mapset*s. The variable $LOCATION_NAME denotes the user's current GRASS location.

map coordinate system - Either a geographic coordinate system (like *UTM* or latitude/longitude) or an arbitrary coordinate system (like the X,Y coordinate system initially assigned to imported satellite imagery), used to reference the locations of objects in a two-dimensional space.

map database - A directory storing geographic data. In GRASS, the GRASS *database* is known by the variable $*GIS*DBASE. Users can work with only those maps stored beneath a specified *database*, location, and *mapset* during a single GRASS session.

map feature - A *category* designation. *Map feature*s can be either *area* features (*polygons*), linear features (lines), or *point* features (*sites*).

map generalization - The process of reducing the detail of spatial features in a *map layer*. Often undertaken in conjuction with a reduction in *map scale*. The process can be semi-automated for certain kinds of data, such as features, but requires more sensitive analysis for thematic maps. See also *reclass*.

map layer - A set of data, commonly relating to a single theme (e.g., soil mapping units, geology types, etc.) which are represented and stored on maps. See also *raster map* and *vector map*.

mapset - A directory in which GRASS *database* element*s* are stored. At the start of each GRASS session, the user must specify a current *mapset* ($*MAPSET*), current *location* ($LOCATION_NAME), and current *database* ($*GIS*DBASE). Any maps created or modified by the user in the current GRASS session will be stored in the user's current *mapset*. The user may also access data stored in other *mapset*s under the same GRASS *location* as the current *mapset*. All *mapset*s stored under the same *location* directory refer to the same general *geographic region*, and must use the same *map coordinate system*.
A special *mapset* (named 'PERMANENT') is stored under each *location* directory; it contains some files used by all *mapset*s under that *location* (e.g., those specifying a default *geographic region*, the name of the geographic location, the location's *map coordinate system*, etc.), and commonly also contains standard maps needed by many applications.

map threshold - The map *threshold* expresses the *digitizing threshold* in different units. It is expressed in units on the ground, as the width of the corridor surrounding each *point* outside of which digitized *point*s will be saved.

mask - The GRASS command which creates a *raster map* that restricts analysis to a particular *geographic region* designated by the user ("masking out" all other areas from analysis). This map, named MASK, can be altered, created or destroyed with the GRASS "mask" command.

menu - A list of available options displayed on the screen from which the user can choose by using the keyboard or a *point*ing device such as a *mouse*.

monitor - A visual display device to which computer outputs can be sent. Usually refers to a *graphics monitor*. See also *driver* and *graphics monitor*.

mouse - The most commonly used device to *point* to a position on the graphics display monitor. Other devices may include light pens, touch screens, or graphics *tablet*s. If your hardware configuration includes one of these latter *point*ing devices, read the term "*mouse*" to mean whichever *point*ing device your *graphics monitor* uses.

Multispectral *Scanning* System (MSS) - A device, often carried in airplanes or satellites, for recording received radiation in several wavebands at the same time. Inputs are used to map remotely-sensed imagery by received spectral reflectance values.

neighborhood anaylsis - In GRASS, an analytic procedure run on a map in its *raster format* in which a new value is assigned to each *cell* as a (user-specified) function of the values in the surrounding ("neighborhood") cells. Currently, in GRASS, neighborhoods must be square; their size is specified by the user.

no data category - Also referred to a "null category". *Map layer* data that are not assigned any *category value* are automatically assigned a *category value* of "0" and a *category label* of "no data". Each map *layer* contains this category, in addition to others specified by the user.

node - In GRASS, *vector data* are stored in an *arc-node* representation. An *arc* (*vector*) is stored as a series of x,y coordinate pairs. The arc's end-*point*s are called nodes. Two consecutive x,y pairs define an *arc* segment (*line* segment). *Arcs*, either singly or in combination with other *arcs*, form higher-level *map features*: *line* features (e.g., roads or streams) and *area* features (e.g., soil types or timber-cutting units).

northing - The y value of a geographic coordinate. Frequently used to describe the northernmost and southernmost boundaries of a map, especially when stated in *UTM* coordinates.

null category - Also referred to as "no data category". *Map layer* data that are not assigned any *category value* are assigned a *category value* of "0" and a *category label* of "no data". Each map *layer* contains this category, in addition to others specified by the user.

overlay - The process of stacking *digital* representations of various spatial data atop one another so that each position in the *area* covered can be analyzed in terms of these data.

plane coordinates - Coordinates specifying the locations of *point*s in planar space. In cartography, this usually is a *projection* of the earth's surface onto a flattenable cone or cylinder, and the x and y values scaled along the x and y axes are respectively called *eastings* and *northings*.

point- A basic geographic element, referring to an object with one dimension.

polygon - A multi-sided figure representing an *area* on a map. Maps in GRASS *vector format* represent polygons as the *area* edges forming their perimeters. Maps in GRASS *raster format* store areas as a block of grid cells.

projection - The manner in which the spherical surface of the earth is represented on a two-dimensional surface; concerned primarily with minimizing distortion in area, shape, distance, and direction.

pruning - The process of selectively deleting information. Used most often in GRASS to describe the process of deleting unnecessary *point*s from lines in *digit files*. This process reselects *point*s based on a current *threshold* value specified by the user.

raster-to-vector - The process of converting data in *raster format* into a *vector format*.

raster data (raster format) - A data format whose basic unit is the grid cell. Data are stored on a grid containing rectangular *cells* of uniform size. *Cell* dimensions denote data *resolution*. Each *cell* within the grid can be assigned a data value. Smaller grid *cell* size therefore allows more data to be stored for a given geographic *area* (i.e., can present data at a larger *scale*). See *vector data* for comparison.

raster file - The *digital data file* in which *raster map data* are stored in GRASS. Each data *file* describes a specific thematic map in terms of its geographic coordinates. Supporting files describe the *category* values, labels, map title, history, etc., corresponding to this *raster data* file. All of the files relating to a given *raster data file* are collectively termed a raster *map layer*.

raster map - A geographic data *file* stored in a *raster data* format and any existing supporting files relating to it. Includes *cell files*, *category* files, color table files, *cell* header files, etc.

reclass - To recategorize the tagged *attributes* (*category* values) in a map. The function is commonly used to create new maps (e.g., *reclass*ing a map of soil mapping units into a map of soil permeabilities) or to reduce the level of detail (e.g., *reclass*ing a map of true elevation values into a map of elevation ranges).

rectification - The process by which the geometry of an *image* is made planimetric. This is accomplished by mapping (*transforming*) an *image* from one coordinate system to another.

registration - The process of establishing a system of spatial reference for mapping using *control points*. *Control point*s are used to map *digitized point*s onto a *map coordinate system*.

remote sensing - The act of detecting and/or identifying an *image* without having the sensor in direct contact with the object. Includes satellite and other aerial *scanning* or photographic methods.

resolution - The smallest spacing between display *elements*; the smallest size of a feature that can be mapped or sampled. In GRASS, because grid cells can be rectangular, *resolution* in the x direction may differ from that in the y direction. Fine *resolution* implies large map *scale*.

residual - In map *registration*, a *residual* is the amount of error that a re*GIS*tered *control point* contributes to the overall map re*GIS*tration error. "Error" is defined as the difference between the value assigned by the user to a *control point* and its expected value based on a user-specified *map coordinate system* and the values of the other re*GIS*tration *control point*s.

scale - The relation between the size of an object on a map and its size in the real world. "Large-scale" and "small-scale" are relative terms, denoting the relationship of mapped objects to their actual size. Objects mapped at large-*scale* are closer to their actual sizes, while objects mapped at small-*scale* are less close to their actual sizes. See also *resolution*.

scanning - The process of using an electronic input device to convert analog information such as paper maps, photographs, overlays, etc., into a *digital* format usable by computer software.

screen coordinates - An x,y coordinate *location* on the graphics display monitor, stated in terms relevant to the monitor. Frequently, x values are stated as a percentage of the monitor's width, while y values are stated as a percentage of the monitor's height. Same as *display coordinate system* coordinates.

site - A *point* whose spatial *location* is specified as a coordinate pair. *Category* values, labels, and graphic icons may also be associated with sites.

site map - A *raster map* that consists of a collection of *sites*.

snap - The process of connecting two or more lines, using the GRASS digitizing program, by giving them a common *node*.

snapping threshold - The radius within which *vector nodes* will be "*snap*ped" and said to be at the same geographic coordinate location. Used during digitizing, when importing *vector file*s to GRASS *vector format* and when converting *vector file*s to GRASS *raster format*. Generally expressed as a map distance but may also be expressed as ground distance.

SPOT - Acronym for Système Probatoire pour l'Observation de la Terre. A satellite with high *resolution* sensors launched by France in January 1986 to map the earth.

search path - In GRASS, users may access maps, created by others, that are stored in *mapsets* under the same geographic *location*. Users can specify the names and order in which *mapset*s will be searched for maps in their *mapset search path*. More generally, in *UNIX* terminology, a *search path* specifies the names of and order in which directories will be searched for user-specified files and programs.

subgroup - An imagery term. The process of *grouping* raster *map layer*s together that enables the user to run analyses on any combination of the raster *map layer*s in a group or *subgroup*.

support files - Data files associated with each *raster map* and *vector map*, that contain supporting information on map *category* values and labels, color tables, map history, etc. specific to each map.

tablet - The sensitized surface of a *digitizer*, used for the *interactive* input of non-*digital data* into *digital* format in GRASS. A *tablet* is one component of a GRASS *workstation*.

target - In GRASS usage, the *database location* with which images re*GIS*tered to an arbitrary x,y coordinate system can be linked, in order to co-re*GIS*ter images to maps under this location.

threshold - The level, *point*, or value above which an action will take place and below which it will not. Usually referred to in the context of *digitizing* maps.

TM - Acronym for Thematic Mapper. An imagery product, containing seven spectral bands, collected by *LANDSAT* satellites.

topology - The way in which geographic element*s* are linked together in space.

topographic map - A map showing the elevation (z) values associated with x,y locations on a map.

transform - The process of changing the *scale*, *projection*, or orientation of a mapped image.

UNIX - The operating system, under which GRASS runs, on various machines.

UTM - Acronym for Universal Transverse Mercator. The Universal Transverse Mercator grid system is a system of *plane coordinates* based on 60 north-south tending zones, each 6 degrees of longitude in width, that encircle the globe.

UTM zone - One of 60 zones, each 6 degrees of longitude in width, into which the earth is longitudinally divided under the Universal Transverse Mercator (*UTM*) coordinate system.

vector data (vector format) - A data format in which area, linear and *point* features are represented and stored as the series of *arcs* (also known as *vector*s) forming their perimeters. In GRASS, map data can be converted between *vector* and *raster* data formats. Typically, data are *digitized* in *vector format* and analyzed in their *raster format*. Data can be displayed and output in either format.

vector file - A *file* storing data in an *arc-node* (*vector*) format. A GRASS *vector file* is generally associated with other supporting files describing *attribute* (category) values and *labels*, and *topology*.

vector map - Refers to a data *file* in *vector* (*arc-node*) format and any existing supporting files relating to it. Together, these files (including *vector files*, *attribute* values (*category values*) and labels (*category labels*), and *topology* files) constitute a *vector* map. See also *arc* and *raster* map.

vector - A quantity having both magnitude and direction. See also *arc*.

workstation - The computer disk, keyboard, display monitors, digitizers, printers, etc., that are connected together as a unit for working with maps or graphics in an *interactive GIS*.

zone - See *UTM zone*.

zoom - A capability for proportionally enlarging or reducing the *scale* of a figure or map display on a *CRT*.

Bibliography

"GRASSClippings Newsletter", (GRASS Inter-Agency Steering Committee, ISSN 0899-7853).

Brooks, Douglas, Michael Higgins, and Mark Johnson, "GRASS Hardware Configurations Guide", ADP Report N-89/21, (USACERL, March 1989).

Burrough, P. A., "Principles of Geographical Information Systems for Land Resources Assessment", (Oxford University Press, Oxford, England, 1987).

Goran, William D., "Testing Guidelines for GRASS Ports and *Driver*s", ADP Report N-89/22 (USACERL, January 1989).

Goran, William D., and R. E. Riggins, "Geographical Information Systems for Training Land Evaluation", (in Army Research, Development and Acquisition, Sept/Oct 1983, pp 26-28).

GRASS Information Center, U.S. Army Construction Engineering Research Laboratory, Environmental Division, Attention: Brenda Johnson, P.O. Box 9005, Champaign, IL 61826-9005, Phone: 217/373-7220, Fax: 217-373-7222, email: brenda@cerl.cecer.army.mil

Martin, Mary V., William D. Goran, Robert C. Lozar, Jean M. Messersmith, Marilyn S. Ruiz, and James D. Westervelt, "GRASS/*GIS* Implementation Guide", Special Report N-90/02 (USACERL, October 1989).

Ressler, John Q., William Swain, Sandra Anderson, and Jerry Thompson, "The GRASS Problem-Solving Manual", ADP Report N-89/15 (USACERL, August 1989).

Ruiz, Marilyn, and Jean M. Messersmith, "Cartographic Issues in the Development of a Digital GRASS *Database*", Special Report N-90/16 (USACERL, September 1990).

Shapiro, Michael, James Westervelt, et al., "GRASS 3.0 Programmer's Manual", ADP Report N-89/14 (USACERL, September 1989).

Snyder, "Map Projections - A Working Manual" (1989).

Westervelt, James, Michael Shapiro, William D. Goran, et al., "GRASS User's Reference Manual", ADP Report N-87/22 (USACERL, September 1988).