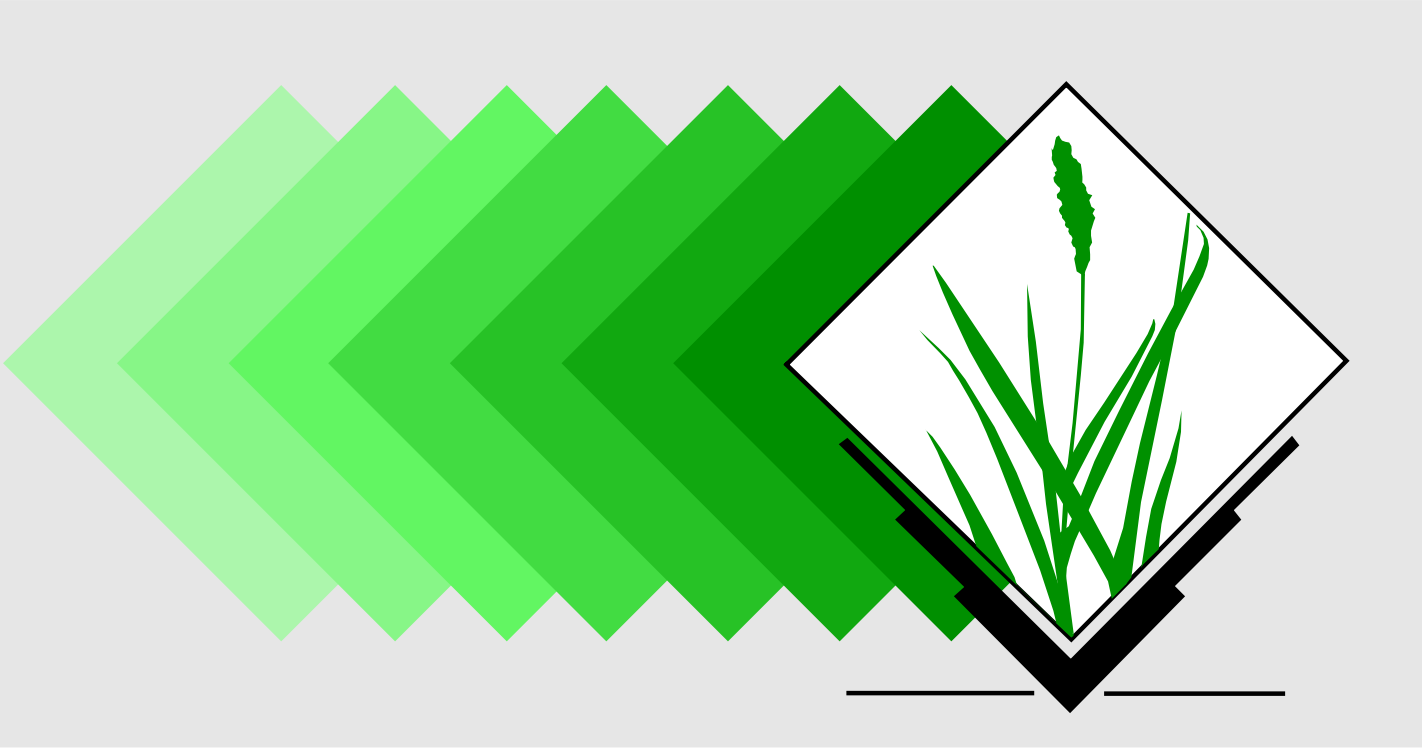
GRASS HANDBUCH :: VERSION :: 1.1

Leitfaden zum Geographischen Informationssystem GRASS

Markus Neteler



GRASS-Handbuch

Der praktische Leitfaden zum Geographischen Informationssystem GRASS

Markus Neteler

Version 1.2 (2000, 2003)

Dieses Buch ist keine Originaldokumentation zur Software.

Die in diesem Buch genannten Soft- und Hardwarebezeichnungen sind in den meisten Fällen auch eingetragene Warenzeichen und unterliegen als solche den gesetzlichen Bestimmungen. GRASS GIS ist ein eingetragenes Warenzeichen und unterliegt der GNU General Public License. Nähere Informationen beim *GRASS Development Team*, ITC-irst, Trento, Italien.

Dieses Dokument wurde mit LaTeX gesetzt. Es ist als Quelltext, im PDF- und HTML-Format sowohl online als auch gedruckt erhältlich.

Die in diesem Buch enthaltenen Angaben, Daten, Ergebnisse usw. wurden vom Autor nach bestem Wissen erstellt und mit Sorgfalt überprüft. Dennoch sind inhaltliche Fehler nicht völlig auszuschließen. Daher erfolgen alle Angaben ohne jegliche Verpflichtung oder Garantie. Autor und Herausgeber übernimmt daher auch keinerlei Verantwortung oder Haftung für Fehler und deren Folgen. Hinweise auf eventuelle Irrtümer werden gerne entgegengenommen.
Anschrift: *Dipl.-Geogr. M. Neteler, Email: neteler@itc.it*

Vorwort zur ersten Auflage

Das in diesem Leitfaden vorgestellte Geographische Informationssystem GRASS – „Geographical Resources Analysis Support System“ – ist ein GIS mit einer weitreichenden Entstehungsgeschichte. Entwickelt wurde es ab 1982 vom U.S. Army Corps of Engineers/CERL (Construction Engineering Research Lab) mit einem Aufwand von einigen Millionen U.S.$ für militärische Planungszwecke. Ende der 80er Jahre stellte CERL das gesamte Softwarepaket samt Quellcode der zivilen Öffentlichkeit zur Verfügung. Die starke Verbreitung des Internets seit den frühen 90er Jahren hat dazu beigetragen, dass sich GRASS in Kürze weltweit etablieren konnte. Im Jahr 1995 zog sich CERL aus dem Projekt zurück, seit 1997 haben das „GRASS Development Team“ an der Baylor University, Texas, U.S.A., und an der Universität Hannover, Deutschland, sowie weltweit weitere Personen die Weiterentwicklung übernommen. Ende 1997 wurde GRASS 4.2 vom CAGSR an der Baylor University veröffentlicht. Kurze Zeit später, im Frühjahr 1998, folgte dann die darauf aufbauende Version GRASS 4.2.1, die vom Autor am Geographischen Institut der Universität Hannover bis Ende 1999 koordiniert wurde. In dieser Version konnten quasi sämtliche bekannte Programmfehler beseitigt und GRASS um rund 50 neue Module im Bereich Vektor-/ Rasterdatenverarbeitung erweitert werden. GRASS 4.2.1 stellt die derzeit vollständig stabile Fassung dar. Seit Frühjahr 1999 wird parallel an GRASS 5.0 gearbeitet, dessen Funktionalität erweitert ist und ansatzweise bereits die kommende Weiterentwicklung von GRASS in Richtung 3D/4D-GIS (mit Voxelverarbeitung und der Berücksichtigung der Zeitebene) zeigt. GRASS 5.0 ist inzwischen so stabil, dass es GRASS 4.x ersetzen kann. GRASS wird kostenlos über das Internet verteilt.

Die Projektseiten sind im Internet über die „GRASS GIS Europe“-Adresse `http://grass.itc.it/` bzw. über die „GRASS GIS U.S.A.“-Internetseiten `http://grass.baylor.edu/` erreichbar.

Das vorliegende „GRASS-Handbuch“ hat auch schon ein wenig Geschichte: Ursprünglich aus „Kochrezepten“ entstanden, die seit 1995 als Anleitungen zur Unterrichtung der Studierenden am Institut für Landschaftspflege und Naturschutz (ILN) der Universität Hannover dienten, entwickelte sich 1996 ein erster ausführlicherer Text im Stile von umfangreicheren Programmbeschreibungen daraus. Unterdessen erfuhr das „GRASS-Handbuch“ umfangreiche Überarbeitungen und Erweiterungen bis zur heutigen Form.

Dieses Handbuch richtet sich sowohl an im Umgang mit GIS Erfahrene, die GRASS neu kennenlernen möchten, als auch an GIS-Anfänger. Daher sind der eigentlichen Beschreibung von GRASS ein Abschnitt über Geographische Informationssysteme im Hinblick auf GRASS und eine ausführliche Anleitung zum Thema UNIX/Linux vorangestellt. Ein Schwerpunkt liegt auf der Datenintegration, da hier erfahrungsgemäß generell im Umgang mit GIS viele Fragen auftreten. Einige Beispiele von GIS-Applikationen sollen Anregung für eigene Projekte geben.

Für viele interessante Gespräche und die hilfreiche Kritik am vorliegenden Handbuch danke ich insbesondere Herrn Dr. Heinrich Stillger und Herrn Dr. Manfred Redslob vom Institut für Landschaftspflege und Naturschutz (ILN) der Universität Hannover.

Herrn Dipl.-Geogr. Matthias Akkermann danke ich herzlich für seine kritischen Anmerkungen bezüglich neu hinzugefügter Kapitel.

Herrn Professor Dr. Thomas Mosimann danke ich für die kritische Durchsicht der aktuell vorliegenden Ausgabe und für die Möglichkeit, das Handbuch in der Schriftenreihe „Geosynthesis“ veröffentlichen zu können.

Hannover, im Februar 2000

Markus Neteler

Vorwort zur Version 1.1

Nachdem das Handbuch in der Schriftenreihe „Geosynthesis“ nach zwei kleineren Nachdrucken endgültig vergriffen war, konnte ein Weg gefunden werden, den Text unter der *GNU Free Document License* (GNU FDL) bereitzustellen. Es wurden einige kleinere Änderungen und Korrekturen vorgenommen, der Gesamttext in seiner Form weitestgehend belassen. Der Text konzentriert sich demnach auf die GRASS-Versionsreihe 4.x, kann aber wegen der ausführlichen Hinweise auf GRASS 5.0.x auch für diese derzeit stabile Version problemlos benutzt werden.

Herrn Otto Dassau danke ich für hilfreiche Kommentare für diese Version des Handbuchs.

Ein wichtiger Hinweis für die Benutzung dieses Handbuchs mit aktuellen GRASS-Versionen:

Einige Befehle haben sich von Version zu Version verändert, den Namen gewechselt oder stehen in manchen Versionen nicht zur Verfügung. Ab der GRASS Version 5.0.0pre4 (vom 13 Mai 2002) erreicht man aus verschiedenen Gründen die GIS Variablen `$GISDBASE`, `$LOCATION` und `$MAPSET` nur noch über das Modul `g.gisenv`. Diese Variablen werden hier im Handbuch noch direkt benutzt, müssen also bei neueren Versionen mit `g.gisenv` abgefragt werden. In zukünftigen GRASS Versionen wird es nicht mehr nötig sein, diese Variablen benutzerseitig zu verwenden.

Trento, im Mai 2003

Markus Neteler

Vorwort zur Version 1.2

Einige Aktualisierungen und kleinere Fehler konnten korrigiert werden.

Trento, im Oktober 2003
Markus Neteler

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Übersicht zum geographischen Informationssystem GRASS

Geographische Informationssysteme halten immer stärker Einzug in öffentlichen Institutionen und Planungsbüros. Neben der bisher üblichen linien- und flächenorientierten Arbeitsweise, die mehr einen Datenbank- und Verwaltungscharakter hatte, werden heutzutage mehr und mehr die hybriden Eigenschaften der Geographischen Informationssysteme (GIS) ausgeschöpft. Damit ist die parallele Verarbeitung und Verknüpfung von Punkt-, Linien- und Flächenstrukturen – im GIS entsprechend von Punkt-, Vektor- und Rasterdaten – gemeint. Diese Daten lassen sich nicht nur speichern, sondern es können vor allem durch Datenaggregierung, -analyse und -synthese neue Daten und Ergebnisse berechnet werden. Im Bereich der Simulationsrechnung erlauben ins GIS integrierte und extern gekoppelte Modelle weitergehende Prognosen und Abschätzungen.

1.1 Was leistet GRASS GIS?

Das Geographische Informationssystem GRASS *(Geographical Resources Analysis Support System)* ist ein kombiniertes Raster-/Vektor-GIS mit einem integrierten Bildverarbeitungs- und Visualisierungssystem. Es enthält über 400 Programme und Hilfsmittel, um Raster-, Vektor- und Punktdaten zu bearbeiten, Karten am Bildschirm und auf Papier zu erzeugen, Multispektralbilddaten zu prozessieren und räumliche Daten zu erstellen, zu handhaben und zu speichern. Neben einer fensterorientierten Benutzeroberfläche gibt es auch eine kommandozeilenorientierte Befehleingabe. GRASS kann an Drucker, Plotter, Digitalisierbretter angeschlossen werden und auf externe Datenbanken zum Berechnen neuer Daten sowie Speichern vorhandener Daten zugreifen. Es ist damit ideal für Landschaftsplanung und ingenieurtechnische Anwendungen. Wie bei anderen GIS-Paketen kann auch GRASS Vektordaten für Straßen, Ströme, Grenzen und andere Objekte verarbeiten. Mittels seiner integrierten Vektor-Digitalisierfunktion lassen sich digitale Karten aktualisieren. Für die räumliche Datenanalyse ist die Fähigkeit, Rasterdaten (Rasterzellen) handhaben zu können, eine besonders wichtige Eigenschaft. GRASS-Module können Vektordaten und Rasterdaten wechselseitig im Format umwandeln. Weiterhin ist eine frei verfügbare Internetschnittstelle vorhanden, die mittels GRASS ein vollständiges Online-GIS im World-Wide-Web (WWW) aufbauen lässt. Eine Beispielanwendung ist auf dem GRASS-Server am ITC-irst, Trento (Italien), installiert und in den GRASS-Internetseiten wird auf weitere verwiesen.
Die Stärke von GRASS liegt auf mehreren Gebieten: Die einfache Benutzerschnittstelle bildet eine ideale Möglichkeit für diejenigen, die zum ersten Mal mit GIS arbeiten und es erlernen möchten.

GRASS kann Karten und Daten vieler weitverbreiteter GIS-Pakete einschließlich ARC/INFO und IDRISI importieren und exportieren. Sehr interessant und in keinem proprietären GIS in diesem Umfang möglich ist die eigene Programmierung neuer GIS-Module. Fortgeschrittene Benutzer, die also ihre eigenen GIS-Module schreiben möchten, können dieses anhand der vorhandenen Quellprogramme bzw. durch das Programmierhandbuch lernen und die dokumentierte, umfangreiche GRASS-GIS-Bibliothek (API für die C-Programmiersprache) für ihre Zwecke nutzen. Diese Programmierbibliothek erlaubt, neue, hoch entwickelte Funktionalität in GRASS zu integrieren. Auf Scriptebene können auch einfachere Anwendungen erzeugt bzw. Arbeitsabläufe automatisiert werden.

Die Fähigkeit, Rasterdaten verarbeiten zu können, bietet die Möglichkeit, mit GRASS als einem Oberflächen-Modellierungssystem zu arbeiten. GRASS enthält mehr als 100 Multifunktionsmodule zur Rasteranalyse und -handhabung. Oberflächenprozesse wie Niederschlag-Abfluss-Modellierung, Fließwegberechnung, Hangstabilitätsuntersuchungen und weitere räumliche Datenanalysen sind nur einige der vielen Anwendungsbereiche. Da viele Rastermodule multifunktional sind, können Benutzer eigene Karten mit den in GRASS gespeicherten Daten herstellen.

Zusätzlich zur zweidimensionalen Standardanalyse erlaubt GRASS, Daten in drei Dimensionen zu betrachten. Rasterdaten, Vektor- und Punktdaten können für die Visualisierung verwendet werden. Beispielanwendungen solcher Fähigkeiten umfassen Luftraumanalysen für Flughafenplanung, Geländeanalysen und die Ermittlung räumlicher Tendenzen. Visuelle Hilfsmittel in GRASS erlauben, die Darstellung räumlicher Daten auch zu animieren und hier zwischen Datenebenen umzuschalten. Die in der 3D-Visualisierung erzeugten Ansichten können als Standbilder oder als MPEG-Film für ein späteres Abspielen und Analysieren gesichert werden.

Begleitend zu Landschaftsplanung und Ingenieuranwendungen enthält GRASS eine Sammlung an Modellen für den Bereich der hydrologischen Modellierung und Analyse. Funktionen sind u.a. die Berechnung von Einzugsgebieten, die SCS-Curve-Number Berechnung, Hochwasseranalysen und die Nutzung einiger unterschiedlicher Modelle für eine komplette einzugsgebietsbezogene Abflussberechnung. Andere GRASS-Module können Diagramme und Statistiken zu den modellierten und kalibrierten Daten erzeugen. Zusätzlich kann GRASS Geländedaten verwenden oder sogar entsprechende Parameter simulieren, die auf numerischen Daten basieren.

Die Bildverarbeitungsmodule sind mit proprietären Spitzenprodukten dieses Sektors vergleichbar und z.T. wesentlich umfangreicher als in proprietären Standard-GIS-Paketen. Sie umfassen zahlreiche Programme zur Prozessierung und Auswertung multispektraler Satellitendaten sowie in ein Modul zur Produktion von Orthofotos aus gescannten Luftbildern. Damit stehen quasi sämtliche Wege der Datenintegration ins GIS zur Verfügung.

Ergänzend zur traditionellen Kommandozeilenversion von GRASS ist 1998 eine neue Benutzerschnittstelle, basierend auf Tcl/Tk, geschrieben worden. Damit steht eine einfache graphische Benutzerschnittstelle zur Verfügung, die plattformunabhängig ist. Diese intuitive Benutzerschnittstelle lässt GRASS-Benutzer schnell und leicht Daten importieren, ansehen und bearbeiten. Alle Hauptmodule, die in GRASS enthalten sind, sind auch in der neuen Benutzeroberfläche vorhan-

den. Die Oberfläche stellt in einem Eingabefenster die übliche Standard-Kommandozeile bereit und gibt damit Benutzern Zugang zur gesamten Funktionalität von GRASS.

Das GRASS Entwicklungsteam mit seinen weltweit verteilten Programmierern (auf fast allen Kontinenten) arbeiten daran, die Fähigkeiten von GRASS auszubauen und zu erweitern. Zukünftige Entwicklungen schließen neue Module, die dem Benutzer bzw. der Benutzerin vollständige Arbeitsfähigkeit für die dreidimensionale Datenverarbeitung geben, als eine Fähigkeit mit ein, die nicht in üblichen anderen GIS-Paketen existiert. Benutzer können in der zukünftigen „echten“ 3D-GRASS-Version mit Rasterdaten genauso arbeiten wie mit Vektor- und Punktdaten. Die bisher umgesetzten Verbesserungen gegenüber den GRASS 4.x-Versionen umfassen die Fließkommaverarbeitung im Rasterbereich und die Unterstützung multipler Attribute im Punktdatenformat. So ist die maximale Anzahl an Dimensionen und Attributen im Punktformat bereits quasi unbegrenzt (vgl. auch BRANDON ET AL. 1999). Ab GRASS 5.1 ist die Vektordatenverarbeitung komplett erneuert und auf die Verwendung von Datenbanken (DBMS) erweitert.

GRASS ist sowohl als Binärversion für verschiedene UNIX-Plattformen als auch im originalen C-Programmcode via Internet und auf CD-ROM erhältlich. Ein sehr interessanter Aspekt ist die Lizensierung von GRASS unter der *GNU General Public License* (GPL, s.a. `http://www.gnu.org`). Sie macht GRASS zu einer frei zugänglichen, unverkäuflichen Software unter Wahrung der Autorenrechte. Jedoch können selbstverständlich kommerzielle Dienstleistungen mit GRASS erbracht werden. Damit reiht sich GRASS GIS in die *Open-Source*-Philosophie ein, die aus der Linux-Entwicklung bereits gut bekannt ist und Linux zum Durchbruch verholfen hat (vgl. zur „Open Source“-Thematik auch RAYMOND 1999).

Vom Konzept her betrachtet ist GRASS ein modular aufgebautes GIS. Das bedeutet, dass jede Datenbearbeitung mit einem eigenen Teilmodul durchgeführt wird. Dadurch gliedern sich die einzelnen „Abteilungen“ im GIS sehr klar und bringen Transparenz in die Arbeit.

1.2 Übersicht der GIS-Funktionalität in GRASS

Die im Folgenden aufgelisteten Funktionen stellen nur einen Auszug dar. Sämtliche Module sind in den „GRASS-Manpages“ beschrieben (siehe Internet).

Funktionalitäten im Bereich Vektoranalyse: Automatische Vektorisierung von Linien und Flächen, manuelle Digitalisierung am Bildschirm oder am Digitalisierbrett, Distanzberechnung, Höhenlinienberechnung aus Rasterhöhenmodellen, Interpolation (Splines), Konvertierung Vektor/Raster, Konvertierung Vektor/Punkt, Koordinaten-Transformationen, Reklassifikation, Vektorüberlagerung, Verschneidung von Flächen.

Funktionalitäten im Bereich Rasteranalyse: Zellen- und profilorientierte Abfragen, Höhenmodell-Analyse, automatische Raster-/Vektorkonvertierung, Beleuchtungsberechnung (Besonnung, Schattenwurf), Expertensystem (Bayesische Logik), Farbtabellen-Modifikation, Hangneigungs-/Expositionsberechnung, Geomorphologische Analysen (Profilkrümmung, Hangneigung und -exposition), Interpolationen für fehlende Zellenwerte (bilinear, IDW, kubisch, Splines), Klassifizierung, Konvertierung Raster/Vektor, Konvertierung Raster/Punkt, Korrelations-/Kovarianzanalyse, Kostenoberflächen (kumuliert), Berechnung des kürzesten Weges, Nachbarschaftsanalyse, Oberflächeninterpolation aus Vektorlinien und Punktdaten, Puffern von Punkten, Linien, Flächen, Rasterüberlagerung (gewichtet und ungewichtet), Regressionsrechnung, Reklassifikation, Resampling, Reskalierung, Sichtweitenanalyse (line of sight), statistische und geostatistische Auswertungen, Wassereinzugsgebiet-Berechnungen.

Funktionalitäten im Bereich Punktdatenanalyse: Convex hull-Berechnung, Geomorphologische Analysen (Profilkrümmung, Hangneigung und -exposition), Geostatistik, Oberflächeninterpolation aus Punkthöhen bzw. -werten, Thiessen-Polygone, Interpolation mit Splines, Triangulation (Delaunay, Voronoi).

Funktionalitäten im Bereich Bildverarbeitung (Image processing): Auflösungsverbesserung, Bildentzerrung (affin, polynomisch) auf Raster- oder Vektorgrundlagen, Farbkomposite, Fouriertransformation, Hauptkomponentenanalyse (PCA), Histogrammstreckung und -stauchung, Image Fusion, kanonische Komponentenanalyse (CCA), Kantenerkennung, Klassifikationen: (a) radiometrisch: unüberwacht, teilüberwacht und überwacht (Affinity, Maximum Likelihood), (b) geometrisch/radiometrisch: überwacht (SMAP), Kontrastverbesserung, Koordinatentransformation, IHS/RGB-Transformation, Orthofoto-Herstellung, Radiometrische Korrektur (Filterung), Resampling (bilinear, kubisch, IDW, Splines), Shape Detection, Zero crossing.

Visualisierungsmöglichkeiten: Animationen, 3D-Oberflächen, Bildschirm-Kartenausgabe, Farbzuweisung, Histogramm, Kartenüberlagerung (Raster-/Vektor-/Punktdaten), Postscript-Karten, Zoom-Funktion.

Integrierte Simulationsmodelle (direkt gekoppelt): Erosionsmodellierung (AGNPS 5.0, Answers, KINEROS), Hydrologische Analysen (Finite Elemente, SWAT (vom USDA), Kaskadenmodell CASC2D etc.), Landschaftsstrukturanalyse, Feuer-Simulation.

Weitere detailliertere Modulbeschreibungen zu rund 250 der über 400 bisher in GRASS integrierten Module können dem Anhang entnommen werden.

Das vorliegende Handbuch orientiert sich an den Grundfunktionen des Geographischen Informationssystems GRASS und beschreibt, nach einer Einführung in UNIX und GIS, den Aufbau einer

Datenstruktur in GRASS. Anschließend werden die Raster-, Vektor- und Punktdatenverarbeitung erklärt. Eine Betonung liegt dabei auf dem Import und Export von Daten, da in den seltensten Fällen sämtliche Daten am Rechner von Hand erzeugt werden. Häufig möchte man eine Kartengrundlage importieren und darauf seine Informationsebenen aufbauen. Ein Sonderkapitel stellt die Satellitenbildanalyse als „Spezialdisziplin“ der Rasterdatenverarbeitung dar, schrittweise wird in einem weiteren Kapitel die Herstellung von Orthofotos für den Luftbildbereich erläutert. GRASS bietet hier Möglichkeiten wie kaum ein anderes Standard-GIS, insbesondere im Hinblick auf die unrelevanten Kosten bei der Anschaffung. Das Kapitel Kartenausgabe ist recht knapp gehalten: Die Kartengestaltung in GRASS lässt sich derzeit nur mäßig gut durchführen. Daher wird auf das externe, ebenfalls frei verfügbare Programm „xfig“ verwiesen und seine Benutzung kurz angesprochen. Die Anbindung einer externen Datenbank ist prinzipiell möglich und für GRASS 5.0.x (Planung: 2000) in stabiler Form vorgesehen, sie wird in diesem Handbuch nicht angesprochen. Erste Entwicklungen liegen seit Januar 2000 auf dem GRASS-Server bereit. Seit 2001 wird an GRASS 5.1.x gearbeitet, inzwischen (2003) hat das Paket einen experimentellen, aber bereits benutzbaren Status erreicht.

GRASS erlaubt die Anwendung unterschiedlicher Koordinatensysteme und Projektionsarten. Zum einen können vordefinierte Projektionen verwendet, zum anderen auch nicht explizit aufgeführte Koordinatensysteme definiert werden.

Die unterstützten **Ellipsoide** sind (ab GRASS 5.0.x):

apl4.9, airy, andrae, australian, bessel nam., bessel, clark66, clark80, cpm, delmbr, engelis, everest, evrst56, evrst69, evrstSS, grs67, grs80, hayford, helmert, hough, iau76, international, kaula, krassovsky, lerch, mercury, modified airy, modified everest, merit, modified mercury, mprts, new international, plessis, SEasia, sgs85, sphere, walbeck, wgs60, wgs66, wgs72, wgs84.

Koordinatensysteme können in folgenden **Projektionen** definiert werden (hier die englischen Bezeichnungen):

Airy, Aitoff, Albers Equal Area, Apian Globular I, August Epicycloidal, Azimuthal Equidistant, Bacon Globular, Bipolar conic of western hemisphere, Boggs Eumorphic, Bonne (Werner mit lat=90), Cassini, Central Cylindrical, Chamberlin Trimetric, Collignon, Craster Parabolic (Putnins P4), Denoyer Semi, Eckert I-V, Eckert VI, Equal Area Cylindrical, Equidistant Conic, Equidistant Cylindrical (Plate Caree), Euler, Fahey, Foucaut, Foucaut Sinusoidal, Gall (Gall Stereographic), General Oblique Transformation, General Sinusoidal Series, Ginsburg VIII (TsNIIGAiK), Gnomonic, Goode Homolosine, Hammer & Eckert, Hatano Asymmetrical Equal Area, International Map of the World Polyconic, Kavraisky V+VII, Laborde, Lagrange, **Lambert Azimuthal Equal Area**, **Lambert Conformal Conic**, Lambert Equal Area Conic, Larrivee, Laskowski, **Latitude/Longitude**, Lee Oblated Stereographic, Loximuthal, McBryde-Thomas Flat-Polar (Sine No. 1 & 2, Parabolic, Quartic, Sinusoidal), Mercator, Miller Cylindrical, Miller Oblated Stereographic, Mod. Stereographics of 48 U.S., Mod.

Stereographics of 50 U.S., Mod. Stereographics of Alaska, Modified Polyconic, Mollweide, Murdoch I-III, Near, Nell, Nell-Hammer, New Zealand Map Grid, Nicolosi Globular, Oblated Equal Area, Oblique Cylindrical Equal Area, Oblique Mercator, Ortelius Oval, Orthographic, Perspective Conic, Polyconic (American), Putnins (P1-P3, P3', P4', P5, P5', P6, P6'), Quartic Authalic, Rectangular Polyconic, Robinson, Sinusoidal, Sanson, Space Oblique for LANDSAT, State Plane, Stereographic, Swiss. Oblique Mercator, Tilted perspective, Tissot, Transverse Central Cylindrical, Transverse Cylindrical Equal Area, **Transverse Mercator (für Gauß-Krüger)**, Two Point Equidistant, Universal Polar Stereographic (UPS), **Universe Transverse Mercator (UTM)**, Urmaev (Flat & V), Vitkovsky I, Wagner I (Kavraisky VI), Wagner II-VII, Werenskiold I, Winkel I+II, Winkel Tripel, van der Grinten I-IV.

2 Voraussetzungen für den Einsatz von GRASS

GRASS läuft auf einer Vielzahl von UNIX-Plattformen einschließlich GNU/Linux, Mac OSX, SUN Sparc und Ultra, DEC-Alpha, HP-UX, Silicon-Graphics, iPAQ und Zaurus Handhelds sowie unter MS-Windows-NT/2000/XP (mit Cygwin).

2.1 Hardware- und Software-Voraussetzungen

Die GRASS-Software (C-Quellcode) kann auf jeder UNIX-Plattform (PCs mit Linux, Mac OSX Rechner, Workstations unter UNIX-Derivaten von HP, SCO, SGI Indy, SUN (Solaris und SunOS), DEC Alpha etc.) übersetzt werden. Geeignet sind sowohl freie (z.B. GNU C-Compiler) wie auch proprietäre C-Compiler. Zur Übersetzung des Quellcodes werden mindestens 150MB Festplattenkapazität benötigt. Der Umfang von GRASS beträgt inzwischen über 1 Million Programmzeilen. Ein ganz großer Vorteil von GRASS gegenüber üblichen GIS besteht darin, dass das Programm im Betrieb aufgrund des modularen Aufbaus kaum Arbeitsspeicher benötigt. Damit steht entsprechend viel Platz für die zu verarbeitenden GIS-Daten zur Verfügung.

Interessant ist vor allem die „low cost"-Lösung, GRASS in Kombination mit Linux auf einem handelsüblichen PC (oder sogar Notebook) zu verwenden. Dabei sollte der PC mindestens ein 486er/-32MB RAM sein. Mehr Arbeitsspeicher ist generell besser als höhere Taktgeschwindigkeiten. Das UNIX-Betriebssystem Linux ist ebenfalls frei im Internet erhältlich oder alternativ über aufgearbeitete kommerzielle CD-ROMs. Soll MS-Windows auch auf dem PC laufen, wird Linux dabei parallel zu MS-Windows in einer eigenen Partition eingerichtet.

Für Linux selbst (incl. X11, dem UNIX-Window-System) ist mindesten 150MB zu reservieren. Die GRASS-Binärprogramme benötigen dank der guten Linux-Konzeption nur rund 50MB (gegenüber 150MB bei anderen UNIX-Derivaten). Der zu reservierende Platz für die 100MB Quellcode ist größer: mindestens 150MB. Es lohnt sich also zu Beginn, einfach die aktuelle Binärdistribution von GRASS aus dem Internet zu kopieren (Bezug: siehe nächster Abschnitt). Äußerst angenehm bei der Arbeit mit GRASS ist eine Dreiknopfmaus und ein Monitor mit einer Größe von mindestens 17". Insgesamt stellt sich diese Plattform als stabil und schnell dar.

Für Programmier-Interessierte sei angemerkt, dass Linux-Distributionen üblicherweise auch die benötigten Compiler (z.B. GNU C Compiler) enthalten.

Bezugsadressen

Seit August 2001 ist die jeweils neueste Version GRASS 5.0.x (bzw. 4.3 als letzte Produktion der 4.x-Serie) sowie inzwischen auch GRASS 5.1.x über das ITC-irst, Trento (Italien), online lieferbar:
GRASS Web Server:
`http://grass.itc.it/`

Dort liegt das Quellcode-Paket (geeignet für GNU/Linux, SUN Solaris, HP-UX, DEC-Alpha, MS-Windows (mit Cygwin) usw.) sowie das installationsfertige GRASS als Linux-x86-/MS-Windows-/SGI-Binaries in seiner jeweils neuesten Version bereit. Inzwischen sind rund 400 Module enthalten.

Auf dem Server befindet sich weiterhin das „GDP – GRASS Documentation Project“, mit dem die früher schwierige Suche nach Dokumentation v.a. zu extern entwickelten GRASS-Modulen erleichtert werden soll. Die Seite ist erreichbar unter:
`http://grass.itc.it/gdp/`

Einer der zwanzig Spiegelserver für GRASS liegt in den U.S.A.:
`http://grass.ibiblio.org/`
Außerdem werden Mailing-Listen angeboten, die Sie zur Teilnahme abonnieren können (subscribe). Mailing-Listen gibt es derzeit in Englisch (internationale User-Liste), Deutsch, Französisch, Italienisch und Japanisch. Die Anmeldeseite steht auf den GRASS-Seiten bereit („Support“).

Interessant für diejenigen, die an den neuesten Quellcode-Enwicklungen von GRASS 5 teilnehmen wollen und sich zumindest ein wenig mit Programmierung auskennen, ist das „GRASS-CVS“. In diesem elektronischen Verwaltungssystem (CVS: *Concurrent Versioning System*) wird der Quellcode seit Dezember 1999 weiterentwickelt. CVS erlaubt nach initialem Laden des Gesamtpakets, in weiteren Datentransfers allein die Verbesserungen herunterzuladen, um das Transfervolumen zu minimieren. Es koordiniert auch automatisiert die Verbesserungen an GRASS, die weltweit an das CVS gesendet werden. Sie finden weitere Informationen zu diesem Thema auf den GRASS-Servern.

2.2 Keine Angst vor UNIX!

Das Geographische Informationssystem GRASS ist eines der vielen heute verfügbaren Anwendungsprogramme unter UNIX. Es kann sowohl mit einer Maus als auch über menügesteuerte Kommandoeingaben bedient werden. GRASS läuft prinzipiell auf allen UNIX-Varianten (Derivate) und wird, wie oben beschrieben, für einige dieser Derivate installationsfertig angeboten.
Die Benutzung des Betriebssystems UNIX ist nicht schwierig zu erlernen - Sie brauchen nur wenige Befehle und Strukturen zu kennen, um Dateien verarbeiten oder Programme aufrufen zu können (z.B. GRASS). Die Kommandos, die in diesem Kapitel vorgestellt werden, besitzen für alle UNIX-

Systeme Gültigkeit. Sie können Ihr UNIX-Wissen damit also beispielsweise unter Linux, SUN Solaris oder HP-UX anwenden.

2.2.1 Was ist UNIX?

UNIX ist ein Betriebssystem mit langer Geschichte, das in den letzten Jahren seinen Weg von Großrechnern zu normalen PCs genommen hat. Die Entwicklung begann 1969. Es gab und gibt seitdem verschiedene parallele Entwicklungen, doch ist die UNIX-Welt einigermaßen standardisiert und kann aus Anwendersicht als homogen betrachtet werden. Verschiedene Hersteller bieten UNIX-Varianten an, „UNIX“ selbst ist ein geschütztes Warenzeichen, deshalb gibt es entsprechend der Hersteller eine größere Namensvielfalt. Neben „SUN Solaris“, „HP-UX“ und vielen weiteren Systemen verbreitet sich seit einigen Jahren immer stärker „GNU/Linux“. Diese freie UNIX-Variante, die auf handelsüblichen PCs eingesetzt werden kann, steht inzwischen sogar für die klassischen Workstations zur Verfügung. Linux lässt sich als direkter Konkurrent zu MS-Windows-basierten Oberflächen ansehen, da es in Preis – es wird frei via Internet verteilt bzw. ist zu geringen Kosten auf CD-ROM erhältlich – und Leistung MS-Windows-NT/2000/XP-Systeme in vielen Punkten übertrifft. Da verschiedene graphische Benutzeroberflächen für die Mausbedienung Bestandteil von Linux-Paketen sind, wird für die Arbeit unter Linux kein Expertenwissen mehr benötigt. Inzwischen stehen auch (sogar z.T. freie) Officepakete zur Verfügung, weitere Entwicklergruppen und Softwarehersteller sind dabei, aus der MS-Windows Welt bekannt Standardprogramme in ähnlicher Form für GNU/Linux bereitzustellen.

UNIX zeichnet sich dadurch aus, dass verschiedene Programme gleichzeitig benutzt werden und sogar mehrere Personen über ein Netzwerk zum selben Zeitpunkt auf einem Rechner arbeiten können. So kann aus einem PC eine „Workstation“ unter Linux mit optimaler Auslastung werden. Systemabstürze sind so gut wie unbekannt, da alle Programme und Personen voneinander geschützt arbeiten. Sollte ein Programm wirklich einmal „hängen“, kann es über einen speziellen Befehl aus dem Speicher entfernt werden, ohne dass das Gesamtsystem betroffen ist. Die lange Entwicklungsgeschichte von UNIX hat hier ein ausgereiftes Systemkonzept hervorgebracht. Es ist interessant zu beobachten, dass sich die MS-Windows-Varianten immer mehr dem UNIX-Konzept annähern.

Eine angenehme Begleiterscheinung besteht in der Quasi-Virenfreiheit, da hier das ausgereifte Schutzkonzept von UNIX Virenausbreitung verhindert. Es sind so gut wie keine UNIX-Viren bekannt. Bei vernetzten Systemen steht dagegen eher die Systemsicherheit im Vordergrund (Hackerangriffe), bei Einzelplatzrechnern ist diese Problematik glücklicherweise nicht von Bedeutung.

Im Folgenden werden einige Grundbefehle von UNIX beschrieben, die bei allen UNIX-Varianten Gültigkeit haben. Kennen Sie UNIX bereits, können Sie die nachfolgende Einführung auch überblättern und in Kapitel 3 wieder einsteigen.

2.2.2 Die Anmeldung im Rechnersystem

Wenn Sie an einem UNIX-Rechner arbeiten wollen, werden Sie vom „login“-Bildschirm begrüßt. Sie sehen ihn nach dem „Hochfahren“ Ihres Systems, beispielsweise Ihres Linux-PCs. Diese „login“-Prozedur wurde entwickelt, um die Nutzerdaten zu schützen. So ist das Rechnersystem auch für mehrere Personen zugänglich, die natürlich „eingerichtet“ sein müssen, also über die Systemverwaltung Festplattenplatz und Zugriffskennwort zugewiesen bekommen haben. Bei vernetzten Systemen sieht der Zugang genauso aus: Sie arbeiten an einem „Terminal“ und steuern einen anderen Rechner über Ihre Befehlseingaben. Via Internet kann dieser Rechner auch auf einem anderen Kontinent liegen – die Programmausgaben werden dann auf Ihren Bildschirm zurückübertragen (via *ssh* oder *telnet*-Verbindung). **Ganz wichtig ist, dass UNIX Groß- und Kleinschreibung unterscheidet.** Das gilt sowohl für den „login“-Prozess als auch für alle Befehlseingaben und Dateinamen. Darauf sollten Sie vor allem bei der Erzeugung von neuen Dateien achten.

Ein Wort zur Orientierung: In diesem Handbuch folgen nach den Dollarzeichen UNIX/GRASS-Kommandos, die direkt übernommen und eingegeben werden können.

Nach der erfolgreichen Anmeldung im Rechnersystem wird entweder automatisch das graphische UNIX-Oberflächensystem „X-Window“ (auch als X11 bezeichnet) gestartet, oder Sie müssen X-Window manuell aufrufen. Das erfolgt entweder über den Befehl

```
$ startx
```
oder
```
$ openwin
```

Es gibt für X-Window unterschiedlich gestaltete Benutzeroberflächen: Unter Solaris ist „openlook“ und „CDE“ verbreitet, unter Linux „fvwm“, „KDE“ (sieht MS-Windows ähnlich, vgl. Abb. 1), „GNOME“ usw. Häufig haben Sie die Möglichkeit, zwischen mehreren Varianten zu wählen.

Zentrale Elemente der Oberflächen sind die Fenster, die Informationen bereithalten oder zur Kommandoeingabe dienen. Das wichtigste Fenster ist das sogenannte „Terminalfenster“ – darin geben Sie Ihre Befehle ein. Viele Befehle werden Sie an MSDOS erinnern; UNIX war schon immer Vorbild für andere Betriebssysteme. Im Terminalfenster ist das „Promptzeichen“ zu sehen, das üblicherweise u.a. den Verzeichnispfad angibt, in dem Sie sich gerade befinden. Unter X-Window gibt es verschiedene Terminalfenster, die in ihren Eigenschaften voneinander abweichen. Zu nennen ist vor allem das „xterm“ (*Xterminal*), mit dem man sehr gut arbeiten kann. Mit der linken („markieren“, „copy“) bzw. mittleren („einfügen“, „paste“) Maustaste steht eine einfache Möglichkeit zur Verfügung, Dateinamen und Befehle zu übernehmen. Mit etwas Übung lassen sich so viele Befehlseingaben durch geschickten Maus-Kopiereinsatz minimieren. So können Sie auch Texte aus einem Fenster in einen Editor übernehmen, der in einem anderen Fenster geöffnet ist. Nicht benutzen sollten Sie die „kconsole“ unter KDE, da es hier Probleme mit den GRASS-Menüs geben kann.

Im Terminalfenster selbst läuft die sogenannte „shell“, der Kommandointerpreter, der Ihre Befehle entgegennimmt und an das Betriebssystem weiterreicht. Es gibt verschiedene „shells“, die bei der Einrichtung des Benutzernamens festgelegt werden. Man kann unter verschiedenen Möglichkeiten

wählen: Verbreitet sind die „csh“ (*C-Shell*), „ksh“ (*Korn-Shell*), „bash“ (*GNU Bourne-Again Shell*) und die „tcsh“ (*TC-Shell*). Alle „shells“ akzeptieren sämtliche Befehle, jedoch unterscheiden sie sich in ihren Eigenschaften bezüglich der Cursortasten-Steuerung und der automatischen Dateinamen-Komplettierung. Gerade letztere Eigenschaft ist sehr angenehm, da sich dadurch Schreibarbeit sparen lässt. Man braucht nur einige Anfangsbuchstaben eines Befehls oder eines Dateinamens einzugeben und komplettiert den Namen durch Tastendruck auf die Ergänzungstaste, sofern ein Befehl bzw. eine Datei mit diesen Anfangsbuchstaben vorhanden ist. Die Ergänzungstaste ist in der „tcsh“ die „ESC“-Taste (zweifaches Drücken), in der „bash“ ist es dagegen die Tabulatortaste. Alte Befehle können generell mit den „Cursor-Hoch“- und „Cursor-Runter“-Tasten editiert und benutzt werden. Die bei der Benutzereinrichtung voreinzustellende „shell“ wird immer automatisch gestartet, sobald man ein Terminalfenster öffnet.

Wenn Sie auf der X-Window-Hintergrundfläche die rechte oder linke Maustaste drücken, erscheint ein Menü. Darin werden verschiedene Programme zur Auswahl angeboten, eines wird ein solches Terminalfenster (z.B. „xterm“) sein. Im Terminal können Sie dann Ihre Befehle eingeben, einen Da-

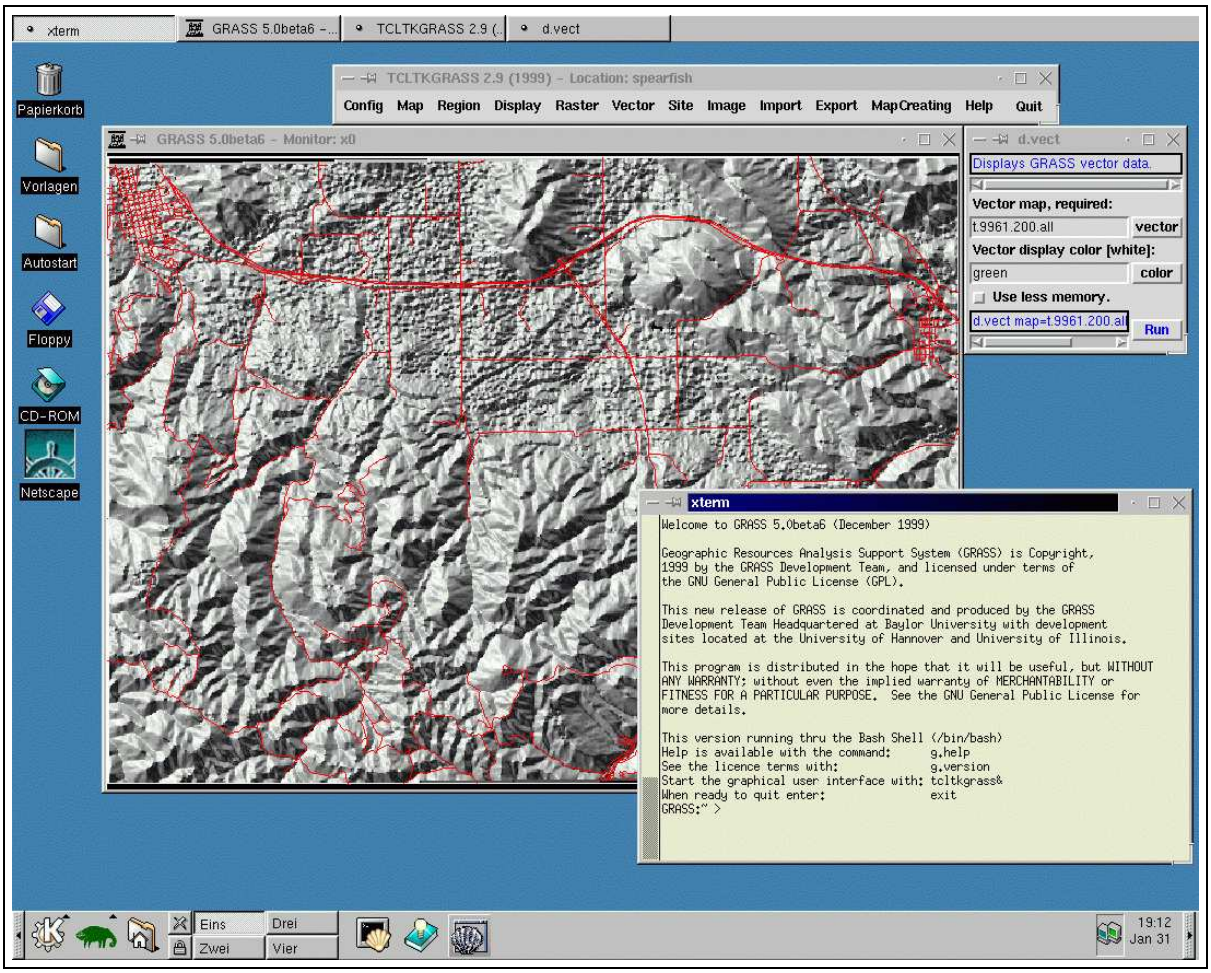


Abbildung 1: GRASS in der KDE-Umgebung unter Linux

teimanager starten usw. Bei den meisten Benutzeroberflächen ist am Bildrand auch eine Menüzeile mit graphischen Symbolen eingeblendet.
Ihre X-Window-Oberfläche bietet Ihnen neben dem Terminalfenster häufig auch einen elektronischen Briefkasten (*mailbox*) an, eine Uhr, ein kleines Fenster mit Angaben über die Systemauslastung, eine Menüzeile und ein „Konsolenfenster“. In letzterem erscheinen Botschaften des Betriebssystems: Die wichtigste Nachricht könnte hier eine Warnung sein, wenn einmal eine Festplattenpartition voll ist. Gerade im Umgang mit Geoinformationen werden Sie gelegentlich an die (Speicher-)Grenzen Ihres Rechners stoßen.
Sie kommen mit wenigen Befehlen aus, so dass UNIX nicht weiter kompliziert ist. Grundbefehle umfassen die Dateiverarbeitung und das Starten von Programmen. Die hohe Leistungsfähigkeit von UNIX (erkennbar auch an der Unmenge von Programmen, die generell existieren und sich teilweise bereits auf Ihrem Rechner befinden) werden Sie ganz von selbst erkennen.

2.2.3 Verzeichnisstruktur

Beginnen wir mit einem Befehl zur Anzeige der Dateien im aktuellen Verzeichnis (ls: listing, -l: Option „long listing“):

```
$ ls -l
```

Die angezeigte Liste könnte so aussehen wie in Abbildung 2 gezeigt. Auf den ersten Blick sieht es etwas komplizierter aus als beispielsweise unter MSDOS, doch benötigen Sie nicht unbedingt alle Informationen für Ihre tägliche Arbeit. Die erste Spalte (*permissions*) zeigt Informationen über Dateiart und Zugriffsrechte an. Die nächste Zahlenspalte gibt die Zahl der Unterverzeichnisse an, anschließend folgen Informationen über Besitzer/-in (*user*) und die Benutzergruppe (*group*). Benutzer werden bei UNIX einzelnen Arbeitsgruppen zugeordnet. Nach der Spalte über die Dateigrößen (*size*) folgen das Speicherungsdatum (mit Uhrzeit, *date*) und schließlich Dateinamen (*file*) bzw. Namen von Unterverzeichnissen (*directory*).
In unserem Beispiel erkennen Sie das aktuelle Verzeichnis an „ . “, das übergeordnete Elternverzeichnis an „ .. “. Bei „grassdata“ handelt es sich um ein Unterverzeichnis (directory), erkennbar am Buchstaben „d“ in der ersten Spalte. Die Datei „latex“ ist dagegen ein „link“ („l“ in der ersten Spalte), also ein Verweis auf eine andere Datei. Mit diesen „links“ können Sie platzsparende Verweise auf Dateien in anderen Verzeichnissen erzeugen, dabei ist diese neue „Datei“ normal benutzbar, obwohl sie physikalisch woanders liegt. „nations.ps“ ist nun eine „echte“ Datei (hier eine Postscriptdatei mit Endung „.ps“), erkennbar am Strich in der ersten Spalte.
Relevant für UNIX sind die Benutzungsrechte (*permissions*). Sie können damit festlegen, wer Ihre Dateien lesen bzw. verändern darf und ob es sich um ein ausführbares Programm (*executable*: ein bis drei „x“ in der *permissions*-Spalte) handelt. Die Rechte gliedern sich in Dreiergruppen, dabei werden die Attribute für Lesen, Schreiben, Ausführen für Sie selbst (*user*), für die Mitglieder Ihrer Gruppe (*group*) und die Mitglieder anderer Gruppen im Rechner (*other*) angegeben. Bei Verzeichnissen gibt das „x“ für Ausführen an, dass die Person in das Verzeichnis wechseln darf. Beispielsweise ist die Datei „nations.ps“ in Abbildung 2 für alle lesbar, kann aber nur durch „emil“

verändert werden. Die Verzeichnisse „mail“ und „projekte“ sind nur „emil“ zugänglich. Dagegen ist das Verzeichnis „grassdata“ für alle zugänglich, aber nur zum Lesen. Die Datei „ps4mf.txt“ kann von allen gelesen, aber nur von Mitgliedern der Gruppe „users“ verändert werden (also auch „emil“). Die Zugriffsrechte setzt UNIX üblicherweise automatisch korrekt, jedoch sollten Sie die Schreibrechte für andere Nutzer in Systemen mit mehreren Nutzern gegebenenfalls kontrollieren. Hier sehen Sie nun ein Beispiel für die Veränderungen der Dateiattribute mit `chmod`, auf `chown` und `chgrp` soll hier nicht weiter eingegangen werden:

```
$ chmod ug+w nations.ps
```

Damit wird die Datei „nations.ps“ auf die *permissions* `-rw-rw-r--` gesetzt, also für „emil“ und alle Mitglieder der Gruppe „users“ les- und schreibbar, für alle anderen Nutzer weiterhin nur lesbar.

2.2.4 Dateiorganisation

Dateien werden auch unter UNIX natürlich auf der Festplatte gespeichert. Allerdings sind die Festplatten in bestimmte Bereiche eingeteilt, in „Partitionen“. Jede Partition entspricht wiederum einem Verzeichnis, gegebenenfalls mit Unterverzeichnissen. Unter UNIX gibt es eine klassische Einteilung, sie ist bei jeder UNIX-Variante ähnlich (deshalb könnten Sie sich mit Linux-Kenntnissen auch auf einem UNIX-Großrechner zurechtfinden!).
Wesentliche Bereiche sind:

- das „root“-Verzeichnis (`/`): Diesem Verzeichnis sind alle Verzeichnisse untergeordnet wie beispielsweise der `/home`-Baum. Es ist nicht zu verwechseln mit der Person bzw. der Benutzergruppe „root“, die standardmäßig unter UNIX Administratorrechte haben.

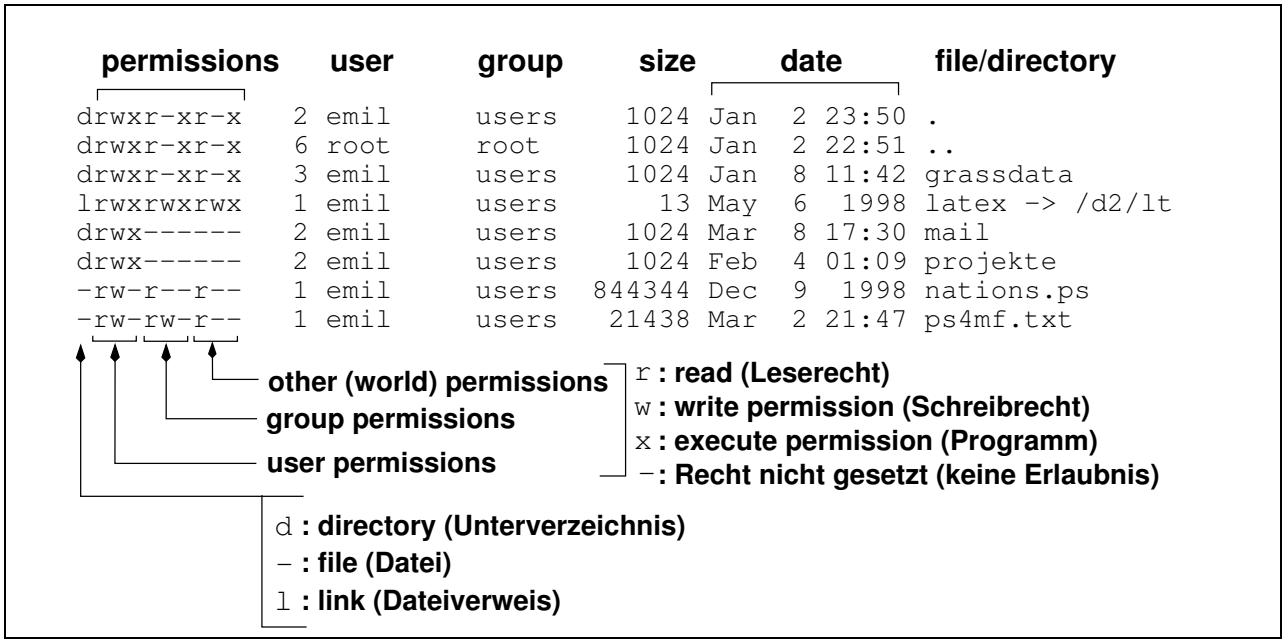


Abbildung 2: Aufbau eines Inhaltsverzeichnisses bei UNIX

- `/home`: Hier liegen die Verzeichnisse aller Nutzer, die auf dem Rechner eingetragen sind.
- Ihr Homeverzeichnis (z.B. `/home/emil/`): hier liegen Ihre persönlichen Dateien. Im Allgemeinen haben Sie nur hier die Möglichkeit, Daten zu speichern.
- `/usr`: In diesem Verzeichnisbaum sind Anwendungsprogramme, Hilfetexte usw. gespeichert.
- `/lib`: In diesem Verzeichnis sind Programmbibliotheken abgelegt, die von den Anwendungsprogrammen gemeinschaftlich genutzt werden. Sie entsprechen den „DLL“-Dateien bei MS-Windows-basierten Systemen. Eine „Registrierung“ gemäß MS-Windows entfällt.

2.2.5 Dateiverwaltung, Diskettenzugriff, Kopieren von CD-ROM

Möchten Sie eine Datei kopieren, werden Sie entweder einen Dateimanager verwenden oder den Kopierbefehl direkt in das Terminalfenster eingeben. Sie müssen dabei den Dateinamen angeben und das Ziel, wohin diese Datei kopiert werden soll (neuer Dateiname oder Zielverzeichnis):

```
$ cp <Quelle> <Ziel>
```

Also beispielsweise:

```
$ cp nations.ps projekte/
```

In diesem Beispiel wird die Datei „nations.ps“ vom aktuellen Verzeichnis in das Unterverzeichnis „projekte“ kopiert (vgl. Abb. 3). Gäben Sie statt „projekte“ das Elternverzeichnis „ .. “ an, würde die Datei nach „/home“ kopiert (sofern Sie dort schreiben dürften, was üblicherweise nicht erlaubt ist). Da das Elternverzeichnis hier aber dem Besitzer *root* gehört und, da *root* zur Benutzergruppe *root* gehört, das Schreibrecht für *other* (also „emil“, da „emil“ zur Gruppe *users* gehört) nicht gesetzt ist, kann das Kopieren in das Elternverzeichnis nicht erfolgen. Es klingt vielleicht etwas kompliziert, Sie werden aber schnell ein Gefühl dafür entwickeln, wie einfach der Vorgang ist. Außerdem meldet das UNIX-System Ihnen, wenn eine Aktion nicht erlaubt ist („*Permission denied*“ in diesem Fall).
Sie können Dateien auch verschieben oder umbenennen (move-Kommando):

```
$ mv <Quelle> <Ziel>
```

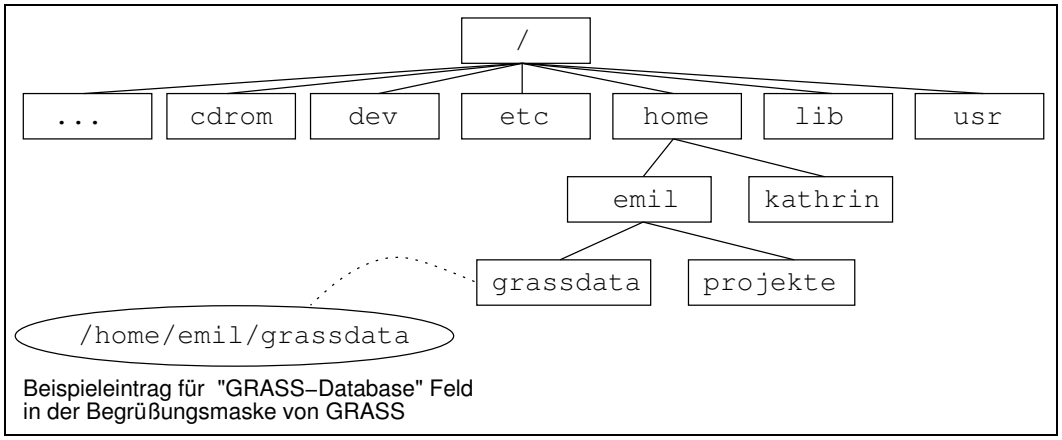


Abbildung 3: Aufbau eines Verzeichnisbaums bei UNIX

Dabei kann „Ziel“ ein Verzeichnis sein (Funktion: „Datei dorthin verschieben“) oder ein neuer Dateiname (Funktion: „Datei umbenennen“). Wollen Sie eine Datei löschen (*remove*-Kommando), geben Sie Folgendes ein:

```
$ rm -i <Dateiname>
```

Der Parameter „`-i`“ ist sehr wichtig, sofern er in Ihrem System nicht standardmäßig gesetzt ist, damit Sie nochmals gefragt werden, ob Sie wirklich löschen wollen. Unter UNIX sind Dateien nämlich wirklich gelöscht und können nicht mehr rekonstruiert werden. Und UNIX arbeitet schnell! Es gibt natürlich auch den Stern „ * “ und das Fragezeichen, die Sie als Platzhalter beispielsweise für Dateiendungen verwenden können. Gerade bei der Benutzung eines Platzhalters sollte man mit dem Löschbefehl vorsichtig sein.

Die meisten UNIX-Varianten (wie auch Linux) erlauben, normale MSDOS-Disketten zu lesen. Dazu gibt es die „mtools“, eine Sammlung von Programmen. Ist diese Sammlung auf Ihrem System installiert, ergänzen Sie die Ihnen vielleicht noch vertrauten MSDOS-Kommandos um ein „m“. Zwei Beispiele sollen diese Struktur dokumentieren:
Folgender Befehl zeigt das Disketteninhaltsverzeichnis an:

```
$ mdir a:
```

Der Befehl

```
$ mcopy a:<Dateiname> .
```

kopiert eine Datei von der Diskette in das aktuelle Verzeichnis („ . “). Mcopy verlangt immer ein Ziel, hier wurde der Punkt als Kürzel für das aktuelle Verzeichnis benutzt. Mit

```
$ mcopy <Dateiname> a:
```

kopieren Sie eine Datei vom UNIX-System auf die MSDOS-Diskette.
Wollen Sie Daten von einer CD-ROM lesen, wird es etwas umständlicher. CD-ROMs gehören wie auch Band- und Diskettenlaufwerke bei UNIX zu den sogenannten Geräten. Mit einem bestimmten Befehl können Geräte, wie hier die CD-ROM, in das System integriert werden. Mit

```
$ mount /dev/cdrom  /cdrom
```

wird die Dateistruktur der CD-ROM in das Verzeichnis `/cdrom` eingeblendet (vgl. Abb. 3). Sie benötigen für diesen Befehl allerdings Administratorrechte (mit `$ su` werden Sie Administrator, sofern Sie das Passwort kennen). Bei manchen Rechnersystemen werden CD-ROMs auch automatisch erkannt und integriert (*automount*-Funktion).

2.2.6 Programme starten

Sie können Programme (wie beispielsweise „GRASS“) starten, indem Sie den jeweiligen Programmnamen im „xterm“ eingeben. Einige Programme sind auch über das Menü der Oberfläche erreichbar, das Sie durch Drücken der rechten Maustaste in der Hintergrundfläche von X-Window öffnen können. Weiterhin erlauben vorhandene Dateimanager, durch Anklicken Programme zu starten.
Heutzutage verfügen die meisten Programme über ansprechende Benutzeroberflächen. Im allgemeinen kann man mit der Maus die installierte Software einfach bedienen.

Anleitungen zu fast jedem UNIX-Befehl sind über das „man“-Kommando abrufbar. So zeigt

```
$ man <Befehl>
```

die entsprechenden Erläuterungsseiten zum gewünschten Befehl an.

Wenn Sie ein Anwendungsprogramm, wie beispielsweise die graphische Benutzeroberfläche „tcltkgrass“ von GRASS, über den Aufruf im Terminalfenster starten, wäre an sich dieses Fenster blockiert. Mit der zusätzlichen Angabe des „ & “-Zeichens nach dem Programmnamen erreichen Sie dagegen, dass das Programm unabhängig vom Terminal gestartet wird und Sie damit weitere Eingaben im Fenster machen können. Das ist aber nur bei Anwendungen, die ihr eigenes Fenster haben, sinnvoll. Beispiel (Start eines Texteditors):

```
$ textedit &
```

Das Fenster des Texteditors wird sich öffnen, Sie haben aber das Terminalfenster frei für weitere Kommandos. Die Prozesse „textedit“ und „shell im Terminalfenster“ laufen nun parallel ab. Haben Sie einmal das „ & “-Zeichen vergessen, können Sie „`CTRL-Z`“ im Terminalfenster eingeben und dann

```
$ bg
```

Damit wird das „ & “-Zeichen dem vorherigen Befehl angefügt. GRASS dürfen Sie **nicht** mit dem „ & “-Zeichen starten, da Kommandos auch im Terminalfenster eingegeben werden.

Eine Spezialität von UNIX ist das sogenannte „Piping“. Über „pipes“ (Zeichen: `|`, `<`, `>`) werden Programmausgaben in ein anderes Programm als Eingabedaten umgeleitet. So lassen sich mehrere Programme direkt miteinander verknüpfen, ohne dass Daten in Dateien zwischengespeichert werden müssen. In GRASS können Sie diese „pipes“ beispielsweise einsetzen, wenn die Ausgabe (Datenstrom) bestimmter Module in eine Datei umgelenkt werden soll (`modul > datei.txt`). Sie können sich Pfeilspitzen vorstellen, die die Richtung des Informationsflusses angeben. Das Zeichen „ | “ dient der Datenübergabe zwischen Programmen, die Zeichen „ < “ und „ > “ der Umlenkung aus bzw. in Dateien. Damit ergeben sich große Möglichkeiten speziell für die Scriptprogrammierung, die Verfahrensabläufe automatisiert. Hier bietet gerade GRASS als modulares GIS ein sehr großes Potential (vgl. Abschnitt 13.1).

Prinzipiell können Sie unter UNIX beliebig viele Programme (also auch Terminalfenster) gleichzeitig benutzen, allein durch die Speichergröße des Rechners und die benötigte Rechenzeit sind Grenzen gesetzt.

Abschließend folgen noch einige praktische Hinweise:

- Mit „SHIFT-Bildhoch/Bildrunter“ können Sie im xterm blättern (d.h. alte Befehle sehen).

- Es ist bei der GIS-Arbeit sinnvoll, Protokolle mitzuschreiben. Mit „xedit“ können Sie per „Copy-Paste“ Ihre Befehle in einer Textdatei speichern. Markieren Sie Texte (Befehle) am besten zeilenweise mit linker Maustaste, das Einfügen erfolgt in „xedit“ mit der mittleren Maustaste.

- Wollen Sie die Schrift des xterms vergrößern: „STRG“ + rechte Maustaste im xterm gedrückt halten und im nun erscheinenden Auswahlmenü „large“ oder „huge“ gleichzeitig mit der linken Maustaste als Schriftgröße wählen.

2.2.7 Die Arbeit beenden: UNIX verlassen

Bevor wir gleich in die Arbeit mit GRASS einsteigen, soll noch das korrekte Beenden einer UNIX-Sitzung beschrieben werden. Der Rechner darf nicht einfach ausgeschaltet werden (wie es ja auch bei anderen Betriebssystemen der Fall ist), sondern wird „heruntergefahren“. Zunächst sind geöffnete Anwendungsprogramme zu schließen (gegebenenfalls veränderte Dokumente speichern), dann sollten geöffnete Terminalfenster geschlossen werden. Prinzipiell bekommen Sie durch das Drücken der rechten Maustaste auf den Fensterrahmen (z.B. „xterm“-Fenster) ein Menü, das einen „Quit“-Knopf enthält. Doch um sicherzugehen, dass keine unbeendeten Programme mehr laufen, sollten Sie stattdessen „exit“ in das Terminalfenster eingeben. Es schließt sich dann automatisch, wenn keine Programme innerhalb dieses Fensters mehr aktiv sind. Sie werden merken, dass diese Vorgehensweise gerade bei Netzwerkverbindungen sehr ratsam ist. So sind Sie sich immer bewusst, auf welchem Rechner Sie gerade arbeiten. Anschließend wird die X-Window-Oberfläche geschlossen. Dazu gibt es einen „Exit“- oder „Logout“-Knopf im Menü der Oberfläche. Nun sollten Sie wieder im Textmodus des Terminals sein, in dem Sie die Oberfläche aufgerufen hatten. Mit „exit“ loggen Sie sich aus dem System aus.

Möchten Sie den Rechner nun komplett abschalten, dürfen Sie auch jetzt keinesfalls den Netzschalter betätigen. Sie sind zwar nicht mehr eingeloggt, doch arbeitet das UNIX-System normal weiter und wartet auf die nächste Anmeldung! Sie müssen sich als Administrator anmelden („root“ als Loginname) und können dann das „shutdown“- oder „halt“-Kommando absetzen (bitte prüfen Sie mit dem Befehl `$ w` bei vernetztem Rechner, ob noch jemand auf Ihrem System arbeitet). Erst, wenn der Rechner richtig „heruntergefahren“ ist, darf der Netzschalter betätigt werden. Bei manchen Linux-Systemen ist die Tastenkombination „Alt“-„Ctrl“-„Del“ (bzw. „Alt“-“Strg“-“Entf“) auch mit dem Shutdown-Kommando belegt, so dass Sie sich nicht als „root“ einloggen müssen.
Diese Vorgehensweise für das Abschalten des Systems ist insbesondere für Linux-Anwender und Anwenderinnen wichtig, die allein zuhause oder im Büro an einem Rechner arbeiten und ihn nicht ununterbrochen benutzen oder beispielsweise zu einem alternativen Betriebssystem wie MS-Windows-System umschalten möchten.
Auf der folgenden Seite finden Sie eine Übersicht der wichtigen UNIX-Dienstprogramme. Es gibt folgende Varianten der Kommandostruktur:

Typisch für den Aufruf eines Anwendungsprogramms wie z.B. GRASS:

```
kommando
```

Typisch für den Aufruf von UNIX-Dienstprogrammen:

```
kommando dateiname
```

Typisch für den Aufruf von UNIX-Dienstprogrammen mit Optionen:

```
kommando optionen dateiname
```

Vorschläge zum Weiterlesen über UNIX

Krienke, R. (1996): UNIX für Einsteiger. Eine praxisorientierte Einführung. München.

Leibner, P. (1998): Einführung in das Betriebssystem UNIX. Münster.

Kommando	wichtige Option(en)	Bedeutung des Kommandos	Bedeutung der Option(en)/Bemerkung
`cat`	`datei`	Textdatei ausgeben	
`cp`	`quelle ziel`	Datei kopieren	`ziel` kann neuer Dateiname oder Verzeichnis sein
`df`	`-k partition`	freien Plattenplatz anzeigen	`-k`: Angabe in Kilobyte, Punkt für akt. Verzeichnis
`exit`		Terminalsitzung/Programm beenden	z.B. zum Beenden von GRASS einzugeben
`file`	`datei`	Dateityp ausgeben	zeigt, ob ASCII oder Binärdatei vorliegt
`ftp`	`rechnername`	Dateitransfer via Netzwerk	Rechnername: Internetadresse
`grep`	`zeichenkette datei`	Textdatei nach Zeichenkette durchsuchen	Zeichenkette sollte in Hochkomma stehen
`gunzip`	`datei`	Datei mit Endung „ .gz “ entpacken	
`gzip`	`neuespaket.gz datei`	Datei komprimieren	
`head`	`-n datei`	Kopfzeilen einer Textdatei anzeigen	`-n`: für n Anzahl Zeilen einsetzen
`kill`	`-9 pid`	abgestürzten Prozess entfernen	`-9`: Entfernung erzwingen, `pid` über `ps` ermitteln
`lpr`	`-P druckername datei`	Datei auf Drucker ausgeben	
`ls`	`-l` bzw. `-la`	Verzeichnisinhalt anzeigen (listing)	`-l`: lang, `-la`: mit versteckten Dateien
`man`	`programmname`	Online-Manual	Aufruf: `man programmname`
`mkdir`	`verzeichnis`	Verzeichnis erzeugen	
`more`	`asciidatei`	seitenweise Ausgabe einer Textdatei	nach Anhalten weiter mit <return> zur nächst. Seite
`mv`	`-i quelle ziel`	Datei/Verz. verschieben oder umbenennen	`quelle` und `ziel`: neuer Dateiname oder Verz.
`passwd`		eigenes Passwort ändern	
`ps`	`-aef` (oder `-aux`)	Prozesse anzeigen	Prozessdaten anzeigen, speziell `pid` (*Process-ID*)
`pwd`		Aktuelles Arbeitsverzeichnis anzeigen	
`rm`	`-i datei`	Datei löschen	`-i`: Sicherheitsabfrage
`rm`	`-ir verzeichnis`	Verzeichnisstruktur komplett löschen	`-r`: rekursiv mit Unterverzeichnissen löschen
`rmdir`	`verzeichnis`	leeres Verzeichnis löschen	
`script`	`logdatei`	Protokollierung einer Sitzung in einer Logdatei	Beenden mit `CTRL-D`
`tail`	`-n datei`	Endzeilen einer Textdatei anzeigen	`-n`: für n Anzahl Zeilen einsetzen
`tar`	`-xvf paket.tar`	Dateien mit Verzeichnisstruktur entpacken	
`tar`	`-cvf neuespaket.tar quelle`	Dateien mit Verzeichnisstruktur packen	Nicht `-c` und `-x` verwechseln, Überschreibgefahr!
`telnet`	`rechnername`	Terminalprogramm für externe Rechner	Rechnername: Internetadresse
`which`	`datei`	Datei in Pfad suchen	nützlich, um Kommando auf Festplatte zu finden
`zip`	`neuespaket.zip datei`	Datei komprimieren	mit Zusatzoption `-k` wird PKZIP-Kodierung erreicht

3 GRASS als Geographisches Informationssystem

Ein Geographisches Informationssystem (GIS) wie GRASS ist eine umfassende Sammlung von Werkzeugen zur Sammlung, Speicherung, Abfrage, Transformation und Darstellung räumlicher Daten der realen Welt (nach BURROUGH ET AL. 1998, S. 11). Dabei werden die digital kodierten Phänomene und Objekte der realen Welt primär anhand ihrer geographischen Lage gespeichert und lassen sich somit zueinander in Bezug setzen.

3.1 Die Verwaltung geographischer Daten

Die im Geographischen Informationssystem zu speichernden Phänomene und Objekte (auch als Entitäten bezeichnet) treten in zwei wesentlichen Strukturen auf. Es gibt

- kontinuierliche Erscheinungen, die flächenhaft und quasi unbegrenzt im Raum sind (z.B. Temperatur der untersten Luftschicht) und
- diskrete Erscheinungen, d.h. zum einen abgrenzbare Flächen (z.B. Seen oder Gebäude), zum anderen linienhafte Phänomene (z.B. Straßen oder Eisenbahnstrecken) sowie punkthafte Informationen (z.B. geologische Bohrdaten).

Für diese in ihrem Charakter sehr unterschiedlichen Strukturen wurden unterschiedliche Datenstrukturen geschaffen, um diese Daten mit möglichst geringen Informationsverlusten im Rechner digital speichern zu können.
Bei der Auswahl einer geeigneten Datenstruktur kommt es immer auf den Maßstab und damit auf die gewünschte Auflösung der Daten an. Ein Fußballfeld sieht aus dem All wie ein Punkt aus, aus direkter Nähe betrachtet wird es dagegen zur abgegrenzten Fläche.

In Geographischen Informationssystemen gibt es vier wesentliche Gruppen von Datenstrukturen (vgl. Abb. 4). Die ersten drei Datentypen sind Lagedaten, beim vierten Typus handelt es sich um die zugehörigen Sachdaten:

Rasterdaten: Sie werden vor allem für kontinuierlich im Raum verteilte Daten eingesetzt. Dabei wird eine regelmäßige Matrix quadratischer und gleich großer Zellen erzeugt. Jede Zelle bekommt ein Attribut (Eigenschaft, Sachdatum) zugewiesen, das das zu speichernde Phänomen repräsentiert (z.B. einen Temperaturwert). Die Speicherung der Zellen erfolgt über ihre Koordinaten. Die Zellen lassen sich aufgrund der Matrixstruktur in Reihen (*rows*) und Spalten

(*columns*, *cols*) gliedern, der Datenzugriff kann demnach über die geographischen Koordinaten oder über die Angabe der Reihe/Spalte geschehen.

Vektordaten: Sie werden zur Speicherung von Linieninformationen bzw. bei geschlossenen Linienzügen (Polygone) zur Speicherung homogener Flächen benutzt. Eine Linie verbindet jeweils zwei Endpunkte, die wiederum Koordinaten besitzen, und hat ein oder mehrere Attribute (Eigenschaften, Sachdaten).

Punktdaten: Sie dienen zur Speicherung unregelmäßig im Raum verteilter Informationen. In manchen GIS wie z.B. GRASS ist dieser Datentyp alternativ auch in Form von Vektorpunkten speicherbar. Jeder Punkt hat seine Koordinaten und ein oder mehrere Attribute (Eigenschaften, Sachdaten).

Sachdaten: Bei Sachdaten handelt es sich um die Attribute, die mit obigen Datensätzen verknüpft sind. Ihre Speicherung erfolgt häufig in einem an das GIS gekoppelten Datenbanksystem bzw. in begrenztem Maße auch im GIS selbst.

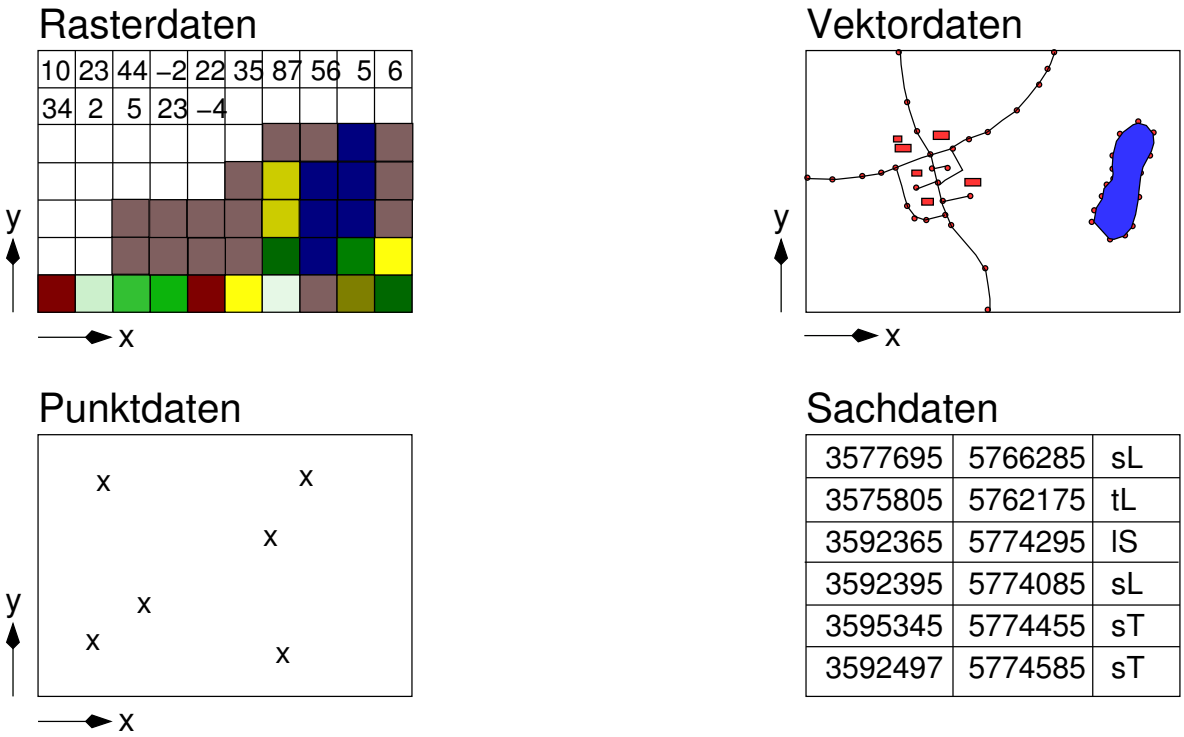


Abbildung 4: Datenstrukturen im GIS:
Rasterdaten: Darstellung der Attribute als Zellenwerte bzw. Zellenfarben
Vektordaten: linien- und flächenhafte Informationen sind mit Attributen verknüpft, Vektorlinien über Knotenpunkte
Punktdaten: mit Attributen verknüpft
Sachdaten: Speicherung der Attribute mit Koordinaten in Tabellenform in einer Datenbank

Die Datenstrukturen im Geographischen Informationssystem sind nicht als unveränderlich anzusehen, es gibt Konvertierungsmöglichkeiten zwischen den verschiedenen Formaten. Beispielsweise werden Höhenlinien, die aus einem flächenhaften Geländemodell (im Rasterformat) abgeleitet werden, als Vektorlinien abgelegt. Im Gegenzug wird bei der Interpolierung einer geschlossenen Geländeoberfläche aus digitalen Höhenlinien (im Vektorformat) die Karte im Rasterformat gespeichert. Die Konvertierung zwischen den Datenstrukturen läuft intern im jeweiligen GIS-Modul ab. Je nach Auflösung ist ein mehr oder weniger starker Datenverlust unvermeidbar (vgl. Abb. 5).
Daten in heutigen Geographischen Informationssystemen wie GRASS oder ARC/INFO sind, wie Abbildung 6 zeigt, überwiegend zweidimensional (2D) oder zweieinhalbdimensional (2.5D). Im zweidimensionalen Fall liegen flächenhafte Informationen vor. Ist eine dritte Information vorhanden, spricht man von zweieinhalb Dimensionen. Ein Beispiel dafür sind Höhendaten: Hier existieren Höheninformationen zusammen mit Positionsdaten (x-, y-, z-Daten), es fehlen aber die Beschreibungen für die „Seiteninformationen“. Erst bei echten 3D-Systemen werden auch Beschreibungen für die Seitenflächen von Körpern (z.B. bei Gebäudeoberflächen oder Bodenprofilen) gespeichert. Zur Zeit gibt es eine starke Tendenz zu diesen echten 3D-GIS mit Blickrichtung auf 4D-GIS unter Berücksichtigung der Zeitkomponente. GRASS 5.0.x weist ein neues Datenformat für dreidimensionale Rasterdaten und ein erweitertes Punktdatenformat auf (vgl. BRANDON ET AL. 1999). Ab GRASS 5.1.0 ist auch die Verarbeitung von 3D Vektordaten mit DBMS-Integration möglich.

Verwaltung von Rasterdaten

Bei Datenspeicherung in Rasterstruktur wird das Projektgebiet in quadratische Rasterzellen gleicher Größe aufgeteilt. Die Daten sind damit in einer Datenmatrix abgelegt. Die Speicherung flächenhafter Daten kann sowohl kontinuierlich als auch arealhaft mit eingefügten Nullwerten („no data“) sein. Im ersten Fall liegt für jede Rasterzelle ein Datenwert vor, im zweiten Fall nur für bestimmte Flächenausschnitte. Jede Zelle kann einen anderen Wert aufweisen, der Zelleninhalt ist homogen. Sie deckt also eine geographische Fläche gemäß der gewählten Rasterauflösung (z.B. $12\text{m} * 12\text{m}$) ab. Da die Größe der Rasterzelle die Grundeinheit des Rasterformats darstellt, bestimmt die Auflösung somit den Informationsgehalt bzw. die Datenqualität. Zweidimensionale Rasterzellen werden als „Pixel“ (*Picture Element*) bezeichnet, dreidimensionale Zellen als „Voxel“ (*Volume Pixel*). Typische Daten, die im Rasterformat gespeichert werden, sind Bilddaten wie Luft- und Satellitenbilder oder regelmäßig verteilte Höheninformation. Auch gescannte Karten sind Rasterdaten.

Die Vorteile des Rasterformats liegen in der Möglichkeit, mit mathematischen Rechenoperationen mehrere Rasterkarten zellenweise verknüpfen und so neue Karten ableiten zu können. Es lassen sich algebraische Funktionen und Boolesche Logik anwenden sowie Bedingungen formulieren. Die Rasterzellen der unterschiedlichen Karten werden gemäß ihrer geographischen Lage miteinander verrechnet. Auch Transportprozesse sind im Raum gut modellierbar, da Informationen sehr einfach an Nachbarzellen weitergegeben werden können.

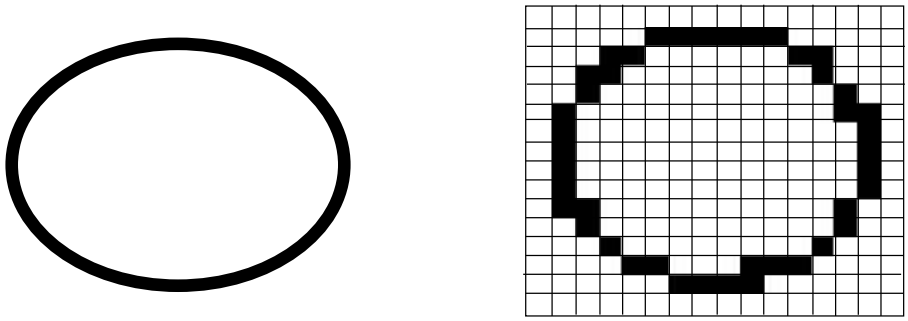


Abbildung 5: Vergleich der Auflösung einer Ellipse im Vektor- bzw. Rasterformat

Die Nachteile des Formats bestehen in der je nach gewählter Auflösung ungenauen Abbildung gekrümmter Flächen („Treppeneffekt“, vgl. Abb. 5) und der in den meisten Geographischen Informationssystemen wie auch GRASS kartenweise homogen zu wählenden Rasterauflösung. Außerdem entstehen sehr schnell große Datenmengen. BURROUGH ET AL. (1999, S. 184) sieht allerdings diesen Nachteil dank stark gestiegener Rechnerkapazitäten als vernachlässigbar an.

Verwaltung von Vektordaten

Im Vektorformat werden diskrete (abgrenzbare) Daten wie Grundstücke, Straßen, Flüsse usw. entweder anhand ihrer Grenzen oder anhand der Mittelpunkte/-linien abgespeichert. Es gibt folgende Elemente bei den Vektordaten:

- Punkte: werden zur Standortdarstellung eingesetzt (z.B. Lage eines Bohrprofils) bzw. sind Verknüpfungspunkte (Knoten, *vertices, nodes*) von Vektorlinien. In manchen GIS wie z.B. GRASS können sie auch parallel als eigenes Format (Punktformat: „Sites-Format“) existieren.

- Linien: dienen der Darstellung von linienhaften Informationen (v.a. Verkehrswege) oder in Form geschlossener Linienzüge (Polygone) als Grenzlinien von Flächen. Je nach Maßstab werden entweder Grenzlinien oder nur die Mittelpunktlinien dargestellt.

- Flächen: sind von geschlossenen Linienzügen umgebene arealhafte, homogene Objekte.

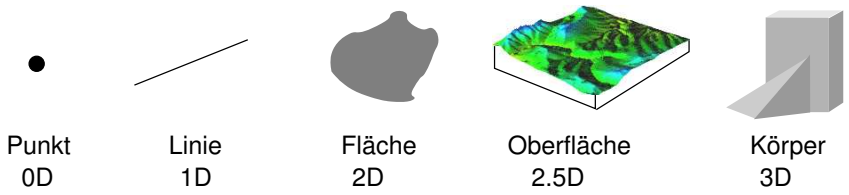


Abbildung 6: Datendimensionen im GIS, nach RASE (1998, S. 19)

Im GIS erfolgt beim Vektorformat eine Speicherung von Punkten (Knoten) und Verbindungslinien zur Beschreibung der Linien oder Flächen bzw. bei Punktinformationen als singuläre Knoten. In GRASS ist der jeweile Typ (Punkt, Linie, Fläche) vor der Datenerzeugung zu wählen.
Vektorflächen sind homogene Flächen, sie haben aber im Gegensatz zu Rasterflächen frei definierbare Grenzen und sind keinen Auflösungseinschränkungen unterworfen. Mit Vektorflächen kann also eine exakte Formbeschreibung stattfinden. Da jede Vektorlinie über ihre Knoten (*nodes*) Positionskoordinaten besitzt, ist sie genau im Raum lokalisiert.
Bei einigen GIS wie GRASS ist eine exakte Formbeschreibung gekrümmter Linien und Flächen allerdings dadurch leicht eingeschränkt, dass eine Kurve durch zusammengesetzte Geraden (Linienzüge) erzeugt wird. Andere GIS haben dagegen zusätzlich Bögen (*arcs*) oder Splines zur Kurvenerzeugung als Vektorelement. In GRASS sollten die aus Geraden zusammengesetzten Kurven keine zu langen Strecken aufweisen.

Im GIS werden Geometrie und Topologie unterschieden. Die *Geometrie* gibt die Position eines Objekts im Raum an, die *Topologie* die relative Lage von Teilobjekten *zueinander*. Letztere ist quasi die Beschreibung, wie Objektteile bzw. verschiedene Objekte zueinander im Raum gelagert sind. Die Topologieinformationen werden intern im GIS aufgebaut.
Vektordaten zeichnen sich durch die wesentliche Eigenschaft aus, dass im GIS ihre Topologie berechnet und gespeichert wird. Über diese topologischen Informationen können dann unter anderem folgende Fragestellungen beantwortet, also mit GIS-Modulen aus dem Vektordatensatz ermittelt werden (BARTELME 1995, S. 18):

- Nachbarschaftsbeziehungen zwischen Objekten
- Enthaltensein eines Objektes in einem anderen (Punkte bzw. sogenannte „Inselflächen“)
- Überschneidungen von Objekten
- Nähe zweier Objekte zueinander

Für topologische GIS wie GRASS bedeutet die Berücksichtigung der Topologie in der Praxis, dass angrenzende Flächen eine gemeinsame Grenzlinie aufweisen und nicht als zwei übereinanderliegende Linien beschrieben werden. Daraus ergibt sich ein wesentlicher Vorteil gegenüber Desktop-GIS wie ARCView, die über diese Methode der Datenverwaltung nicht verfügen.

Die Vorteile des Vektorformats liegen in der guten, maßstabsunabhängigen Darstellung diskreter Raumerscheinungen, da die Flächen- und Linienformen annähernd exakt dargestellt werden können. Es sind lediglich Linien- und Knoteninformationen neben den Attributen zu speichern, so dass die anfallenden Datenmengen im Vergleich zu Rasterdaten sehr gering sind. Bei flächenhaften Informationen können die Datenmengen auch über die Methode der Isoliniendarstellung stark reduziert werden. Die Darstellung von Objekten erfolgt eindeutig. So lassen sich später objektbezogene Abfragen im GIS durchführen.

Das Vektorformat ist allerdings für die Speicherung quasi-kontinuierlich veränderlicher Flächendaten (wie z.B. Bild- oder Höhendaten) ungeeignet, da es immer homogene Flächen (oder Punkte) abbildet. Auch sind Transportprozesse im Raum kaum darstellbar. Geometrische Verschneidungen verschiedener Vektorkarten (Ermittlung von Flächen mit bestimmter Attributkombination) sind sehr aufwendig und weisen das Problem auf, dass durch kaum vermeidbare Digitalisierfehler eine Vielzahl kleinster Flächen entstehen kann, die später manuell korrigiert werden müssen. Für räumliche Verteilungsanalysen ist das Vektorformat ungeeignet, da die Vektorflächen in sich homogen sind. Für derartige raumdifferenzierte Modellierungen bietet das Rasterformat wesentlich bessere Möglichkeiten. Nach BURROUGH ET AL. (1999, S. 184) überwiegen bei räumlichen Analysen die Vorteile des Rasterformats gegenüber denen des Vektorformats. GRASS unterstützt beide Formate.

Verwaltung von Punktdaten

Punkthafte Daten können beispielsweise Messergebnisse sein, die stichprobenartig im Raum aufgenommen wurden. In diesem Format lassen sich Messpunkte aller Art speichern: Klimastationsdaten, Höhendaten, Bohrdaten etc. Punktdaten werden häufig mit geostatistischen Analysemethoden ausgewertet. Üblicherweise sind diese Stichproben unregelmäßig im Raum verteilt. Durch geostatistische Methoden können dann die fehlenden Werte für eine geschlossene räumliche Verteilung im Rasterformat interpoliert werden.
Bei manchen GIS wie auch GRASS stellen sie zusätzlich zu den Vektorpunkten ein eigenes Format dar, bei anderen sind sie mit den Vektorpunkten identisch. In GRASS werden Punkte als *Sites* (entsprechend: „Sites-Format“) bezeichnet. Diese *Sites* sind direkt in das Vektorformat als Vektorpunkte bzw. vom Vektorformat in das „Sites-Format“ konvertierbar. Eine Reihe von geostatistischen Modulen in GRASS basieren auf diesem Punktformat.

Verwaltung von Sachdaten – GIS und Datenbanken

Sachdaten sind die Informationen, die an die Koordinateninformationen geknüpft werden. Es handelt sich also um thematische Daten, die sogenannten „Attribute“. Sie bilden damit eine eigene Kategorie. Sachdaten werden in einer Datenbank abgelegt, die zusätzlich auch die entsprechenden Koordinaten oder Objektnummern als Bezug gespeichert hat. Diese Datenbank kann entweder GIS-intern sein oder extern über eine Schnittstelle gekoppelt. Für GRASS sind voll funktionsfähige Schnittstellen zu externen Datenbanken (*Oracle, Informix, PostgreSQL* etc.) für die Version GRASS 5.1 geplant. In der GRASS-internen Datenbank lässt sich pro Karte leider nur ein Attribut pro Vektorobjekt bzw. Rasterzellenwert verwalten.

GRASS als hybrides GIS

In der Regel gibt die Datenlage die kartenweise zu wählende Datenstruktur vor. Prinzipiell gleichen sich die Vor- und Nachteile des Raster- bzw. Vektordatenformats gegenseitig annähernd aus,

so dass bei der richtigen Formatwahl nur wenig Probleme auftreten sollten. Im Vorfeld eines GIS-Projektes ist demnach zu prüften, in welchem Format die jeweiligen Datensätze zu speichern sind, sofern überhaupt Wahlmöglichkeiten bestehen. Da Konvertierungsmodule existieren, können unter Berücksichtigung etwaiger Informationsverluste Formate auch geändert werden. Eine allgemeingültige Aussage, welches Format besser ist, lässt sich aufgrund der unterschiedlichen Zielsetzungen nicht treffen. In den meisten Fällen geben die zu integrierenden Rohdaten das Format vor.

3.2 GIS-Konzepte

Um die Einordnung von GRASS innerhalb der Vielfalt an heute existierenden Geographischen Informationssystemen zu verdeutlichen, soll im Folgenden eine kurze Gliederung üblicher GIS-Konzepte erfolgen.
Es gibt verschiedene Möglichkeiten, räumlich verteilte Daten in einem Geographischen Informationssystem zu organisieren. Der Aufbau der Datenorganisation ist abhängig von der Struktur der räumlichen Daten. Die bereits vorgestellten Raster-, Vektor- und Punktdatenformate sind in den meisten GIS-Konzepten vertreten.

Layer-GIS

Im Layer-GIS werden geographische Informationen in verschiedenen Ebenen abgelegt, die geokodiert sind. Beispielsweise haben topographische Karten unterschiedliche Informationsebenen: Es gibt Waldflächen, Verkehrswege, bebaute Flächen, Wasserflächen etc. Im Layer-GIS wird jede Informationsschicht in Anlehnung an kartographische Ansätze einzeln abgelegt. Da diese Schichten räumlich orientiert sind, kann durch ihre Überlagerung in diesem Beispiel wieder eine topographische Karte aggregiert werden. Dieses GIS-Modell entspricht sozusagen „fliegenden Teppichen“. GRASS ist wie die meisten handelsüblichen Geographischen Informationssysteme hier einzuordnen.
Der Datenaustausch ist im Layer-GIS sehr einfach; Raster-, Vektor- und Punktdaten lassen sich über Schnittstellen im- und exportieren. Jede externe Datei entspricht dabei einer Informationsschicht im GIS, speicherbar im Vektor-, Raster- oder Punktformat. Der Zugriff auf einzelne Informationen erfolgt durch Angabe der Koordinaten und der abzufragenden Informationsebene, das GIS liefert dann die dort gespeicherte Information. Verschneidungen sind einfach durchzuführen, die Berechnung von neuen Schichten erfolgt unter Verwendung vorhandener Informationsschichten.
Alle Daten können im jeweils optimalen Format abgelegt werden: Kontinuierliche Daten werden im Rasterformat gespeichert, lokal homogene Flächendaten oder linienhafte Verbindungen (z.B. Verkehrswege) dagegen im Vektorformat. Messdaten werden als Punktinformationen abgelegt. Konvertierungen zwischen den Datenstrukturen sind möglich. Eine topographische Karte kann beispielsweise gescannt und als Rasterdatei importiert werden. In einem zweiten Schritt lassen sich relevante Informationen dann manuell oder automatisiert vektorisieren („digitalisieren“, ein etwas

unscharfer Begriff). Vektordaten können bei Kopplung an ein Datenbanksystem zur Speicherung von Attributinformationen Objektcharakter haben.

Der Nachteil der Layer-GIS besteht in der unwirtschaftlichen Verwaltung von objektorientierten Daten wie Katasterdaten, da die einzelnen Objekte nicht sehr gut kapselbar sind. Die für diese Daten wünschenswerte objektweise geschlossene Datenhaltung („black-box"-Speicherung) und objektweise Speicherung lässt sich nicht realisieren. So kann eine weitere interne Objektstruktur im Prinzip nicht aufgebaut werden (bei Katasterdaten z.B. Verwaltung von Grundstücken mit gekoppelten Gebäudeinformationen).

Netzwerk-GIS

Der Schwerpunkt liegt bei den Netzwerk-GIS (NIS) auf der Verwaltung von linienhaften Informationen bzw. Einzugsbereichen. Sie werden vor allem im Ver- und Entsorgungsbereich, für Netzplanung (Verkehr etc.) und als Leitungskataster eingesetzt. Ziel ist die Verwaltung einer Netzwerktopologie (BILL 1996, S. 296).

Im Vordergrund steht die Verwaltung von Attributen für Liniendaten und ihre Verknüpfungen, also die Verknüpfung mit weiteren Sachdaten wie beispielsweise kaufmännische Daten oder von Netzinformationen mit lokalen Standortdaten. Ein Beispiel ist die Verwaltung eines Kanalnetzwerks mit Kanalzuständen unter Einbindung externer Daten wie Fotos von Kanalbefahrungen usw. Neuere Entwicklungen findet man in der Fahrzeugnavigation: Hier werden Ortsdaten, Streckeninformationen (Zustand, erlaubte Geschwindigkeit etc.) zusammengeführt.

In der Standardversion GRASS 5.0.x sind derzeit keine relevanten Module für Netzanalysen vorhanden. In GRASS 5.1.0 wurde Vektornetzwerk-Funktionalität integriert, verschiedene Algorithmen wie kürzeste Wege, Subnetzwerkbildung, Travelling-Salesman etc. stehen zur Verfügung.

Objektorientiertes GIS

Objektorientierte Geographische Informationssysteme sind eine neuere Entwicklung, sie weisen gegenüber den Layer-GIS ganz andere Eigenschaften durch die völlig verschiedene Datenstruktur auf. Die Datenspeicherung erfolgt in Form von geschlossenen Objekten mit Lage- und Attributinformationen. Ein Objekt könnte beispielsweise ein Gebäudekomplex sein, dessen „Inhalt" mit einem GIS verwaltet werden soll.

Durch die Kapselbarkeit der Objekte ist der Aufbau eines „Black-Box-Modells" möglich. Die Objekte besitzen sogenannte „Eigenschaften" und „Methoden". Die Eigenschaften beschreiben die Datenstruktur, die Methoden dagegen den Datenaustausch von und zu anderen Objekten.

Es gibt verschiedene Möglichkeiten der Objektbildung (BARTELME 1995, S. 45):

- Aggregation: Teilobjekte werden zu einem Komplexobjekt zusammengefasst. Speicherung als „black-box", die nach außen geschlossen wirkt;

- Assoziation: Zusammenfassung unter Beibehaltung des Zugriffs auf Teilobjekte;
- Generalisierung/Spezialisierung: Zuweisung von Teilobjekten zu einer übergeordneten Gruppe

Eigenschaften und Methoden sind „vererbbar“, sie können also zwischen Objekten bei der Definition neuer Objekte übertragen werden. Teilobjekte erben dabei von den Komplexobjekten. Der Zugriff auf einzelne Objekte erfolgt über die Objektnummern oder durch eine Koordinatenabfrage. Der Vorteil dieses Konzepts liegt darin, komplexe Datenstrukturen, die an Objekte gebunden sind, zu bilden. Damit ergeben sich gute Verwaltungsmöglichkeiten v.a. im Katasterbereich. Es kann beispielsweise ausgehend von einem Grundstück der Gebäudebestand und die unbebaute Fläche verwaltet werden, wobei diesen Objekten wiederum weitere Informationen wie eine Gliederung in Wohnungen und Büroflächen mit Eigentümerinformationen usw. zugeordnet werden können. Über Aggregation lassen sich alle Untergruppen beispielsweise im Komplexobjekt „Gebäude“ ablegen, eine Assoziation könnte alle unvermieteten Büros zusammenfassen.
Der wesentliche Nachteil dieses Konzepts besteht darin, dass die in der Physischen Geographie und Landschaftsökologie wichtige Verwaltung von kontinuierlichen, flächenhaften Daten in objektorientierter Form ungünstig ist. In GRASS sind derzeit keine objektorientierten Funktionalitäten vorhanden.

Multimedia-GIS

Bei den Multimedia-GIS handelt es sich meistens um einen Aufsatz (also eine Ergänzung) zu einem layer- oder objektorientierten GIS. Man findet sie heute vor allem als Internet-GIS[1], es können aber auch an das GIS gekoppelte Animationen (Filme) und Tondokumente sein. Insbesondere sind die Darstellung virtueller Landschaften als 3D-Visualisierungen üblich („Rundflug“ im GIS über ein Projektgebiet). Die Abbildung 7 zeigt das CGI/PERL-basierte „GRASSLinks“, das die problemlose Integration von GRASS in Online-Applikationen demonstriert.
Standard-GRASS bietet zwar keine Möglichkeit, Multimediaereignisse direkt in eine Karte zu integrieren, jedoch lassen sich auf einfache Weise Animationen produzieren (Module: *xganim* und *NVIZ*).

3.3 Projektionen in GRASS

Da analoge und digitale Karten eine zweidimensionale Darstellung der dreidimensionalen Geoid-Gestalt der Erde bzw. von Erdausschnitten sind, werden geographische Daten immer in einer Projektion dargestellt und gespeichert. Üblicherweise sind die Daten in den jeweiligen lokalen Landessystemen projiziert, in Deutschland und anderen europäischen Ländern im Gauß-Krüger-System.

[1] z.B. GRASS/UMN MapServer:
`http://grass.itc.it/start.html`

Alternativ ist das UTM-System sehr verbreitet. Auch das Längen-/Breitengradsystem gehört dazu, es ist fast auf jeder Karte abgebildet.

Projektionen werden sowohl auf Karten als auch im Geographischen Informationssystem benötigt. Die im GIS gespeicherten Daten bzw. Karten können bei diesem Ansatz als „elektronische Karten“ aufgefasst werden.

In GRASS wird, bevor Geodaten importiert werden können, immer erst ein Projektgebiet definiert. Zu diesen Angaben gehören auch die Projektionsangaben, die für den Import, die Verwaltung und Darstellung der Daten wichtig sind.

Im Folgenden sollen zwei wesentliche generelle Arten von Kartenabbildungen mit ihren Koordinatensystemen kurz dargestellt werden.

Kartographische Abbildungen arbeiten mit Rotationsellipsoiden zur Darstellung des Erdkörpers unter Verwendung eines geographischen Koordinatensystems (basierend auf Längen- und Breitengradangaben, *Meridians* und *Parallels*). Die Angaben der Gradzahlen erfolgen entweder im Sexagesimalsystem (Grad:Minuten:Sekunden) oder im Dezimalsystem (Grad mit Nachkommastellen).

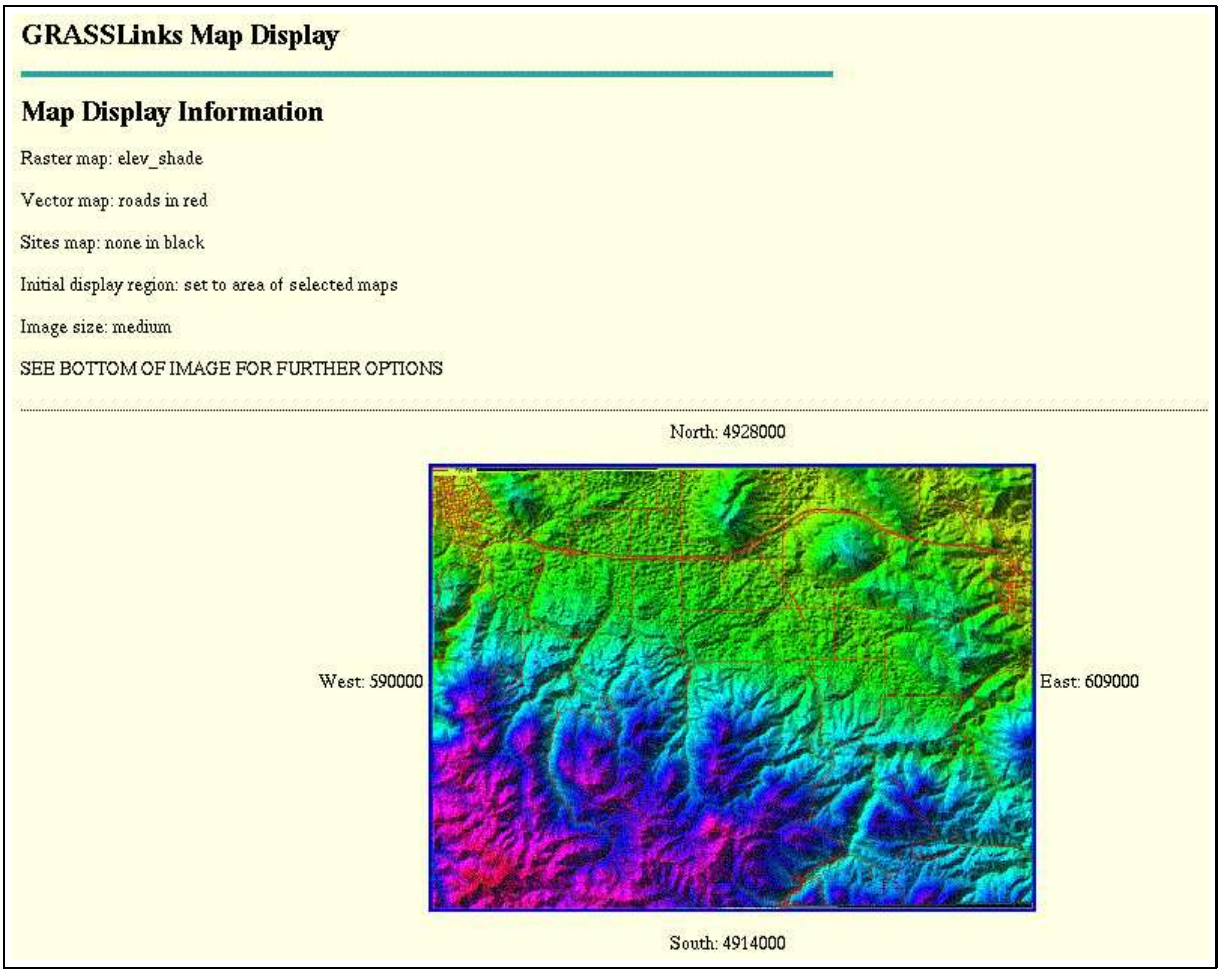


Abbildung 7: GRASSLinks-Beispielanwendung: GRASS als Online-GIS

Im GIS werden bei der Definition eines Projektgebiets die Gebietsgrenzen üblicherweise im Dezimalsystem angegeben. Da Koordinatenangaben auf Karten im sexagesimalen Gradsystem erfolgen, ist eine Umrechnung in das Dezimalsystem durchzuführen. Dabei wird die Gradangabe übernommen, die Minuten einmal durch 60 und die Sekunden zweimal durch 60 (also 3600) geteilt. Dann werden die Gradzahl, die in Grad umgerechneten Minuten und Sekunden addiert.

```
Umrechnungsbeispiel: 12 Grad 45 min 50 sec (12:45:50)

          12 Grad       -> 12.00000 Grad
          45 min/60     ->  0.75000 Grad
          50 sec/3600   ->  0.01389 Grad
                          -------------- Addition der Werte
                           12.76389 Grad (Angabe in Dezimalgrad)
```

Entsprechend der Angabe bei den Gebietsgrenzen ist auch bei der Definition der Rasterdatenauflösung zu verfahren, die im GIS ebenfalls im Dezimalgradsystem erfolgt. Bei GRASS wird in jedem Projektgebiet eine Standardauflösung definiert, jede Rasterkarte kann aber ihre eigene Auflösung bekommen. Für Vektor- und Punktdaten ist die hier gesetzte Auflösung nicht relevant, da sie koordinatenscharf verwaltet werden. Wenn Sie Angaben in Dezimalgrad wieder in Grad, Minuten, Sekunden zurückrechnen wollen, gehen Sie folgendermaßen vor:

```
Umrechnungsbeispiel: 12.7639 Grad (Angabe in Dezimalgrad)

          12.76389 Grad -> 12 Grad,    Rest 0.76389
          0.76389 * 60  -> 45 Minuten, Rest 0.8334
          0.8334 * 60   -> 50 Sekunden
                          --------------
                           12:45:50 Grad (Angabe in Sexagesimalgrad)
```

Ein Hinweis: In GRASS werden bei Angaben in Dezimalgrad West- und Südwerte negativ gezählt, Nord und Ost damit also positiv (z.B. die Stadt Murcia, ES: -1.167, 38.0). bei Angaben in Sexagesimalgrad wird direkt der Quadrant als Buchstabe angehängt bei generell positiver Zählung (Murcia, ES: 1:10:0W, 38:0:0N).

Geodätische Abbildungen wie das Gauß-Krüger-System oder das UTM-System (*Universal Transverse Mercator*) beruhen auf der Forderung nach Winkeltreue (konforme Abbildung, BILL 1996, S. 200). Als Bezugskörper wird auch hier nicht eine „Kugel“, sondern ebenfalls ein Rotationsellipsoid benutzt. Es gibt keine Längen- und Breitenkreise, sondern stattdessen ein ebenes, rechtwinkliges Koordinatensystem. Zur Kartendarstellung wird dieser Zylinder dann in die Ebene abgerollt. Eingesetzt werden geodätische Abbildungen für groß- und mittelmaßstäbige Karten. Das

Ziel besteht darin, durch Abbildung einzelner Regionen die Verzerrungen möglichst gering zu halten. Viele regionale Abbildungen bauen dann beispielsweise die Kartenüberdeckung eines Landes auf.

Das Gauß-Krüger-System entsteht durch „Überstülpung“ eines senkrecht auf der Erdachse stehenden Zylinders über den Erdkörper (hier: Rotationsellipsoid nach Bessel). Damit kommt es zu einer Berührung zwischen diesem Ellipsoid und dem Zylinder genau in einem Meridian (auch als Berührmeridian, Mittelmeridian oder Hauptmeridian bezeichnet). Für die Kartendarstellung werden nun zwei schmale Streifen westlich und östlich dieses Berührmeridians auf den Zylinder projiziert. Sie haben eine Erstreckung von je 2° auf der Erdoberfläche mit einer Verzerrung von maximal 12cm auf 1km am Streifenrand (BILL 1996, S. 204). Diese Projektion wird auch als Transversale Mercatorprojektion bezeichnet. Die projizierten Streifen besitzen keine parallelen Ränder, Kartenränder sind damit also nicht parallel zu Blatträndern bei topographischen Karten. Darauf muss bei einer Georeferenzierung dieser Karten im GIS geachtet werden.

Das UTM-System ist mit dem Gauß-Krüger-System vergleichbar, allerdings ist der „übergestülpte“ Zylinder etwas kleiner (Faktor 0.9996). Als Rotationsellipsoid dient das Internationale Ellipsoid nach Hayford bzw. WGS84. Damit handelt es sich nicht um einen Berührzylinder, sondern um einen Schnittzylinder mit zwei längentreuen Meridianen.

Werden die Zylinder um einen bestimmten Winkel gedreht (Gauß-Krüger-System: 3°, UTM-System: 6°), ergeben sich sogenannte Zonensysteme mit geringfügiger Überlappung. Als Bezug für jede Zone (auch als Streifen bezeichnet) gilt der jeweilige Berührmeridian. In Deutschland sind die Streifen des Gauß-Krüger-Systems rund 100km breit, sie überlappen sich um rund 23km (BILL 1996, S. 202).

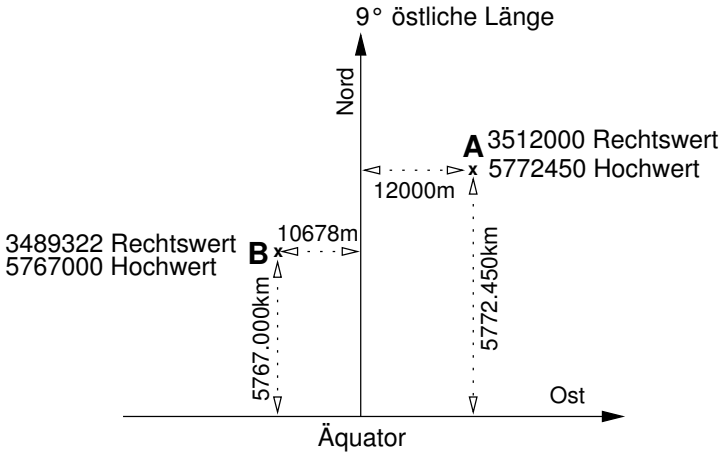


Abbildung 8: Das Gauß-Krüger-Koordinatensystem mit zwei Beispielpunkten A und B

Das Koordinatensystem des für Deutschland, Österreich und andere Länder relevante Gauß-Krüger-Systems ist folgendermaßen aufgebaut (vgl. Abb. 8): Der Mittelmeridian bildet die x-Achse, der Äquator die y-Achse. Wegen der transversalen Projektion ist allerdings das gesamte Koordinatensystem gegenüber einer mathematischen Betrachtungsweise gedreht.

- Die x-Achse erstreckt sich in Nord-Süd-Richtung und gibt den Hochwert an (*Northing*, nach Norden positiv gezählt, Angabe in Meter vom Äquator aus).
- Die y-Achse erstreckt sich in Ost-West-Richtung und gibt den Rechtswert an (*Easting*).

Beim Rechtswert besteht eine Besonderheit: Die erste Ziffer wird aus dem Längengrad geteilt durch Drei berechnet (Mittelmeridian/3). Um negative Werte zu vermeiden, addiert man den Wert von *n*500000m dazu (sogenanntes *False Easting*). Für *n* wird die aus dem Bezugsmeridian berechnete erste Ziffer des Rechtswerts eingesetzt (z.B. 3500000).
Um die Lage eines Punktes zu erhalten, wird die Distanz zwischen Bezugsmeridian und Punkt gemessen.
Vertikal ergibt sich die Distanz zum Äquator, horizontal die Distanz zum jeweiligen Bezugsmeridian. Die Distanzen ergeben die Koordinaten eines Punktes.

Gauß-Krüger - Beispiel A: *Rechtswert: 3512000m, Hochwert 5772450m*
Bedeutung: Mittelmeridian: 9° östlicher Länge (Rechtswert mit führender 3, 9=3∗3), damit liegt der Punkt *12000m östlich* vom Mittelmeridian (zweite Ziffer: 5, 512000m - 500000m=12000m), sowie *5772450m nördlich* vom Äquator.

Gauß-Krüger - Beispiel B: *Rechtswert: 3489322m, Hochwert 5767000m*
Bedeutung: Mittelmeridian: 9° östlicher Länge (Rechtswert mit führender 3, 9=3∗3), damit liegt der Punkt *10678m westlich* vom Mittelmeridian (zweite Ziffer: 4, 500000m - 489322m=10678m), sowie *5767000m nördlich* vom Äquator.

In manchen Projektionen wie dem UTM-System wird noch ein „Scale Factor“ relevant (Zylinder schneidet dann Ellipsoid), beim Gauß-Krüger-System beträgt er 1.0.
Den DDR-Karten liegt ebenfalls die Transversale Mercatorprojektion zugrunde, statt des Bessel-Ellipsoids wurde jedoch das Krassovskij-Ellipsoid benutzt. In GRASS kann auch diese Projektion definiert werden.

Zur Umrechnung von Koordinaten zwischen verschiedenen Projektionen ist ab GRASS 5.0.x das Modul `$ m.proj` hervorragend geeignet (vgl. Beispielumrechnung Anhang A.5). Es leistet Umrechnungen zwischen 121 Projektionen, die definiert werden können. Neben der manuellen Umrechnung einzelner Punkte kann auch per Datei ein Punktdatensatz automatisch umgerechnet werden.

3.4 Berechnungen und Analysen geographischer Daten

Der Schwerpunkt der GIS-Arbeit liegt auf der Datenauswertung. Darunter fallen verschiedene Möglichkeiten:

- Abfrage von räumlichen Informationen
- Aggregierung gespeicherter Informationen zu neuen Daten
- Verschneidung von Datensätzen zur Ermittlung gemeinsamer Flächen und Linien
- Analyse von räumlichen Verteilungsmustern
- Simulationsrechnung für prozesshafte Zusammenhänge

Generell unterscheidet man die GIS-internen Berechnungen und Analysen von den Modellanwendungen. Die GIS-internen Berechnungen und Analysen sind allgemeiner gefasst, Modelle sind dagegen auf bestimmte Probleme spezialisiert. Derartige Modelle können entweder im GIS direkt integriert sein (Vorteil: Sie können GIS-Routinen direkt nutzen, GRASS hat einige direkt gekoppelte Modelle), oder sie sind als externe Modelle (Programme) über Datenaustausch-Schnittstellen angebunden. Die Integrationsfähigkeit von Modellen hängt vor allem von der Offenheit des GIS ab. Da GRASS ein „Open-Source"-Programmpaket ist, sind hier durch die vollständige Offenlegung des Quellcodes samt einer Dokumentation beste Voraussetzungen gegeben. Eine sehr gute Übersicht zur GIS-Funktionalitäten mit praktischen Beispielen (GRASS, ARC/INFO und Idrisi im Vergleich) bieten WADSWORTH UND TREWEEK (1999).

GIS-interne Berechnungen und Analysen

Eine grundsätzliche Methode ist im GIS die raumbezogene Datenabfrage, sie kann intern oder in der gekoppelten externen Datenbank erfolgen. Ein GIS ermöglicht die Analyse von Nachbarschaftsbeziehungen zwischen einzelnen Objekten, in reinen Datenbanksystemen ist sie dagegen nicht adäquat durchführbar.
Ein Schwerpunkt der GIS-Arbeit liegt auf der Erzeugung neuer Daten aus gespeicherten Informationen. Das kann beispielsweise über algebraische Funktionen oder logische Abfragen erfolgen. Ausweisungen von Flächen mit bestimmten Eigenschaften oder die Berechnung fehlender Werte in einer Oberfläche gleichmäßig verteilter Daten können hier ein Ziel sein.
Die Konvertierungsmöglichkeiten zwischen den im GIS vorhandenen Datenstrukturen (Raster-, Vektor- und Punktdatenformat) erlauben die Verwendung von Daten auf vielfältige Weise. Da jede Datenstruktur unterschiedliche Analysemöglichkeiten bietet, kommt den Konvertierungsmodulen eine große Bedeutung zu.
Geostatistische Verfahren ermöglichen die Untersuchung der Charakteristika räumlich verteilter Daten, also der Analyse von räumlichen Verteilungsmustern. Vornehmlich wird die Geostatistik zur Bewertung oder auch zur Berechnung fehlender Werte eingesetzt, die mit hoher Wahrscheinlichkeit

aus diesen räumlichen Verteilungsmustern interpoliert werden können. So lassen sich aus stichprobenhaft im Gelände ermittelten Daten flächenhafte Verteilungen der untersuchten Phänomene abschätzen.

Anbindung bzw. Integration von Simulationsmodellen

Ein Simulationsmodell im GIS führt einen prozessorientierten Verfahrensablauf oder eine Prognose eines oder mehrerer bestimmter Zusammenhänge aus. Das kann beispielsweise die Simulation eines Abflussgeschehens sein oder die Entwicklung von Kaufkraftströmen in einer Stadt-Umland-Beziehung sein.

Die vorhandenen Modelle sind selbstverständlich immer eine Vereinfachung der Realität. In der Anwendung werden sie oft als „Black-Box-Modelle“ eingesetzt. Doch ist immer darauf zu achten, dass die Eingabebedingungen erfüllt sind, damit auch plausible Ergebnisse produziert werden.

Die meisten Modelle beinhalten die Zeitkomponente, deren direkte Erfassung im GIS allerdings derzeit noch unterentwickelt ist. In GRASS 5.0.x sind erste Ansätze einer „Zeitachsen“-Umgebung integriert, die sogenannte „*DateTime-Library*“. Sie ermöglicht in ihrer aktuellen Fassung, Raster- und Punktdaten einen Zeitstempel zu geben. Vorhandene Routinen der GRASS-Programmierbibliothek können in eigenen GRASS-Programmen genutzt werden, um absolute Zeitpunkte (der Datenerfassung beispielsweise) sowie Zeitdifferenzen abzufragen. Dadurch wird die Programmierung von Simulationsrechnungen, die den Zeitfaktor (4D-GIS) benötigen, auf einfachere Weise möglich. GRASS bietet großes Potential für eigene Programmierungen, da es nicht nur diese Routinen vorhält, sondern gegenüber proprietären Systemen auch den vollen Zugriff auf die gesamte GIS-Bibliothek (sogar im Quellcode) ermöglicht.

GRASS verfügt über zwei Modelltypen, die die beiden Kopplungsmöglichkeiten zwischen GIS und Modell zeigen:

- Modelle, die indirekt über eine Schnittstelle zum Datenaustausch angekoppelt sind (z.B. Erosionsmodell *r.agnps50.**)
- Modelle, die direkt in GRASS integriert sind und die GRASS-GIS-Routinen direkt nutzen (z.B. Feuersimulation *r.ros*, Erosionsmodelle *r.answers* und *r.kineros*, Abflussmodelle *r.hydro.CASC2D, r.water.fea*).

Der zweite Modelltyp bietet den Vorteil, dass auf diese Art direkt in der GIS-Datenbank gearbeitet und GIS-Routinen eingesetzt werden können. Da GRASS ein modulares GIS mit offenem Quellcode ist, lassen sich vergleichbare eigene Entwicklungen unter Voraussetzung von C-Programmierkenntnissen gut durchführen.

3.5 Unterstützte GIS-Datenformate

GRASS unterstützt eine Vielzahl von Datenformaten. Nachstehend sind sie ihrer thematischen Zugehörigkeit entsprechend aufgelistet:

Rasterbereich Import:
ASCII, ARC/INFO ascii grid, ERDAS LAN, HDF, GIF (8 bit), TIFF (8/24 bit), SUN Raster (8 bit), PPM (24 bit), TGA (24 bit)

Rasterbereich Export:
ASCII, BIL, ARC/INFO ascii grid, ARCTIFF, ERDAS LAN, HDF, MPEG, Povray, PPM (24 bit), TIFF (8/24 bit), TGA (24bit)

Vektorbereich Import:
ASCII, ARC/INFO ungenerate, ARC/INFO E00, ArcView SHAPE, DLG (U.S.), DXF, DXF3D, GPS-ASCII, USGS-DEM, IDRISI, MOSS, TIGER, VRML

Vektorbereich Export:
ASCII, ARC/INFO ungenerate, ATLAS, DLG, DXF, IDRISI, MAPINFO, MOSS, SDTS, TIGER, XFIG

Bildverarbeitung Import:
BIL/BSQ, CEOS, ERDAS LAN, HDF, LANDSAT TM/MSS, NHAP aerial photos, SAR, SPOT (vgl. auch Raster)

Bildverarbeitung Export:
BIL, ERDAS LAN, HDF (s. auch Raster)

Punktdatenbereich Import / Export:
ASCII, U.S. Census PL94-171 dBase3 CD-ROM, U.S. Census STF1A dBase3 CD-ROM, direktes Schreiben XYZ

Im Anhang werden die Vektor-Datenkonvertierung vom proprietären GIS ARC/INFO nach GRASS erläutert, viele Module kurz beschrieben und spezielle Hinweise gegeben. Die Abbildung 3.5 zeigt den Datenfluss in GRASS.

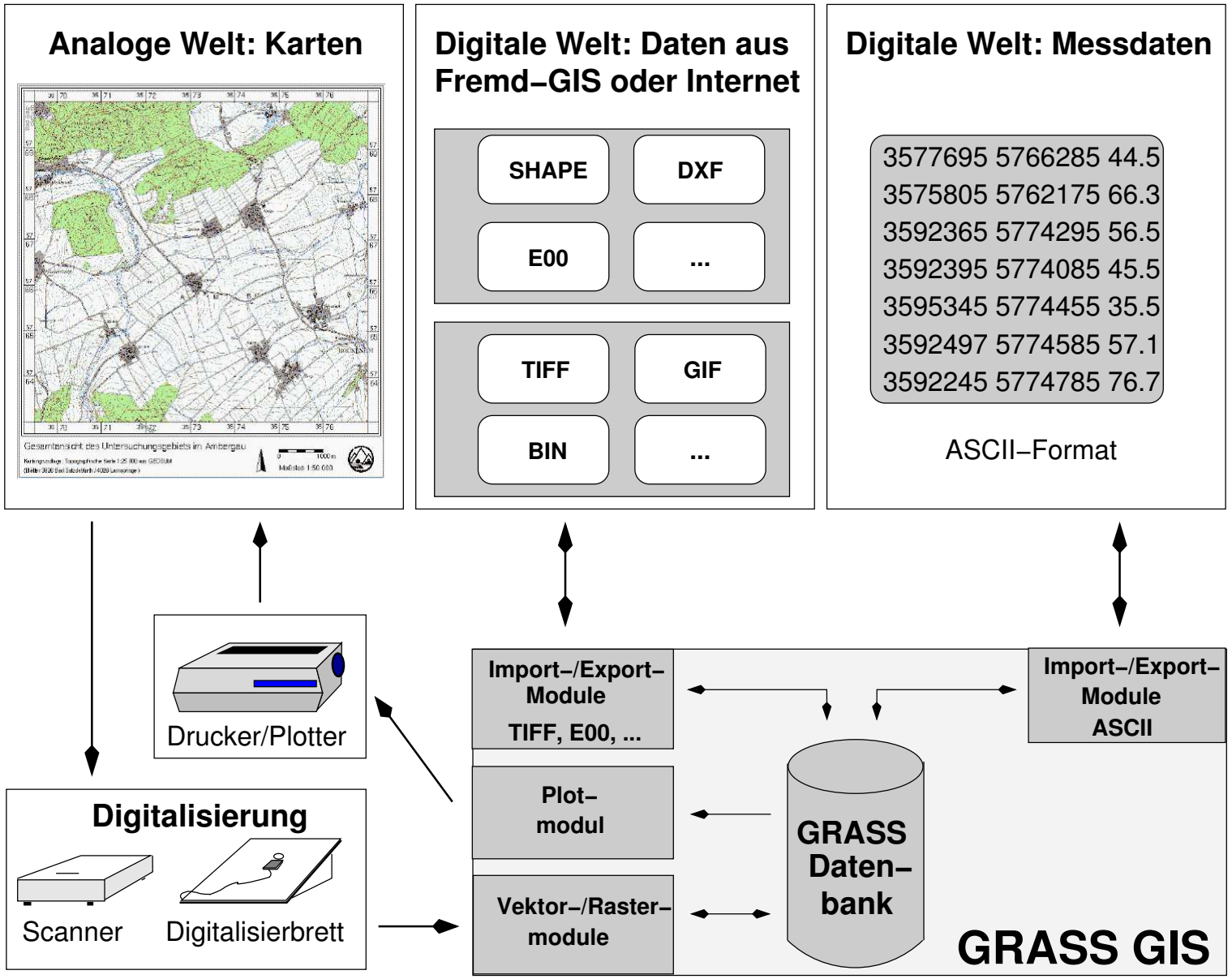


Abbildung 9: Allgemeiner Datenaustausch bei GRASS

Vorschläge zum Weiterlesen über GIS und Projektionen

Bartelme, N. (1995): Geoinformatik. Modelle, Strukturen, Funktionen. Heidelberg

Bill, R. (1996): Grundlagen der Geo-Informations-Systeme. Analysen, Anwendungen und neue Entwicklungen. Band 2, Heidelberg

Burrough, P.A., R.A. McDonnell (1998): Principles of Geographical Information Systems. Spatial Information Systems and Geostatistics. Oxford

GRASS Development Team (2001): GRASS 5.0 Programmer's Manual. Trento, Italien. Im Internet zu finden unter: `http://grass.itc.it/grassdevel.html`

Clarke, K.C. (1997): Getting started with Geographic Information Systems. New Jersey

Evenden, J. (1995): Projections in UNIX

Johnston, C.A., (1998): Geographic Information Systems in Ecology. Methods in Ecology. Oxford

Saurer, H., F.-J. Behr (1997): Geographische Informationssysteme. Eine Einführung. Darmstadt

Snyder, J.P., Ph.M. Voxland (1989): An album of map projections. U.S. Geological Survey. Professional paper 1453, Denver

Wilhelmy, H., A. Hüttermann, P. Schröder (1990): Kartographie in Stichworten. 5. Aufl., Unterägeri

Wadsworth, R., J. Treweek (1999): Geographical Information Systems for Ecology. An Introduction. Essex

4 Der erste Einstieg in GRASS

Prinzipiell ist GRASS ein ganz normales Anwendungsprogramm. Es weist seit der Version 4.2.1 eine graphische Benutzeroberfläche auf, die seine Bedienung mit der Maus ermöglicht. Daneben können, wie aus vorherigen GRASS-Versionen gewohnt, die GIS-Kommandos auch in dem GRASS-Terminalfenster eingegeben werden. Die Programmstruktur ist allerdings etwas anders als bei üblichen Anwendungsprogrammen: Die GRASS-Kommandos (typischerweise als *GRASS-Module* bezeichnet) sind den übrigen UNIX-Kommandos nach dem Start von GRASS gleichgestellt. Das bedeutet, dass im selben Fenster, in dem GRASS gestartet wurde, auch alle UNIX-Befehle zur Verfügung stehen. Der Sinn liegt darin, so auf die gesamte UNIX-Kapazität bei der Arbeit mit GRASS zugreifen und leistungsfähige Programmabläufe erstellen zu können. GRASS-„Neulinge“ müssen sich vielleicht zunächst an diese Struktur gewöhnen, sie werden aber schnell die Vorteile des Konzepts erkennen.

Die Benutzeroberfläche „TclTkGRASS“, die mit der Maus bedient wird, ist eine Erweiterung ohne eigene GIS-Funktionalität. Vielmehr steuert „TclTkGRASS“ einen Teil der vorhandenen GRASS-Module und soll der Arbeitserleichterung dienen (vgl. Abbildung 10).

Die GIS-Daten werden bei GRASS in einer Verzeichnisstruktur gespeichert. Bevor mit der Arbeit in GRASS begonnen werden kann, ist dafür ein „Standardunterverzeichnis“ (als *GRASS-Database* bezeichnet) nutzerseitig zu erzeugen und später in GRASS anzugeben. Darin organisiert GRASS dann selbsttätig seine Daten in Form von weiteren Unterverzeichnissen. Jedes Projekt bekommt bei der Definition des Projektgebiets automatisch einen Unterverzeichnisbaum innerhalb dieses Standardverzeichnisses. Die Datenorganisation sollte GRASS überlassen werden. Auch alle Dateioperationen wie das Umbenennen oder Kopieren von Karten müssen mit GRASS-Befehlen erfolgen, da intern immer mehrere Dateien betroffen sind. Manuelle Änderungen sind nur in Ausnahmefällen sinnvoll.

Die Grafikausgabe, also das Kartendarstellungsfenster, ist in GRASS kein „normales“ Fenster, sondern stellt geographische Daten mit Koordinaten dar. Es sollte, anders als üblich, mit Vorsicht behandelt werden. Diese Grafikausgabe-Fenster (sogenannte *GRASS-Monitore*) dürfen keinesfalls zur Beendigung der Arbeit per Mausklick geschlossen werden, sondern ausschließlich mit dem GRASS-Befehl `d.mon`. Die Benutzeroberfläche ermöglicht ebenfalls die Konfiguration und Bedienung dieser Fenster.

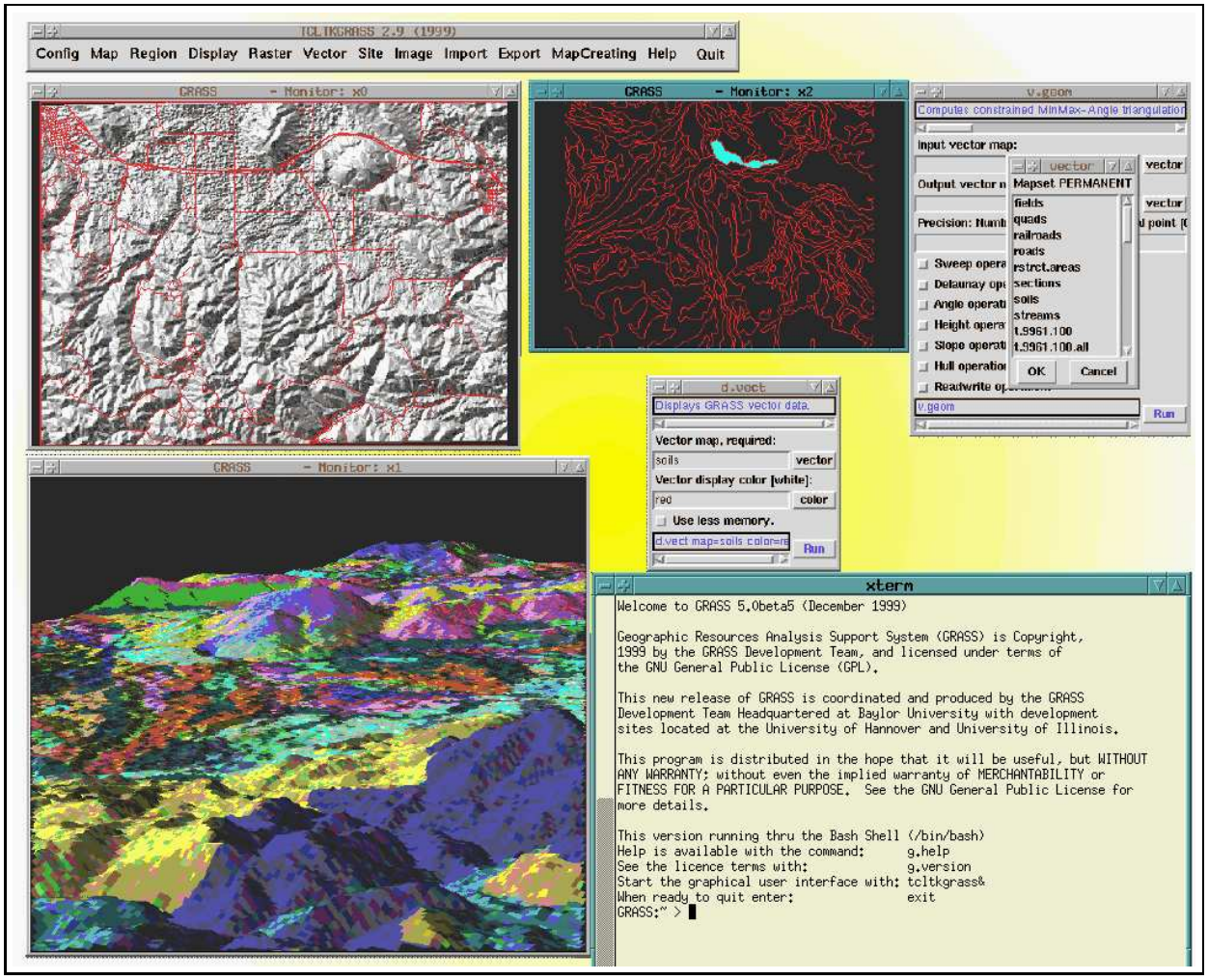


Abbildung 10: Benutzung von GRASS mit der graphischen Benutzeroberfläche „TclTkGRASS“

Die Module und damit die GIS-Funktionalität sind in GRASS bereits durch ihre Namen sehr klar gegliedert. Es gibt verschiedene Funktionsklassen: Module zur Datenvisualisierung, zur Vektor-, Raster- und Punktdatenverarbeitung, allgemeine Dateiverarbeitungs-, Kartenerstellungsbefehle usw. Dabei gibt der erste Buchstabe die Funktionsklasse an, anschließend folgt ein Punkt und nun ein oder zwei weitere Worte, letztere durch einen weiteren Punkt getrennt. Diese Worte sind der englischen Sprache entnommen und leicht verständlich. Tabelle 4.1 listet die wesentlichen Funktionsklassen auf.

Einige Beispiele für GRASS-Befehle: `v.in.shape` ist ein Vektorbefehl zum Import einer ESRI/-SHAPE-Datei, `r.buffer` berechnet eine Pufferzone um Rasterlinien und -flächen, `d.measure` erlaubt Strecken- und Flächenmessungen per Maus in der Grafikausgabe (im GRASS-Monitor), `i.ortho.photo` erzeugt ein Orthofoto aus einem gescannten Luftbild. Eine Kurzübersicht zu rund 250 GRASS-Befehlen finden Sie in Abschnitt A.2. Fast alle Module können sowohl kommandozeilenorientiert (Modulaufruf und Parameterangabe in einer Zeile) benutzt werden als auch interaktiv. Sie können jedes Modul ohne Parameter aufrufen und werden durch Menüs abgefragt.

Präfix	Funktionsklasse	Bedeutung der Befehle
d.∗	*display*	für Grafikausgabe und visuelle Abfragen
g.∗	*general*	allgemeine Dateioperationsbefehle
i.∗	*imagery*	für Bildverarbeitung
m.∗	*misc*	verschiedene Befehle
p.∗	*paint*	Kartenerstellungsbefehle
ps.∗	*postscript*	Kartenerstellungsbefehle für Postscriptformat
r.∗	*raster*	für Rasterdatenverarbeitung
s.∗	*sites*	für Punktdatenverarbeitung
v.∗	*vector*	für Vektordatenverarbeitung

Tabelle 4.1: Struktur der GRASS-Modulnamen

Nun noch einige Worte zur weiteren GRASS-Terminologie: Ein Projektgebiet wird in GRASS als *location* bezeichnet. Sie wird über ihre geographischen Ränder mit Koordinatenangaben und zusätzliche Projektionsangaben definiert. Innerhalb dieser *location* können Arbeitsgebiete, sogenannte *mapsets*, festgelegt werden. Häufig erzeugt man aber nur eine *mapset*, die so groß wie die *location* ist. Mehrere *mapsets* bieten sich bei Teamarbeit an. Bei der Definition der *location* sollte man bedenken, dass sie sich nachträglich nicht erweitern lässt (bzw. mit Fachwissen nur durch Modifikation in der Datenbank). Sie sollte also lieber zu groß gewählt werden, um später Probleme zu vermeiden. Gebiete, in denen keine Daten vorliegen, sehen einfach „schwarz“ aus und beinhalten keine Daten. Die Datenbank in GRASS wird als *database* bezeichnet.

Die Planung der Datenbank benötigt generell in einem Geographischen Informationssystem ein wenig Vorbereitungszeit. Da mit der gewählten Struktur die GIS-Arbeit steht und fällt, sollte hier sehr sorgfältig vorgegangen werden. Besondere Aufmerksamkeit kommt der Auflösung zu - eine sehr gute Auflösung erfordert große Rechen- und Speicherkapazitäten, eine geringe Auflösung führt dagegen selten zu den gewünschten Ergebnissen. Das Optimum liegt also dazwischen und hängt von den Anforderungen, aber auch sehr stark von den zur Verfügung stehenden Daten ab.
Bevor Sie nun mit GRASS erstmals arbeiten, legen Sie ein Unterverzeichnis (die oben angesprochene *GRASS-Database*, in der GRASS seine Daten speichert) in Ihrem home-Verzeichnis an (vgl. auch Abb. 3). Ein Beispiel:

```
$ cd $HOME
$ mkdir grassdata
```

Anschließend kann GRASS gestartet werden (Aufrufname versionsabhängig):

`$ grass4.3` oder `$ grass5`

Der prinzipielle Ablauf sieht folgendermaßen aus: Als Erstes erscheint der „Begrüßungsbildschirm“ von GRASS. Es werden einige Angaben verlangt, die die zu bearbeitende Region und den Pfad zur

verwendeten *GRASS-Database* betreffen. Der Name für die *location* (also das gesamte Projektgebiet), die *mapset* und der Pfad zur *database* sind anzugeben (z.B. /home/emil/grassdata).
Wenn Sie „`list`“ in einer Formularzeile und anschließend „ESC“-„RETURN“ zum Verlassen der Eingabemaske eingeben, listet GRASS alle in dieser Rubrik vorhandenen Daten auf. Sind *location*, *mapset* und *GRASS-Database* angegeben, geht es mit „ESC“-„RETURN“ weiter.
Die *location* braucht nun geographische Informationsangaben wie das zu verwendende Koordinatensystem samt Bezugsellipsoid, die Randkoordinaten des Projektgebiets und die Standardauflösung für Rasterdaten. Im Allgemeinen wird in Deutschland und einigen anderen europäischen Ländern mit dem Gauß-Krüger-Koordinatensystem gearbeitet (vgl. Abschnitt 3.3). Deshalb wird es in diesem Handbuch schwerpunktmäßig benutzt. Eine prinzipielle Darstellung, wie *locations* definiert werden, finden Sie in Abbildung 11. Im Unterschied zu anderer GIS-Software ist bei GRASS zuerst das Projektgebiet zu definieren, bevor man die Benutzeroberfläche sieht. Effektiv ergibt sich aber kein Unterschied, in ESRI-Software muss beispielsweise das Projektgebiet nach dem Aufruf des Programms auch sofort definiert werden. Es ist also nur die Reihenfolge anders.

Nun soll ein Beispiel folgen, wie eine *location* eingerichtet wird. Nach dem Start von GRASS erscheint der besagte „Begrüßungsbildschirm“. Hier werden nun Namen für *location* und *mapset* gewählt sowie der Pfad auf die GRASS-Datenbank angegeben:

```
                                GRASS 5.0
LOCATION:
   This is the name of an available geographic location. -spearfish-
   is the sample data base for which all 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 with the command: g.region
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
LOCATION:   celle_________   (enter list for a list of locations)
MAPSET:     flussaue______   (or mapsets within a location)
DATABASE:   /home/emil/grassdata_____________________________

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

Nach dem Drücken von „ESC“-„RETURN“ geht es mit den Definitionen für dieses Projektgebiet weiter, GRASS fragt die Parameter ab (die Reihenfolge der Abfrage weicht in GRASS 4.x leicht ab).

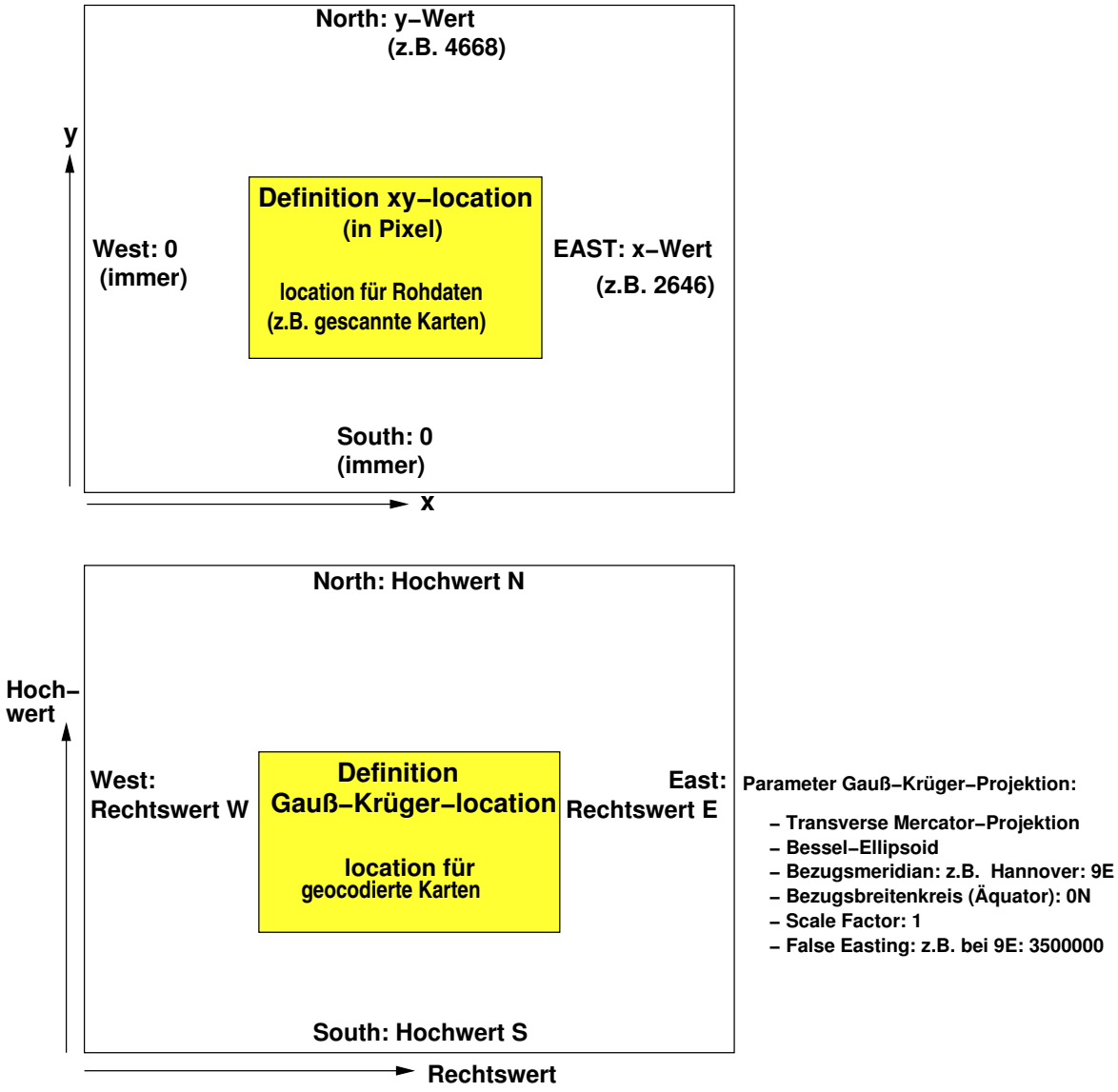


Abbildung 11: Prinzipielle Definition einer xy-*location* bzw. einer Gauß-Krüger-*location*

Als erstes ist aus einem Menü die Projektion auszuwählen; da das Gauß-Krüger-System nicht in der Auswahlliste vorkommt, muss es unter „other“ selbst definiert werden.

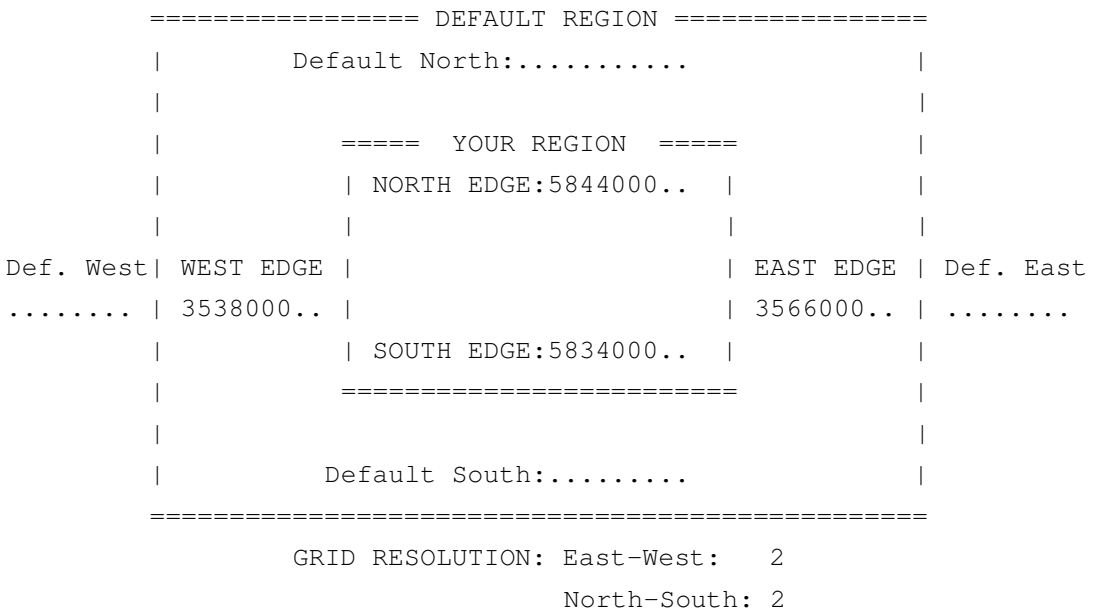
- Für das Gauß-Krüger-Koordinatensystem wird demnach angegeben:
  `coordinate system for location: „other\ (D)`
- Es folgt eine einzeilige Kurzbeschreibung des Projektgebiets:
  `one line description for location`: z.B. Flussgebiet bei Celle
- Dann werden die *location*-Grenzen über die Randkoordinaten festgelegt (Beispiel). An Gauß-Krüger-Werte aus topographischen Karten sind drei Nullen anzuhängen, so wird beispielsweise 5866 zu 5866000. Die letzte Stelle ist also die Meterangabe, sollen auch Zentimeter angegeben werden, folgt ein Punkt mit Nachkommastellen. Wenn das Projektgebiet in ei-

nem Gebiet liegt, bei dem im Gauß-Krüger-System zwei Bezugsmeridiane liegen, also im westlichen Teilgebiet ein anderer Bezugsmeridian gilt als im östlichen Teilgebiet, so wählt man den Meridian, dem die größere Projektfläche zugehörig ist. Dieser bestimmt dann die führende Zahl für die anzugebenden Rechtswerte. Der entstehende Projektionsfehler darf vernachlässigt werden.

Bei Projektgebieten im Längen-/Breitengradsystem können die Gebietsrandkoordinaten entweder im Sexagesimalsystem (Grad) getrennt mit „:“ angegeben werden (z.B. 10:30:00) oder im Dezimalsystem (10.5).

- Es folgen nun weitere Angaben zur Projektion (hier für das Gauß-Krüger-System, „list“ gibt die wählbaren Möglichkeiten an, vgl. auch Abschnitt 3.3):
  - `specify projection name`: *tmerc* (Transverse Mercator)
  - `specify ellipsoid name`: *bessel* (Bessel-Ellipsoid)
  - `Enter Central Parallel`: *0N* (der Äquator ist Bezugsbreitenkreis)
  - `Enter Central Meridian`: *9E* (als Beispiel für Celle, dieser Wert ist den topographischen Karten des Projektgebiets entnehmbar; 9° entspricht einer führenden 3 beim Rechtswert)
  - `Enter Scale Factor at the Central Meridian`: *1*
  - `Enter False Easting`: *3500000* (bedingt durch Gauß-Krüger-System, führende 3 wg. 9E als Central Meridian)
  - `Enter plural form of units`: *meters*

Nun folgt die Beschreibung der Randkoordinaten des Projektgebiets und die Definition der Standard-Rasterauflösung. Ein Beispiel für die Definition einer *location* in Gauß-Krüger-Koordinaten:



Die anzugebende Standard-Rasterdatenauflösung (GRID RESOLUTION) wird den Anforderungen entsprechend gewählt. In obigem Beispiel entspricht die Länge einer Rasterzelle zwei Metern in der Natur. Prinzipiell ist anzuraten, bei der Auflösung in Schritten von 0.25 zu arbeiten (0.25, 0.5, 1.75, 2.00, 12.25 usw.). Für Vektor- und Punktdaten spielt diese Auflösungsangabe keine Rolle, da sie koordinatenscharf gespeichert werden. Jede Rasterkarte kann aber ihre eigene Auflösung haben.
Mit „ESC"-„RETURN" verlässt man diese Eingabe und akzeptiert die nun erscheinende Auflistung der Parameter, wenn alles richtig ist.
Nun gelangt man wieder zur Begrüßungsseite. Mit einem weiteren „ESC"-„RETURN" wird diese Seite endgültig verlassen, ein Name für die *mapset* war ja bereits angegeben. Die Erzeugung der innerhalb der *location* liegenden *mapset* wird mit „yes" durchgeführt. Alle benötigten Werte wie die Projektionsart etc. gelten natürlich auch für die *mapset*.

Nun ist also das Projektgebiet, nämlich die *location* mit einer *mapset* eingerichtet. Sie sind im GRASS-System „angekommen". Sofern installiert, können Sie nun die graphische Benutzeroberfläche „TclTkGRASS" aufrufen. Das geschieht folgendermaßen:

```
$ tcltkgrass &
```

Das kaufmännische Und-Zeichen (&) ermöglicht, dass Sie das Terminalfenster weiterbenutzen können (vgl. Abschnitt 2.2.6). Es handelt sich ja schließlich um das GRASS-Kommandofenster. Alle wesentlichen GRASS-Module sind bei TclTkGRASS per Maus bedienbar. Nun sollte die Grafikausgabe aufgerufen werden, damit Sie später auch Karten betrachten können. Diese Grafikausgabe wird als „GRASS-Monitor" bezeichnet. Es gibt maximal sieben Grafikausgaben, die mit `x0` bis `x6` bezeichnet sind. Sie können diese GRASS-Monitore in TclTkGRASS über „DISPLAY"⇝„MONITORS"⇝„START" aufrufen bzw. mit „STOP" wieder beenden. Alternativ können Sie das entsprechende GRASS-Kommando auch in das Kommandofenster eingeben. So wird beispielsweise der erste Monitor `x0` mit

`$ d.mon start=x0` oder vereinfacht mit

`$ d.mon x0` gestartet.

Wenn Sie nur d.mon gefolgt von „return" eingeben, können Sie das Modul auch menügeführt bedienen. Diese Art der Menüführung ist prinzipiell in (fast) allen GRASS-Modulen integriert.
Die Grafikausgabe wird standardmäßig mit 32768 Farben gestartet. Sie können mit der „nlev"-Option (number of color levels) auch mehr Farben bis zu 16,7 Millionen (24bit) bekommen:

```
$ d.mon start=x0 nlev=256
```

Die Farbtiefe errechnet sich nach ncolors = nlev^3. Bei voller Farbtiefe kann der Start des GRASS-Monitors je nach Rechnerleistung einige Sekunden dauern, da intern entsprechend temporärer Speicherplatz belegt werden muss.

Wie die Grafikmonitore in ihrer Größe über das Setzen von UNIX-Umgebungsvariablen (z.B. für /etc/profile) eingestellt werden, steht in Abschnitt A.1. Wenn Sie TclTkGRASS benutzen, können

Sie in den Menüs „CONFIG“⇝„OPTIONS“⇝„DISPLAY DIMENSIONS“ die Größe angeben. Diese Änderungen sind allerdings erst beim nächsten Öffnen eines Monitors wirksam. Beim Verlassen von TclTkGRASS ist dann „SAVE CONFIG“ mit „YES“ zu beantworten, um diese Konfigurationsinformationen für weitere Sitzungen zu speichern. Allgemein gilt, dass ein Grafikmonitor nicht *mit der Maus* in seiner Größe verändert werden sollte. Dann kann es nämlich zum Verbindungsverlust zwischen GRASS und dem Fenster kommen, so dass der GRASS-Monitor keine Daten mehr ausgibt. Die Größe sollte also nur über die entsprechenden Environment-Variablen oder TclTkGRASS eingestellt werden. Diese Sensibilität liegt darin begründet, dass es sich beim Monitorfenster um ein besonderes Fenster mit interner Koordinatenverwaltung handelt.

Bevor die Arbeit mit GRASS weiter besprochen wird, soll zunächst das Beenden des Gesamtprogramms beschrieben werden. Das ordnungsgemäße Verlassen von GRASS ist sehr wichtig, damit keine Daten verloren gehen. Benutzen Sie TclTkGRASS, so ist dieses zuerst zu beenden. TclTkGRASS fragt Sie, ob Sie die geöffneten GRASS-Monitore schließen wollen. Ist letzlich noch ein GRASS-Monitor geöffnet, wird er mit

```
$ d.mon stop=x0
```

abgeschaltet (weitere Monitore entsprechend) oder einfach mit der Maus das Monitorfenster geschlossen. GRASS selbst wird anschließend mit

```
$ exit
```

verlassen. Geodaten sind bereits direkt nach ihrer Erzeugung automatisch gespeichert worden. Die Reihenfolge beim Beenden sieht also folgendermaßen aus:

1. TclTkGRASS beenden (Quit, Monitore schließen per Menüfenster)
2. Sind noch Monitore offen, mit `$ d.mon` oder der Maus schließen
3. GRASS mit `$ exit` verlassen.

5 Planung und Aufbau einer Datenstruktur

Größte Sorgfalt sollte bei der Einrichtung des Projektgebietes (der *location*) aufgewendet werden. Die Struktur wird von den vorhandenen Daten bestimmt, die sehr häufig in analoger Form, also als gedruckte Karten, vorliegen.

Der folgende Abschnitt zeigt zwei Varianten auf, wie sich Ausdehnung und Auflösung in einer *location* berechnen lassen. Es handelt sich hierbei um Beispiele, die anhand einer gescannten Karte besprochen werden und auf beliebige Rasterdaten (z.B. Karte der Grundwasser-Flurabstände etc.) übertragbar sind. Es besteht aber noch ein *dritter*, auflösungsunabhängiger Weg, der vor allem zum Ausgleich von Scanfehlern sinnvoll ist. In diesem Fall überlässt man GRASS die Umrechnung der GRID RESOLUTION. Dieser Weg wird in Abschnitt 6.6 („Vereinfachte Kartenimportierung“) erläutert.

Doch zunächst geht es um eine Vorgehensweise , die auf genaue Koordinaten- und Auflösungswerte und unverändertes Datenmaterial abzielt.

Generell soll auf den Zusammenhang zwischen geometrischer Kartenstrecke, Maßstab und geographischer Ausdehnung verwiesen werden. Dieser allgemein kartographische Zusammenhang gilt auch bei der Übertragung einer analogen Karte in eine digitale Karte. Da hierzu üblicherweise Scanner eingesetzt werden (zumindest zur Herstellung einer Vorlage für die Vektorisierung im GIS), sind die genannten Begriffe von großer Bedeutung. Selbstverständlich muss das Urheberrecht bei dem Scannen von Karten beachtet werden.

Bei der Nutzung digitaler Daten, die nicht nur (z.B. von amtlicher Stelle) gescannt, sondern bereits auch geocodiert sind (bzw. volldigital hergestellt wurden), vereinfacht sich die Datenstruktur und der Import erheblich. Die Datenstruktur sich ergibt hier direkt aus der Vorgabe der digitalen Karte in geographischen Grenzen und Auflösung.

5.1 Definition des Projektgebiets bei vorgegebener geographischer Auflösung

Die *erste Variante* bei der Einrichtung einer *location* besteht darin, dass die Grenzen des Projektgebiets durch die Aufgabenstellung festgelegt sind und für eine zu scannende Karte die Scaneinstellungen angepasst werden müssen.

Die Anzahl der Rasterzellen in einer *location* ergibt sich hier aus der Länge und Breite des Projektgebiets und der gemäß den Anforderungen an die Ergebnisse gewählten Auflösung (GRID RESOLUTION).
Beispiel: Die Länge einer quadratischen *location* sei 1km, mit einer gewünschten Auflösung von 5m pro Zelle (GRID RESOLUTION) ergibt sich:

$$\frac{1.000m}{5m} = 200\ Zeilen\ (bzw.\ Spalten) \Rightarrow 200 * 200\ Rasterzellen \tag{5.1}$$

Wenn beispielsweise eine Karte für diesen Bereich gescannt wird, muss sie aus $200 * 200$ Zeilen und Spalten bestehen, um verzerrungsfrei importiert werden zu können. Hier bestimmt also die *location* die Scanauflösung, da ja die Länge und Breite des Scanausschnitts durch die Grenzen der *location* festgelegt sind. Vergleichbares gilt für andere Datengrundlagen, deren Auflösungen ebenfalls angepasst werden müssen (z.B. digitales Höhenmodell mit definierter Rasterweite).

Es stellt sich nun also die Frage, welche Auflösung beim Scanner vorzugeben ist, um genau diese Spalten- und Zeilenanzahlen bei festgelegter Seitenlänge zu bekommen. In diesem Beispiel sei ein Maßstab von 1:25000 bei der zu scannenden Karte angenommen. Der *location*-Länge von 1000m entsprechen damit:

$$\frac{100.000cm}{25.000} = 4\,cm, \tag{5.2}$$

also 4cm in der Karte. Die Scanauflösung dieser zu importierenden Karte berechnet sich folgendermaßen:

$$\frac{Rasterzeilen\ (bzw.\ -spalten)}{zu\ scannende\ Kartenstrecke} = \frac{200\ Zeilen\ (bzw.\ Spalten)}{4cm} = 50\frac{Zeilen}{cm} \tag{5.3}$$

Auf dpi (dots per inch, entspricht den Zeilen bzw. Spalten pro inch) umgerechnet bedeutet das:

$$50\frac{Zeilen * 2,54}{inch} = 127dpi \tag{5.4}$$

Dieser Wert muss beim Scanner eingestellt werden, der zu scannende Ausschnitt entsprechend den geographischen Grenzen des gewünschten Ausschnitts ebenfalls. Je nach Kartenmaßstab ergibt sich eine unterschiedliche Auflösungszahl. Sie muss groß genug sein, damit auch kleine Kartenschrift lesbar bleibt. Sehr problematisch (quasi unmöglich) ist beim Scanvorgang, genau die passenden Zeilen und Spalten im gewünschten Ausschnitt zu erhalten und die Karte exakt ausgerichtet zu scannen. So lässt sich bei dieser Variante die Nachbearbeitung mit Bildverarbeitungssoftware nicht umgehen (z.B. mit den netpbm-tools[1]).

Nun folgt eine alternative Methode zur Definition des Projektgebiets in Abhängigkeit von den zu verarbeitenden Daten.

[1] Die netpbm-tools finden Sie im Internet u.a. hier:
`ftp://ftp.uni-stuttgart.de/pub/scivi/imageprocessing/pbm/` als `netpbm-1march1994.tar.gz`

5.2 Definition des Projektgebiets bei flexibel wählbarer Auflösung

Die zweite Variante zur *location*-Einrichtung kann angewendet werden, wenn die Auflösung in der *location* entsprechend einer gescannten Karte (oder anderen vorgegebenen Datengrundlagen) wählbar ist. Hier ergeben sich die Parameter der *location* also aus der Datengrundlage. Das einzulesende Bild und die *location* müssen die gleiche Anzahl von Rasterzeilen und -spalten aufweisen. Mit dem Programm `$ xv` lassen sich diese Angaben des Rasterbildes beispielsweise ermitteln (Menüpunkt „Windows“ - „Image Info“).[2]
Welcher Strecke in der Natur die Länge einer Rasterzelle entspricht, ergibt sich aus der vorher gewählten Auflösung am Scanner. Der Wert wird dann bei der Einrichtung der *location* als GRID RESOLUTION angegeben (vgl. Koordinatenformular auf Seite 44). Das Rasterbild darf aber nicht in der Größe skaliert werden, da sich sonst die Maßstäbe im Bild ändern.
Sinnvollerweise wird beim Scannen mit einer Auflösung im Bereich von 150–300dpi gearbeitet (als Farbfoto mit 256 Farben), die Kartensignaturen sollten lesbar sein. Die Rasterauflösung in GRASS (GRID RESOLUTION der *location*) ergibt sich daraus. Es folgt ein Berechnungsbeispiel für eine quadratische *location*. Als Scanauflösung seien 300dpi angenommen:

$$300dpi = 300\frac{Zeilen}{2,54cm} = 118,11\frac{Zeilen}{cm} \tag{5.5}$$

Der Maßstab der gescannten Karte sei 1:25.000. Ein Zentimeter auf der Karte entspricht also 25.000cm in der Natur. Nun soll berechnet werden, welcher Strecke in der Natur die Länge einer Rasterzelle entspricht:

$$\frac{Strecke\ in\ der\ Natur}{Scanzeilen\ pro\ cm} = \frac{25.000cm}{118,11\ Zeilen} = 211,6\frac{cm}{Zeile} = 2,12\frac{m}{Zeile} \tag{5.6}$$

Dieser Wert von 2.12m wird bei der Einrichtung der *location* als GRID RESOLUTION eingetragen (mit einem Punkt als Komma). Wenn die GRID RESOLUTION eine gerade Zahl sein soll, muss entsprechend rückgerechnet und die Scanauflösung abweichend von 300dpi variiert werden. Normalerweise sollte diese beim Scanner bis zu einem Maximum frei einstellbar sein.

5.3 Universelle Definition des Projektgebiets bei Verwendung gescannter Karten

Die vorherigen Methoden haben den entscheidenen Nachteil, dass kein Ausgleich des Lagefehlers gescannter Karten stattfindet. Beim Scannen ist es quasi unmöglich, eine Karte exakt aufzulegen. Die gescannte Karte wird daher immer eine leichte Drehung gegenüber der Nordausrichtung aufweisen, der für die Arbeit im GIS nicht akzeptabel ist. Daher wird nun ein Weg vorgestellt, der zwar zeitlich betrachtet etwas aufwendiger ist, jedoch einen exakten Kartenimport durch Lagekorrektur anhand von zu setzenden Passpunkten ermöglicht.

[2] Das Programm ist im Internet als Shareware verfügbar: `http://www.trilon.com/xv/`

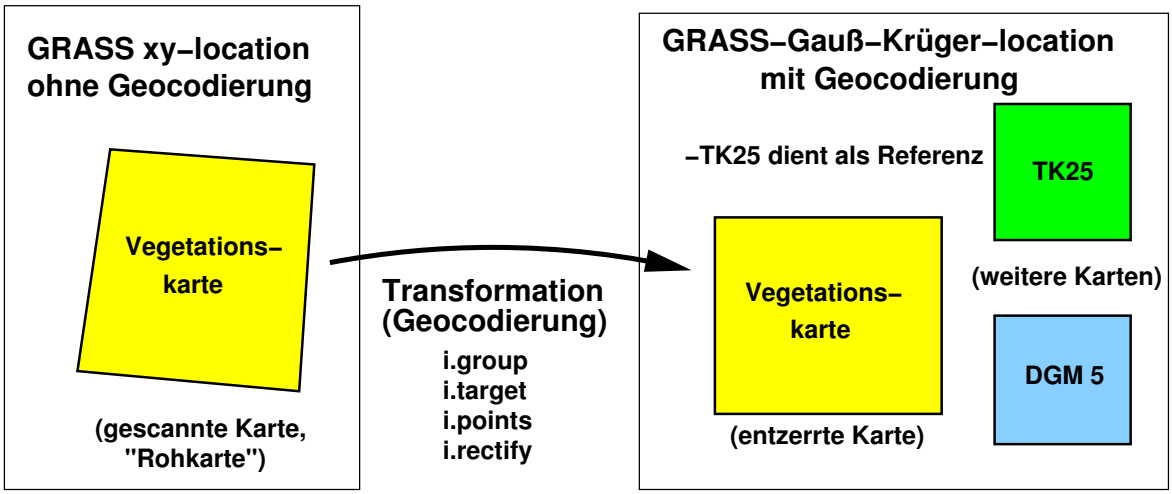


Abbildung 12: Vereinfachter Kartenimport gescannter Karten über Geocodierung mit Passpunkten

Dieser Weg kann als universeller Weg, gescannte Karten zu importieren, bezeichnet werden. Hier sind allerdings **zwei Projektgebiete** mit unterschiedlichen Projektionen zu definieren. Dabei wird die gescannte Karte, zunächst in eine einfache xy-*location* (ohne Projektion) importiert und dann in eine zweite *location* mit Gauß-Krüger-Koordinaten „entzerrt“ (geocodiert). Den prinzipiellen Ablauf zeigt Abbildung 12. Die gescannte Rohkarte wird in der xy-*location* mit Passpunkten versehen, die aus der Kartenvorlage abgelesen und graphisch den Referenzpunkten in der gescannten Karte zugewiesen werden. Anhand einer nachfolgenden Transformation erfolgt die Geocodierung der gescannte Karte aus der xy-*location* in der Gauß-Krüger-*location* für die weitere GIS-Arbeit.

Ein Beispiel: Sie haben eine Karte mit einem Scanner beispielsweise in 300dpi gescannt. Diese Karte wird als Rohkarte bezeichnet. Die Rechenschritte, wie der Scanner bei entsprechendem Maßstab sinnvoll eingestellt werden muss, finden Sie im vorherigen Abschnitt 5.1. Dabei ist es äußerst sinnvoll, eine Karte mit sogenannten „Gauß-Kreuzen“ zu verwenden, also eine Karte, in der kleine Kreuzchen an den Schnittpunkten der Gauß-Krüger-Koordinatenlinien eingetragen sind. Diese Kreuze stellen die für die „Entzerrung“ der Karte maßgeblichen Koordinatenpunkte dar. Beim Scannen der Karte sollten die Kreuze, die am Rand des interessierenden Ausschnitts liegen, gut erkennbar sein, d.h. die Karte muss einige Prozent größer gescannt werden, als sie später in GRASS benötigt wird. *Hinweis*: Der Blattrand stimmt nicht mit dem Gauß-Krüger-System überein, sie sind nicht parallel. Das bedeutet, die Ränder können nicht als *location*-Ränder dienen.

Vor der „Entzerrung“ der Rohkarte wird zunächst einmal die später benötigte Gauß-Krüger-*location* angelegt, in der die Karte letztlich vorliegen soll. Sie wird gemäß der gewünschten geographischen Grenzen und der gewünschten Auflösung angelegt (vgl. Abschnitt 4). Prinzipiell kann natürlich auch jede andere, von GRASS unterstützte Projektion definiert werden. Zu beachten ist jedoch,

dass man die Auflösung der Ziel-*location* nicht zu niedrig wählt. Sofern die Auflösung zu gering ist, bekommt man je nach Maßstab eventuell eine recht schlecht lesbare digitale Karte. Es lohnt sich also, vorher eine Umrechnung zwischen Scanauflösung und geographischer Auflösung durchzuführen.
Später wird bei der „Entzerrung“ der zunächst in eine xy-*location* importierten Karte durch ein passendes GRASS-Modul die gescannte Karte in diese gerade angelegte Gauß-Krüger-*location* übertragen und dabei in Orientierung und Auflösung den hier vorgegebenen Parametern angepasst. Nach Einrichtung der Gauß-Krüger-*location* wird zur folgenden Erzeugung der xy-*location* GRASS wieder beendet.

Jetzt gilt es, die für die gescannte Rohkarte benötigte xy-*location* zu erzeugen. Sie muss mindestens eine Ausdehnung bekommen, die der Pixelzahl in x- und y-Richtung der zu importierenden Karte entspricht. Die hier zu wählende Auflösung beträgt, wie üblich in xy-*locations*, generell 1 Pixel. Zwar steckt an sich eine geographische Auflösung (z.B. in Metern) dahinter, jedoch ist dieser Zusammenhang in der projektionsfreien xy-*location* unerheblich. Erst später, bei der Transformation in die Gauß-Krüger-*location*, wird die „echte“ Auflösung (die ja durch die Scanauflösung und den Kartenmaßstab festgelegt wurde) relevant.

Ein Hinweis für den Fall, dass die Karte aufgrund der Scannergröße nur in mehreren Teilen scannbar ist: In eine xy-*location* werden einfach alle Kartenteile importiert. Bei der Erzeugung dieser xy-*location* ist jedoch darauf zu achten, dass die Größe für alle Kartenteile ausreicht. D.h. ihre Größe ist entsprechend des größten gescannten Kartenteils zu wählen. *Locations* können in GRASS generell auch größer gewählt werden, so kann „auf Nummer sicher“ definiert werden. Um die Pixelausdehnungen in x- und y-Richtung zu erfahren, können Sie das im vorigen Abschnitt erwähnte Programm „xv“ verwenden.

Sie sollten jetzt zwei *locations* erzeugt haben: eine xy-*location* für die Rohkarte und eine Gauß-Krüger-*location*, in die die Rohkarte „entzerrt“, also geocodiert werden wird. Die Anleitung zur „Entzerrung“ der gescannten Karte finden Sie im nachfolgenden Abschnitt 6.6 („Vereinfachter Kartenimport bei gescannten Karten“). Bitte lesen Sie vorher zum Import der Rohkarte in die xy-*location* den Abschnitt 6.2 („Import eines Rasterbildes in eine xy-*location*“) oder einfach hier weiter bis zum Abschnitt 6.6.

6 Rasterdatenverarbeitung

Die Bearbeitung von Rasterdaten umfasst eine Unmenge an Möglichkeiten, es können alle wesentlichen algebraischen Funktionen und auch logischen Bedingungen bei Rasterdaten wie digitalen Karten, Satelliten- und gescannten Luftbildern, topographischen Karten, berechneten Flächen usw. angewendet werden. Da die Informationen punktweise neben- und untereinander angeordnet sind, ist einerseits die einzelflächenorientierte (rasterzellen-/pixelorientiert) und andererseits die matrixorientierte (rechteckiger Pixelverbund) Verarbeitung anwendbar. Vornehmlich werden in Geographischen Informationssystemen thematische Karten auf diese Weise aus vorhandenem Material abgeleitet.
Ein weiterer Vorteil der rasterweisen Datenverarbeitung liegt in der Analyse von Nachbarschaftsoperationen einzelner Punkte. Die Punkte können sich über Berechnungen gegenseitig beeinflussen, so dass sich beispielsweise Transportmodelle (Erosion, Wasserbewegung etc.) aufbauen lassen.
Bitte beachten Sie bei der Darstellung von Daten: Eine Rasterzelle hat zwei Koordinaten (x, y für den Zellenmittelpunkt) und einen Wert (z). Dieser kann, muss aber nicht in einer graphischen Darstellung einem fixen Farbwert entsprechen. Sie können Farbzuweisungen durch Farbtabellen (colortables) vornehmen.

6.1 Hinweise zum Einlesen von Rasterdaten in GRASS

Noch einmal sei der Hinweis gegeben, dass zwei Fälle zu unterscheiden sind:

1. Wenn die *location* samt Auflösung definiert ist und die Karte hineinpassen muss, richtet sich die Scanauflösung nach der gewünschten Auflösung (GRID RESOLUTION). Ein Berechnungsbeispiel ist im Abschnitt 5 zu finden.
2. Können sich die Parameter der *location*, insbesondere die Auflösung dagegen nach der gescannten Karte richten, sollte sie so angelegt sein, dass das Bild unverändert importiert werden kann. Vergleiche dazu Abschnitt 5.2.

GRASS unterstützt verschiedene Rasterformate. Es gibt beispielsweise eine Importmöglichkeit für Daten im TIFF-Format oder im PNG-Format. Eine Konvertierung von nicht unterstützten Bildformaten beispielsweise in das PNG-Format ist problemlos mit dem Programm `$ xv` möglich (dort einfach als neue Datei im PNG-Format abspeichern, der Knopf „Save at normal size“ in diesem Speicherfenster muss aktiviert sein), das auf jedem UNIX-System vorhanden sein sollte. Ansonsten

kann eine Konvertierung auch mit den netpbm-tools durchgeführt werden. Eine Übersicht der unterstützten Formate ist in Tabelle 6.1 angegeben. Erkennbar ist, dass nur über den ASCII-Import bzw. ARC-GRID negative Zellenwerte (DGMs etc.) im- und exportiert werden können. Fließkommadaten können ebenfalls nur mit diesen Formaten importiert (und exportiert) werden. Die Bildformate sind dagegen den „echten“ Bildern bzw. Daten ohne negative Werte und Kommazahlen vorbehalten.

6.2 Import eines Rasterbildes in eine xy-*location*

Sie werden regelmäßig Daten oder Karten vorliegen haben, bei denen Sie die Geocodierung, also die Randkoordinaten bzw. die Auflösung oder gar die Projektion nicht kennen. GRASS kann auch mit solchen Daten umgehen, sie werden in einer sogenannten xy-*location* verarbeitet. Hier werden keine Projektionsangaben benötigt, die Daten werden einfach pixelweise verarbeitet. Zwar entspricht die Pixelausdehnung natürlich einer geographischen Ausdehnung (z.B. in Meter), jedoch spielt sie hier keine Rolle. Sie können eine xy-location so groß wie den zu importierenden Datensatz (x- und y-Pixelzahl) einrichten, aber auch größer. Der Nullpunkt ist die untere linke Ecke, hier liegt ein importierter Datensatz immer verankert.

Der Import einer gescannten Karte, die im PNG-Format gespeichert wurde, wird nun anhand des GDAL-basierten Importmoduls vorgestellt:

Mit `$ r.in.gdal`
wird die Karte eingelesen. Dazu

1. die Bilddatei angeben (deren Namen müssen Sie entweder im Kopf haben oder in einem anderen Terminalfenster nachsehen), die Datei muss im aktuellen Verzeichnis gespeichert sein;

2. einen neuen Name angeben, unter dem GRASS die Datei in der GRASS-Datenbank ablegen soll: z.B.
   `tk25`

   „Verbose mode on?“ - „y“: Damit sehen Sie den aktuellen Importfortschritt, eventuelle Warnungen können Sie ignorieren, gemeldete Fehler natürlich nicht.

Alternativ können Sie in „TclTkGRASS“ importieren: „IMPORT“$\rightsquigarrow$„RASTER MAP“$\rightsquigarrow$„PNG“. Der Knopf „file“ im neuen Fenster öffnet einen kleinen Dateimanager zur Auswahl der Datei, in die zweite Zeile ist der neue Dateiname für GRASS einzutragen. „Verbose mode“ sollte angeklickt werden.

Modus	Format	Datenbereich	Datenstruktur	Besonderheiten
Import	ARC-GRID	alle Werte	Zellwerte	ARC/INFO-Format
	ASCII	-2 E^7 bis 2 E^7	x y z <Textattribut>	ASCII Text
	ERDAS	4, 8, 16 bit	Multiband	ab ERDAS 7.4
	GIF	8 bit	0..255	GIF 87
	PPM	24 bit	0..16,7 Mio	
	SUN-Raster	8 bit	0..255	
	TIFF	8 bit	0..255	unkompr. oder PackBits
	TIFF	24 bit	0..16,7 Mio	unkomprimiert
Export	ARC-GRID	alle Werte	Zellwerte	ARC/INFO-Format
	ASCII	-2 E^7 bis 2 E^7	x y z <Textattribut>	ASCII Text
	PPM	24 bit	0..16,7 Mio	
	ERDAS	4, 8, 16 bit	Multiband	ERDAS 7.4-Format
	PPM/3	3·8 bit	3 · 0..255	gesplittet in drei Dateien (R,G,B)
	TIFF	8 bit	0..255	unkomprimiert
	TIFF	24 bit	0..16,7 Mio	unkomprimiert
	TGA	24 bit	0..16,7 Mio	

Tabelle 6.1: Technische Angaben zu ausgewählten Formaten für den Rasterdatenimport und -export in GRASS (nicht genannt: die auch unterstützten Formate BIL, BSQ, HDF und weitere)

Nach dem Import müssen noch ein paar Informationen angegeben werden, die zur Verwaltung der Daten unerlässlich sind:
Das Modul `$ r.support`
berechnet diese fehlenden Informationen selbstständig. Rufen Sie das Modul auf, Sie werden dann gefragt:

1. „Edit the header?“ - „no“
2. „Update the stats (histogram,range)...?“ - „yes“ Nun werden Mimimum- und Maximumwerte der importierten Karte berechnet und gespeichert.
3. Die weiteren Fragen können Sie mit „return“ übergehen.

Nun können Sie mit der Karte weiterarbeiten oder sie betrachten. Zunächst wird jedoch der Import von Daten mit bekannter Geocodierung erläutert.

6.3 Import eines Rasterbildes in eine *location* mit Gauß-Krüger-Koordinaten

Liegt Ihnen ein Datensatz vor, von dem Sie auch die Geocodierung kennen (also seine Randkoordinaten und die Auflösung), können Sie diese Daten direkt in eine *location* mit Projektion (hier: Gauß-Krüger-*location*) importieren. Sind die Daten im ASCII- oder ARC-GRID-Format

gespeichert (Module: `$ r.in.ascii` bzw. `$ r.in.arc`), werden die Koordinaten automatisch mitübertragen. Anschließend ist nur noch `$ r.support` zur Berechnung statistischer Angaben aufzurufen („Edit header“: no, „update stats...“: yes, weitere Fragen: return). Im Folgenden das Vorgehen für den Fall beschrieben, dass Sie die Koordinateninformationen nur separat kennen.

Nehmen wir an, es liege eine Grundwasser-Flurabstandskarte in gewünschtem Ausschnitt und bekannter Auflösung vor, die Eckkoordinaten seien bekannt. Allgemein ist zu sagen, dass in einer *mapset* sämtliche Rasterdaten ihre individuelle Auflösung haben können. Dazu ist allerdings die jeweilige Auflösung **vor** dem Import bzw. vor jeder Bearbeitung der Daten mit `$ g.region` (Menüpunkt 1, GRID RESOLUTION, oder per Parameterübergabe) entsprechend einzustellen. Ansonsten wird in der gerade aktuellen Auflösung importiert oder gerechnet (Regelfall, der viele Vorteile bringt).

Nach der Einrichtung der *location* (entsprechend den gegebenen Koordinaten und GRID RESOLUTION) kann die Grundwasser-Flurabstandskarte importiert werden. Die Kartendatei muss im aktuell benutzten Verzeichnis (üblicherweise Ihr $HOME-Verzeichnis) liegen. Wie geht man nun vor? Der Import soll anhand des TIFF-Importmoduls gezeigt werden:

Mit `$ r.in.tiff`
wird die Karte eingelesen. Dazu wird

1. die Bilddatei angegeben (deren Namen man im Kopf haben muss oder in einem anderen Terminalfenster nachsehen kann), sie muss im aktuellen Verzeichnis gespeichert sein;
2. ein neuer Name angegeben, unter dem GRASS die Datei in der GRASS-*database* ablegen soll: z.B. `flurabstandskarte`;
3. „Verbose mode on?“ - „y“: Damit sehen Sie Details zum Import wie das Datenformat, gegebenenfalls eine Meldung, wenn es Probleme gibt (es existieren verschiedene TIFF-Unterformate) und den aktuellen Importfortschritt.

Alternativ können Sie in „TclTkGRASS“ importieren: „IMPORT“$\rightsquigarrow$„RASTER MAP“$\rightsquigarrow$„TIFF“. Der Knopf „file“ im neuen Fenster öffnet einen kleinen Dateimanager zur Auswahl der Datei, in die zweite Zeile ist der neue Dateiname für GRASS einzutragen. „Verbose mode“ sollte angeklickt werden.

Nach dem Import rufen Sie, wie oben beschrieben, das Modul `$ r.support` auf. Nach Angabe des Dateinamens werden Sie gefragt:

1. „Edit the header?“ - „yes“:
   Nun besteht die Gelegenheit zur Überprüfung der rows und columns. Die Werte sollten stimmen.

2. Mit „ESC“ gelangt man zum Koordinatenmenü.

3. Im Koordinatenmenü müssen die Randkoordinaten überprüft werden. Sollten die Werte des CELL HEADER auf 0 etc. stehen und nicht mit denen der DEFAULT REGION übereinstimmen, so sind sie zu korrigieren (sonst kann man nicht weiterarbeiten):
   Sofern die Karte so groß ist wie die gesetzte Region (Zoom etc.), können Sie einfach jeweils die Werte von den angegebenen Default-Werten übertragen (also insgesamt vier Werte). Dann geht es weiter mit „ESC“.
   Weitere Fragen mit „return“ beantworten. Jedoch sollte bei der Frage nach „update stats...“ mit „yes“ geantwortet werden.

Weitere Informationen zu den Möglichkeiten (Verändern der Farbtabelle etc.) des Moduls
`$ r.support` können dem GRASS User Manual entnommen (siehe Literaturliste) oder über die Onlinehilfe `$ g.manual` erfragt werden.

6.4 Betrachtung der importierten Rasterkarte

Die erfolgreich importierte Karte (unabhängig vom Koordinatensystem) lässt sich mit dem Modul `$ d.rast` betrachten. Vorher ist die Grafikausgabe (GRASS-Monitor) zu öffnen:

```
$ d.mon x0
```

Die Rasterausgabe auf dem GRASS-Monitor erfolgt mit:

```
$ d.rast
```

Hier ist zuerst der Name der anzuzeigenden Karte anzugeben („list“ liefert eine Übersicht der vorhandenen Rasterbilder). Die Frage nach dem Overlay-Modus (non-zero values only) bedeutet:

„no“ – evtl. im GRASS-Monitor bereits ausgegebenes Rasterbild überschreiben

„yes“ – nur Farben ungleich weiß ausgeben, also eine Bildüberlagerung im GRASS-Monitor durchführen. So kann man beispielsweise eine Linienrasterkarte einem Satellitenbild zur Ansicht überlagern.

Dann erfolgt die Kartenausgabe im GRASS-Monitor.

6.5 Hinweise zum Thema Auflösung von Rasterkarten

GRASS unterscheidet sich bezüglich der Handhabung von Auflösung und aktiver Region gegenüber anderen GIS-Systemen. Gegenüber ARC/INFO ist die Arbeit mit veränderter Rasterauflösung oder der Festlegung eines zu bearbeitenden bzw. zu exportierenden Ausschnitts aus einem Projektgebiet sehr einfach.
Bei GRASS wird immer im aktiven Ausschnitt gearbeitet. Das bedeutet, dass Ergebnisse innerhalb der gerade gesetzten Koordinatengrenzen berechnet werden, auch wenn die Eingangskarten

größer sind. Gleiches gilt prinzipiell auch für die Rasterauflösung: Jede Karte kann ihre eigene Rasterauflösung haben, eine neue Karte wird in den aktuellen Einstellungen berechnet. Ist intern eine Auflösungsanpassung notwendig, wird bei verminderter Auflösung der entsprechende Zellenwert gemittelt. Eine höhere Auflösung lässt sich de facto nur über eine Interpolation erzeugen. Wird dagegen die Auflösung einfach hochgesetzt, liegen zwar mehr Rasterzellen vor, benachbarte Zellen enthalten aber identische Werte.
Bei Import und Export von Rasterdaten gilt oben genanntes ebenfalls, Sie können also beispielsweise ganz einfach Kartenausschnitte mit gewünschter Auflösung exportieren. Hier liegt ein großer Vorteil der Rasterdatenverwaltung von GRASS gegenüber anderen Systemen. Die bei der *location*-Definition eingestellte Auflösung stellt nur einen Standardwert da. Sie können jeder Karte jedoch eine eigene Auflösung geben.

Die Einstellung des aktiven Ausschnitts (*region* genannt) bzw. der Rasterauflösung erfolgt mit

```
$ g.region
```

Nach Aufrufen des Moduls können Sie das Menü benutzen. Aktuelle Einstellungen können hier auch unter einem Namen gespeichert werden. Optional lässt sich das Modul auch kommandozeilenorientiert benutzen. Ein Beispiel:

```
$ g.region res=12.5 -p
```

setzt die Auflösung auf 12.5 Karteneinheiten (üblicherweise Meter).
Sie können auch die Werte von einer vorhandenen Karte übernehmen (Koordinatengrenzen und Rasterauflösung):

```
$ g.region rast=karte -p
```

Der Parameter „-p“ gibt die aktuell gesetzten Einstellungen aus („print“). Die Angaben, die Sie dann sehen, könnten beispielsweise sich so darstellen (Gauß-Krüger-Projektion):

```
projection: 99 (Transverse Mercator)
zone:       0
north:      5768850
south:      5764875
west:       3569550
east:       3574775
nsres:      25
ewres:      25
rows:       159
cols:       209
```

Wenn Sie auf die Standardeinstellungen (Koordinatengrenzen und Rasterauflösung) der *location* zurücksetzen möchten, geben Sie ein:

```
$ g.region -d -p
```

6.6 Vereinfachter Kartenimport für gescannte Karten

In diesem Anschnitt wird nun die „Entzerrung“ einer gescannten Karte auf eine topographische Referenz erläutert, Sie sollten jedoch vorher zum besseren Verständnis Abschnitt 5.3 gelesen haben. Als Dateiformat wird wie üblich das TIFF-Format oder PNG verwendet, die Zahl der Reihen und Spalten (rows, columns) kann entweder im Scanprogramm oder beispielsweise mit `$ xv` festgestellt werden. Das Programm `$ xv` zeigt die Zeilen- und Spaltenanzahl des Rasterbildes mit dem Menüpunkt „Windows“ ⇝„Image Info“ an.

Starten Sie GRASS mit Ihrer entsprechend den gescannten Kartenmaßen angelegten xy-*location*, in die die Rohkarte importiert wurde. Die gescannte Karte soll nun durch eine sogenannte Affin-Transformation[1] (mit dem Modul `i.rectify`) in die ebenfalls vorher angelegte Gauß-Krüger-*location* gebracht werden (vgl. Abb. 13). Zur Erläuterung: Die Affin-Transformation führt Drehungen, Streckungen (bzw. Stauchungen) durch und eignet sich für Rasterkarten, bei denen die innere Orientierung erhalten bleiben soll.

6.6.1 Geocodierung einer gescannten Karte

Nun wird die „Entzerrung“ der Rohkarte, also ihre Transformation zur Geocodierung, vorbereitet. Das hier vorgestellte Verfahren gilt nur für gescannte, unreferenzierte Karten. Haben Sie digitale, bereits geocodierte Karten vorliegen und wollen die Projektion wechseln, lesen Sie bitte Abschnitt A.5 für eine automatisierte Kartentransformation.

Die Vorgehensweise nach dem Import der gescannten Karte ist wie folgt: Die Karte muss in einer sogenannten „Bildgruppe“ eingetragen werden. Das ist einfach eine Liste der zu bearbeitenden Rasterdaten. Anschließend wird ein Transformationsziel festgelegt, das einer weiteren location entspricht, hier also unserer Gauß-Krüger-*location*. Nun müssen geographische Referenzen gesetzt werden, um dem Transformationsmodul anzugeben, welche neuen Koordinaten die einzelnen Pixel in der Gauß-Krüger-*location* bekommen sollen. Dazu werden die Ecken der Karte benutzt, den vier Eckpunkten der Karte in der xy-*location* weist man geograpische Koordinaten zu. Damit sind natürlich nicht die überstehenden Ränder, sondern die echten „geographischen“ Kartenecken gemeint. Sind keine Kartenränder in der gescannten Karte vorhanden, wenn Sie also nur ein Ausschnitt gescannt haben, können Sie auch die auf der Karte gedruckten „Gauß-Kreuze“ verwenden. Diese sollten den vier Kartenecken möglichst nah sein. Diese Punkte müssen unbedingt eine rechtwinklige Fläche umschließen, da der Ausschnitt mit der Affintransformation nur gedreht und gestreckt bzw. gestaucht werden kann. Sind auch keine „Gauß-Kreuze“ vorhanden, nehmen Sie

[1] Bei der Affintransformation handelt es sich um eine lineare Transformation, deren Parameter durch Regressionsrechnung ermittelt werden. Während der Transformation wird von `i.rectify`die Methode „nearest neighbor resampling“ verwendet. Nähere Erläuterungen erhält man mit dem Befehl `$ g.manual i.rectify`

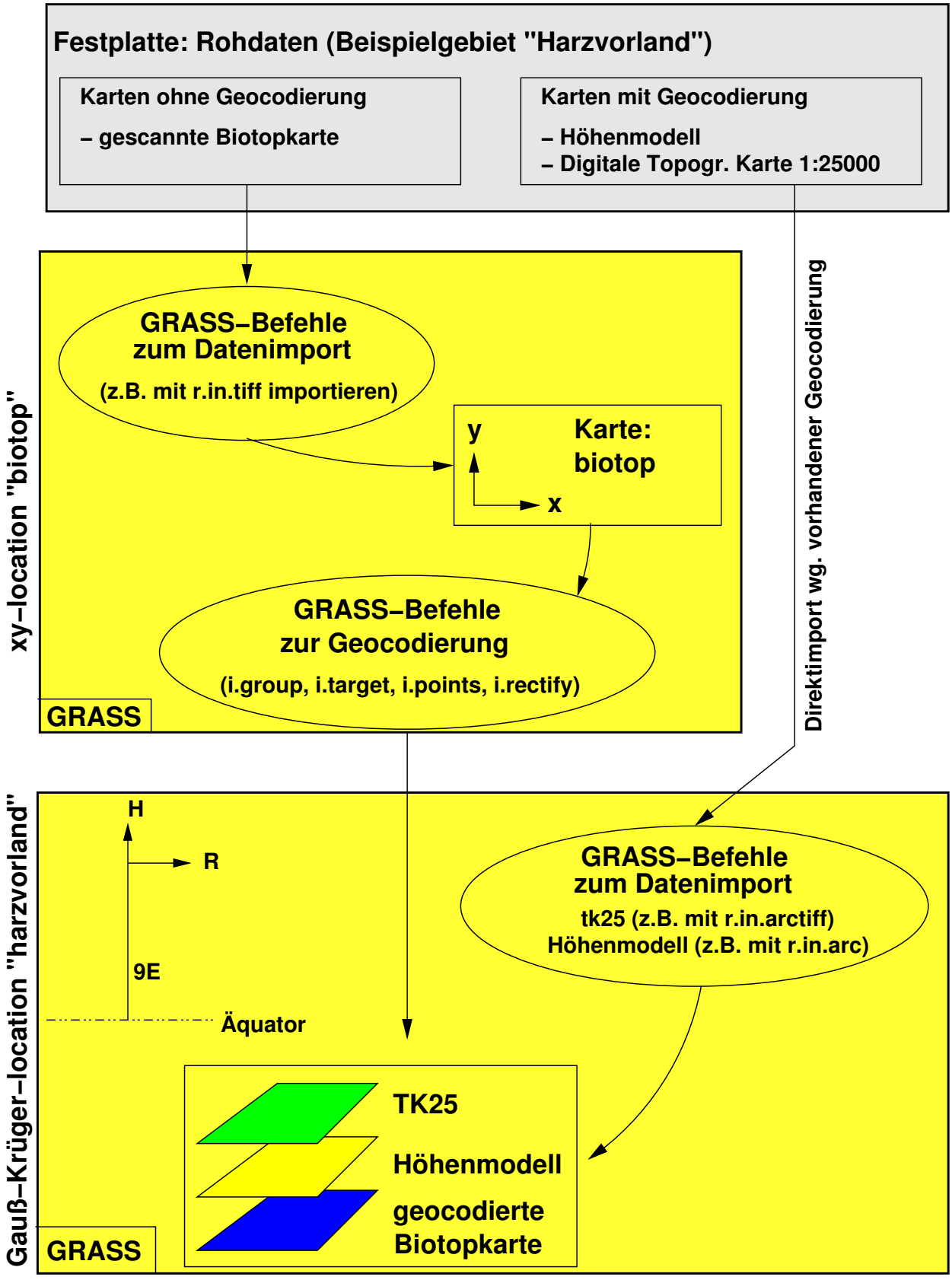


Abbildung 13: Import und Geocodierung gescannter Karten in GRASS

Straßenkreuzungsmitten oder andere, gut identifizierbare Punkte. Sind die geographischen Referenzen zugewiesen, erfolgt die Transformation der Rohkarte in die Gauß-Krüger-*location*. Im Einzelnen läuft der Vorgang so ab:

(a) Angabe des Namens der zu transformierenden Karte in einer neu anzulegenden Bildgruppe:
`$ i.group`

- Namen für diese Gruppe vergeben: z.B. „karte“.
- Mit einem „x“ die importierte Karte markieren.
- `$ i.group` mit „ESC return“ verlassen.

(b) Angabe der Ziel-*location* (target) und -*mapset* (die Gauß-Krüger-*location*, in die die Karte übertragen werden soll) in:
`$ i.target`

(c) Start eines GRASS-Monitors: `$ d.mon x0`

(d) Zuweisung der Gauß-Krüger-Koordinaten zu den vier Ecken des zu transformierenden Ausschnitts:
`$ i.points`

- Die zu transformierende Bildgruppe angeben (enthält nur die Karte), in diesem Beispiel „karte“.
- Im GRASS-Monitor geht es weiter. Dort wird nun die importierte Karte im Menü ausgewählt und dargestellt.
- Mit der Maus wählen Sie nun, so gut es geht, das erste „Gauß-Kreuz“ (erster Eckpunkt des zu transformierenden Ausschnitts etc.) und geben die zugehörigen Gauß-Krüger-Koordinaten im Textfenster ein (den zugehörigen Rechts- und Hochwert, den Sie am einfachsten Ihrer Papierkarte entnehmen können, durch ein Leerzeichen getrennt). So verfahren Sie mit allen vier Ecken. Hilfreich ist dabei die „ZOOM“-Funktion, um diese Passpunkte besser zu finden.

  Mittels der „ANALYSE“-Komponente kann der „rms-error“[2] überprüft werden, der nicht größer sein sollte als die GRID RESOLUTION in der Gauß-Krüger-*location*. Es werden alle Teil-rms-error zusammengerechnet zu einem Gesamtfehler. Gegebenenfalls besteht durch Ausschalten eines ungenau zugewiesenen Eckpunktes (per Doppelklick auf diesen Punkt im „ANALYSE“-Fenster) die Möglichkeit, diesen Punkt erneut zuzuweisen und damit den „rms-error“ zu verringern.

  Wenn Sie die vier Punkte erfolgreich zugewiesen haben, verlassen Sie `$ i.points`, die Koordinaten-Zuweisungen werden automatisch gespeichert.

[2] Der „rms-error“ ist der Abstand des gesetzten Passpunktes zur Lage des ideal richtigen Passpunktes. Die Berechnung erfolgt nach $rms = \sqrt{(x - x_{orig})^2 + (y - y_{orig})^2}$

(e) Jetzt wird das Transformationsmodul
`$ i.rectify`

gestartet. Als Polynomgrad wird 1 verwendet (als „order of transformation“). Damit wird eine lineare Transformation durchgeführt.

Zuerst werden die zu transformierenden Gruppe (hier: „karte“) und der Name der neuen Datei(en) angegeben. Nun wird erfragt, ob
(1.) in die „current region“ der Ziel-*location* oder
(2.) in die „minimal region“ transformiert werden soll.
Hier ist unbedingt Menüpunkt (1.) zu wählen, nämlich „current region“, die ja der gesamten Gauß-Krüger-*location* entspricht. Es würde sonst, falls die aktive Region in der Gauß-Krüger-*location* verändert wäre, evt. nur ein entsprechender Kartenausschnitt transformiert.

Die benötigte Rechenzeit liegt bei einer SUN SPARC/25 für eine Karte im DIN-A4-Format (mit 300dpi gescannt) bei rund fünf Minuten, auf Linux-PCs (ab 300MHz) geht es schneller.

GRASS kann, da UNIX multitasking-fähig ist, während der Transformation problemlos mit anderen Dingen beschäftigt oder auch verlassen werden, da die Rechnung im Hintergrund abläuft.

Ist die Transformation abgeschlossen (i.rectify melden sich mit einer email), sollte selbstverständlich der Erfolg dieser Übertragung in der Ziel-*location* im Gauß-Krüger-System überprüft werden (wenn Sie es noch nicht getan haben, verlassen Sie GRASS in der xy-*location* und rufen es erneut nun mit der Gauß-Krüger-*location* auf). Nach dem Starten eines GRASS-Monitors kann mit `$ d.rast` die transformierte Karte ausgegeben werden.

Das Modul `$ d.what.rast` ermöglicht in Kombination mit der Vergrößerung einzelner Kartenausschnitte (vgl. Abschnitt 6.9), die Koordinaten der „Gauß-Kreuze“ abzufragen. Sie sollten natürlich mit den Koordinaten auf der Papierkarte übereinstimmen. Ansonsten war vermutlich der „rms-error“ zu hoch. Eine Nachbesserung ist möglich: einzelne Eckpunkte sind in der xy-*location* verbessert zuzuweisen. In diesem Fall sollte die transformierte Karte in der Gauß-Krüger-*location* gelöscht werden (mit `$ g.remove`, siehe Anhang Abschnitt A.3.1), da sie während der verbesserten Neutransformation erneut erzeugt wird. Dann wechselt man in die xy-*location*, startet `$ i.points` (Bildgruppe und target sind noch gespeichert) und verbessert den „rms-error“ durch Korrektur einzelner Punkte. Anschließend ist `$ i.rectify`, wie oben beschrieben, erneut zu starten. In der Gauß-Krüger-*location* sollte nach Abschluss der Transformation das Ergebnis überprüft werden.

Ist die Übertragung gelungen, kann die nun nicht mehr benötigte xy-*location*, wie im Anhang (siehe Abschnitt A.3.1) beschrieben, entfernt werden.

6.6.2 Blattschnittfreie Geocodierung mehrerer gescannter Karten

Die im vorherigen Abschnitt beschriebene Transformation lässt sich auch zum blattschnittfreien Import mehrerer „Scans“ verwenden. Nur in seltenen Fällen ist der Zugang zu einem Trommelscanner möglich, normale Karten sind hingegen für handelsübliche Scanner zu groß.
Eine brauchbare Lösungsmöglichkeit besteht im Scannen der Karte in mehreren Teilen. Diese Lösung erfordert zwar etwas Aufwand, erspart aber den Kauf eines teuren Scanners. Dabei ist ganz wichtig, mit „Überlappung“ zu scannen, um die Eckkoordinaten besser festlegen zu können.

Wie oben bereits beschrieben muss eine Ziel-*location* im Gauß-Krüger-System erzeugt werden, die das Gesamtgebiet mit Koordinaten etc. umfasst. Die Auflösung sollte ausreichend hoch sein, damit kleine Kartensignaturen noch lesbar sind.
Anschließend werden alle gescannten Rohkarten in die xy-*location* importiert. Beachten Sie, dass die Größe der größten vorkommenden Rohkarte entsprechen muss.
Nun erzeugen Sie in der xy-*location* für jede Rohkarte eine Bildgruppe (auch wenn nur jeweils eine Karte darin ist) und setzen für alle Bildgruppen das Transformationsziel auf die Gauß-Krüger-*location* (Module `$ i.group` und `$ i.target`). Den folgenden Prozess durchlaufen Sie nun für jede Karte:
Den vier Eckpunkten der zu transformierenden Rohkarte weisen Sie Gauß-Krüger-Koordinaten zu (`$ i.points`, vgl. vorheriger Abschnitt). Diese Punkte müssen unbedingt eine rechtwinklige Fläche umschließen, da der Ausschnitt mit der Affintransformation nur gedreht und gestreckt bzw. gestaucht werden kann. Mit `$ i.rectify` (als Polynom 1. Grades) wird die jeweilige Rohkarte in das Gauß-Krüger-System transformiert.
Anschließend geht es mit der nächsten Rohkarte in der xy-*location* entsprechend weiter.

Sind alle gewünschten gescannten Karten transformiert, können Sie das Ergebnis im Gauß-Krüger-Projektgebiet (*location*) überprüfen (dazu xy-*location* verlassen). GRASS wird neu gestartet und der Name des Gauß-Krüger-Projektgebiets angegeben. Nun sollten Sie die Einzelkartenteile auf Lagegenauigkeit überprüfen: Zunächst wird ein GRASS-Monitor geöffnet und die Region erst einmal auf das Maximum (Defaultwert) eingestellt:

```
$ g.region -d
```

Danach ist unbedingt `$ d.erase` aufzurufen, um die Grafikausgabe von der Koordinatenänderung in Kenntnis zu setzen. Mit dem Aufruf von `$ d.rast` ohne Parameter erhält man nach Angabe der darzustellenden Datei die Möglichkeit, die Rasterbilder im GRASS-Monitor zu überlagern (Overlay-Modus: „yes“ oder Parameter „-o“). So wird jede Datei nacheinander ausgegeben, und die Karten sollten nebeneinander liegen. Mit `$ d.zoom` können Sie die Schnittkanten überprüfen. Idealerweise sind keine Schnittkanten mehr zu sehen.

Wie diese einzeln nebeneinander stehenden Rasterdateien zu einer einzigen Datei zusammengefügt werden, steht im Kapitel „Verschneidung verschiedener Flächen“ in Abschnitt 6.12.

Haben die Karten noch überflüssige Kartenränder, können sie auf einfache Weise ausgeblendet werden. Dazu zoomt man die jeweilige Karte passend mit

```
$ d.zoom
```

und korrigiert gegebenenfalls die neuen Koordinaten mit `$ g.region` um kleine Werte (um auf „angenehme“ gerundete Koordinaten zu kommen). Zur Speicherung erstellt man eine „Selbstkopie“, was im übernächsten Abschnitt beschrieben wird.

6.6.3 Import von Rasterdaten im ARC-GRID-Format

Der Import von Rasterdaten im ARC-GRID-Format (ESRI) stellt kein Problem dar, sie können mit dem Modul:

```
$ r.in.arc
```

importiert werden. Es wird allerdings nur das ASCII-GRID-Format, nicht aber das Binär-GRID-Format unterstützt. Das Modul fragt nach dem Namen der zu importierenden Datei und dem Namen, die sie in der GRASS-Datenbank bekommen soll. Neben einem Titel für die Karte kann auch festgelegt werden, ob sie als Integer- oder Fließkommakarte importiert werden soll.

Wenn Sie das Modul nicht-interaktiv, sondern mit Parametern benutzen, können Sie sehr angenehm mehrere Rasterkarten importieren. Mittels Cursortasten (Pfeil hoch, Pfeil herunter) lassen sich vorherige Kommandos benutzen und editieren (Pfeil links, Pfeil rechts).
Beispielsweise können Sie so zwei (oder mehr) digitale Höhenmodelle schnell importieren:

```
$ r.in.arc in=392619.asc out=392619.dgm12
$ r.in.arc in=392620.asc out=392620.dgm12
$ r.in.arc in=392626.asc out=392626.dgm12
```

Diese Einzelkarten lassen sich nun auch zu einer großen Karte im GIS zusammenfügen (kein Leerzeichen nach den Kommata!):

```
$ r.patch in=392619.dgm12,392620.dgm12,392626.dgm12  out=ildedgm12.5
```

Um die typische Farbgebung (von grün über gelb nach rot) für Höhenmodelle zu erhalten, geben Sie ein:

```
$ r.colors map=ildedgm12.5 col=gyr
```

Nun können Sie die Karte betrachten:

```
$ d.mon x0
$ d.rast ildedgm12.5
```

6.7 Erstellung kleiner Ausschnitte aus Rasterbildern

Häufig stellt sich das Problem, dass große Rasterbilder vorliegen, man aber nur einen kleinen Projektgebietsausschnitt braucht. Um die Rechenzeiten bei Analysen erträglich zu machen, lässt sich der interessierende Bereich ausschneiden und in einer eigenen Datei ablegen. GRASS hat

gegenüber proprietärer Software wie ARC/INFO den grossen Vorteil, immer nur im aktiven Ausschnitt zu rechnen. Daher können Sie einfach einen Ausschnitt aus Ihrem Projektgebiet wählen und dann hier weiterarbeiten bzw. diesen Teil exportieren (ohne das spezielle Datenbankabfragen gemacht werden müssen).

Dazu wird zunächst der Ausschnitt mit `$ g.region` oder `$ d.zoom` festgelegt. Auch das Modul `$ r.mapcalc` arbeitet nur im aktiven Ausschnitt.
Durch die einfache Rechnung

```
mapcalc> <neue Datei> = <alte Datei>
```

wird, wenn als „alte Datei“ der Name der großen Rasterdatei angegeben wird, der aktive Ausschnitt in der „neuen Datei“ abspeichert. Beispiel:

```
$ d.zoom
$ r.mapcalc
  mapcalc> innenstadt = hannover.tk25
```

Die Rasterkarte „innenstadt“ enthält nur den gezoomten Ausschnittsbereich und die entsprechende Geokodierung. Nun muss noch die Farbtabelle umkopiert werden.
In GRASS 5.0.x kann mit dem Modul `$ r.colors` eine Farbtabelle von einer Rasterkarte auf eine andere übertragen werden (hier z.B. von „hannover.tk25“ auf die Karte „innenstadt“):

```
$ r.colors map=innenstadt rast=hannover.tk25
```

In GRASS 4.x ist es noch etwas umständlicher, dort dürfen Sie ausnahmsweise einmal direkt in der GRASS-Datenbank arbeiten. Wechseln Sie in das entsprechende Verzeichnis der GRASS-Datenbank (in der Umgebungsvariablen $LOCATION ist immer der Pfad der aktuellen *location* und *mapset* gespeichert, die ja der Datenbankstruktur entspricht):

```
$ cd $LOCATION/colr
$ cp <alte Datei> <neue Datei> (z.B. $ cp hannover.tk25 innenstadt)
$ cd ~
```

Die Farbtabelle liegt also unter gleichem Namen wie die Rasterdatei im entsprechenden Verzeichnis.

Ganz wichtig ist bei r.mapcalc, immer mindestens einen Buchstaben in den Dateinamen zu haben, da sonst reine „Zahlennamen“ in Anführungsstriche gesetzt werden müssten (werden sie nicht mit angegeben, bekommt die neue Karte dann ausschließlich diesen Zahlenwert).

6.8 Export eines Rasterbildes

Der Export von Rasterdaten kann auf verschiedene Weise erfolgen. Es stehen die Exportformate ASCII (`$ r.out.ascii`), ARC-GRID (`$ r.out.arc`), PPM (`$ r.out.ppm[.cerl]`),

PPM/3 (`$ r.out.ppm3`), MPEG (`$ r.out.mpeg`), TIFF (`$ r.out.tiff`) und TARGA (`$ r.out.tga`) zur Verfügung. Platzsparender als ASCII, TARGA oder PPM ist allerdings das TIFF-Format, das direkt exportiert werden kann, oder das PNG-Format. Andere lassen sich über die netpbm-tools erzeugen (output auf „Standardout“).

Wie im vorigen Abschnitt angemerkt, wird nur der „aktive Ausschnitt“ exportiert. Damit können Sie auf einfache Weise Teilkarten exportieren. Die angegeben Module führen Sie mit Menüs durch den Vorgang. Etwas anders verhält es sich lediglich bei dem ARC-GRID-, ASCII- und PPM-Export, bei dem die Datenausgabe „umgeleitet“ werden muss (sogenanntes „UNIX-Piping“, vgl. Abschnitt 2.2.6):

```
$ r.out.arc map=rasterdatei > rasterdatei.grd
```

Der optionale Parameter „dp“ dient der Beschränkung der Dezimalstellen (hier auf zwei Stellen nach dem Komma):

```
$ r.out.arc map=rasterdatei dp=2 > rasterdatei.grd
```

Nun noch ein Beispiel für den PPM-Export mit Umwandlung in das GIF-Format durch die netpbm-tools (Sie können alternativ auch `$ xv` verwenden):

In GRASS 4.1.x:

```
$ r.out.ppm.cerl input=rasterdatei output=- | ppmtogif > export.gif
```

In GRASS 4.2.x:

```
$ r.out.ppm input=rasterdatei output=- | ppmtogif > export.gif
```

Die erzeugte GIF-Datei liegt dann im aktuellen Verzeichnis. Auch hier wird das sogenannte „UNIX-Piping“ eingesetzt, die Ergebnisse von r.out.ppm werden direkt in ppmtogif übertragen, dort verarbeitet und der Datenstrom in die Datei export.gif geleitet. Der Vorteil der „netpbm-tools“ besteht darin, dass Sie eine Umwandlung automatisch ohne manuelles Eingreifen, beispielsweise in Scripten, erledigen können.

Eine weitere Export-Möglichkeit besteht im ASCII-Sites-Format (als x-, y-Koordinaten mit jeweiligem z-Wert). Im Gegensatz zum ASCII-Raster-Export mit `$ r.out.ascii`, bei dem die Rasterzellenwerte als „Zahlenkolonnen“ ausgegeben werden, kann mit dem folgenden Befehl erreicht werden, zu jeder Zelleninformation die individuellen Koordinaten mit auszugeben:

```
$ r.stats -1 -g input=rasterdatei > hoehendaten.txt
```

Die Struktur der ASCII-Datei hoehendaten.txt ist bei diesem Export:

```
Rechtswert Hochwert Zellenwert
```

Optional kann das Label (jeweiliges Attribut der Rasterzelle) mit ausgegeben werden (vgl. Abschnitt 6.11).

6.9 Zoomen in einer Rasterkarte

Einzelheiten lassen sich mit der Maus vergrößern:

`$ d.zoom`

Mit der Maus kann nun ein Ausschnitt im GRASS-Monitor „gezogen“ werden. Benutzen Sie die linke Maustaste, um die erste Ausschnittecke festzulegen. Mit der mittleren Maustaste wird die gegenüberliegende Ecke festgelegt. Wichtig ist, immer die rechte Maustaste zum Abschluss des Zoom-Vorgangs zu betätigen, ansonsten wartet GRASS darauf und GRASS scheint blockiert zu sein. Die Maustastenbelegung wird immer im Textfenster angezeigt.

Wenn das Zoomen nicht den gewünschten Ausschnitt zeigt, können Sie mit der mittleren Maustaste herauszoomen.

Eine Möglichkeit, den aktiven Bereich auf das Maximum zu setzen, besteht darin, mit `$ g.region -d` (*default*-Einstellungen) den sichtbaren Ausschnitt auf die Größe des Gesamtgebiets zurückzusetzen. Anschließend ist unbedingt der GRASS-Monitor mit

`$ d.erase` zu löschen, ansonsten wird die Regionsänderung nicht durchgeführt. Nun wird mit

`$ d.rast` das ganze Rasterbild wieder ausgegeben. Über `$ d.zoom` kann wieder ein neuer Ausschnitt gewählt werden.

6.10 Automatisierte Rasterdaten-Umwandlung zu Vektordaten

Mit GRASS können Sie unter bestimmten Bedingungen automatisiert Rasterdaten in Vektordaten konvertieren. Es stehen zwei Module zur Verfügung: das erste wandelt linienhafte Strukturen in Vektorlinien um, das zweite Flächen in Polygone (vgl. Abb. 14).

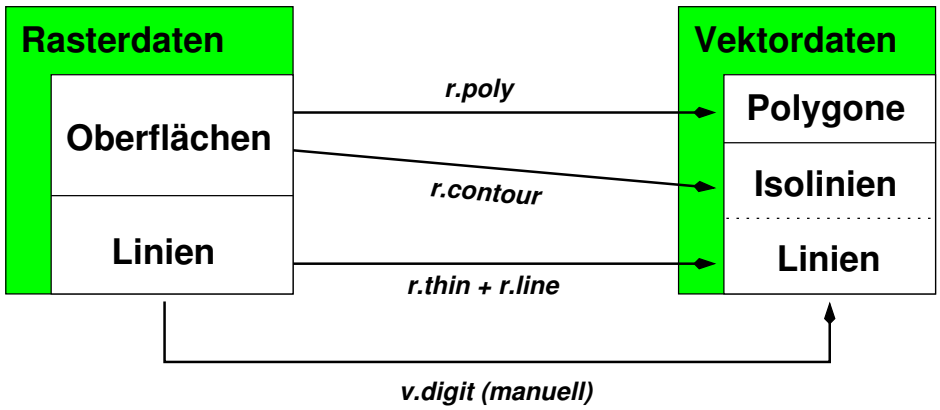


Abbildung 14: Wege zur Umwandlung von Raster- in Vektordaten

Ist beispielsweise ein Rasterbild über Reklassifizierungen (vgl. Abschnitt A.8.2) so verändert worden, dass nur noch die Höhenlinien sichtbar sind, können diese in Linienvektoren konvertiert werden. Dazu werden diese Linien zunächst „verdünnt“, um keine parallelen Vektoren zu erhalten:

```
$ r.thin
```

Die Linien besitzen nun nur noch eine Breite von einem Pixel. Im zweiten Schritt werden die Linien automatisch vektorisiert:

```
$ r.line
```

Für die Vektorisierung der Flächen wird keine weitere „Verdünnung“ durchgeführt, sondern direkt mit

```
$ r.poly
```

eine Vektordatei erstellt.
Diese Vektordateien können dann noch mit `$ v.trim` (bruchstückhafte Kurzvektoren entfernen) und `$ v.prune` (überflüssige Knoten entfernen) automatisch nachbearbeitet werden. Für weitere Arbeiten und zum manuellen Vektorisieren sollte `$ v.digit` verwendet werden.
Die dritte automatisierte Umwandlungsmethode, die Berechnung von Isolinien, wird weiter unten vorgestellt.

Wenn Sie daran interessiert sind, aus Rasterflächen eine Grenzlinienkarte ohne Flächeninhalte zu erzeugen, können Sie diese Umwandlung über das Vektorformat durchführen. Dazu ist mit r.poly eine Vektorflächenkarte zu erzeugen. Die weitere Konvertierung ist Abschnitt 7.11 beschrieben. Diese Grenzlinienkarte können Sie dann beispielsweise für Pufferungen (r.buffer) verwenden.

6.11 Rasterdaten-Wandlung zu Punktdaten

Rasterdaten, wie beispielsweise berechnete Höhenmodelle, sollen manchmal wieder in das Punktformat überführt werden. Dabei kann die Datei mit folgender Struktur exportiert werden:

```
Hochwert   Rechtswert    Zellenwert   optionales Attribut
```

Der Zellenwert entspricht dabei im Beispiel der Höhendaten der Höhe des jeweiligen Geländepunktes. Um die Punkte mit Koordinaten und ohne Attribut in einer GRASS-externen Datei

auszugeben, heißt der Befehl:

```
$ r.stats -1g Rasterdatei > Exportdatei.asc
```

Wenn das Attribut dabei sein soll, geben Sie folgenden Befehl ein:

```
$ r.stats -1gl Rasterdatei > Exportdatei.asc
```

Über das „Größer-als“-Zeichen wird das Ergebnis in eine Datei umgelenkt, die sich später im aktuellen Verzeichnis befindet. Mit

```
$ s.in.ascii
```

kann die Datei dann im Punktformat importiert werden (vgl. Abschnitt 8.2.2). Alternativ zu obigem r.stats/s.in.ascii-Befehl lässt sich auch `$ r.to.sites` verwenden (in GRASS 5 mit Option „-a“).

6.12 Verschneidung verschiedener Flächen

Um zwei Karten zu einer Karte miteinander zu verschneiden, wird

```
$ r.patch
```

verwendet. Die Verschmelzung der anzugebenden Dateien führt – je nach Reihenfolge – zu unterschiedlichen Ergebnissen: Die erste Karte wird dabei über die zweite gelegt usw. Beispiele finden sich bei ALBRECHT 1992.

Problematisch ist die Überlagerung bzw. das Aneinanderfügen von Bildrasterkarten mit jeweils 256 Farben, da es bei dem derzeitigen Farbmodell in GRASS die Darstellung von Karten mit mehr als 2500 Farben äußerst langsam wird. Alternativ zum GRASS-Script `$ i.image.mosaic` (zum Aneinanderfügen) ist der beste Weg die GIS-externe „Verschmelzung“. Dazu werden die Karten aus GRASS exportiert. Die Addition der Karten erfolgt mit einem externen Programm (z.B. mit den netpbm-tools: „pnmcat“), anschließend wird die neue Karte in ihren Farben reduziert, es findet eine Quantisierung der Ergebniskarte auf 8bit statt. Diese neue Karte kann dann wieder in GRASS importiert werden. Das externe Verschmelzen erfolgt so:

```
$ pnmcat -lr bild_a.ppm bild_b.ppm > bild_c.ppm
```
(nebeneinander: left right)

```
$ pnmcat -tb bild_a.ppm bild_b.ppm > bild_c.ppm
```
(übereinander: top bottom)

Mit `$ r.in.ppm` können die zusammengefügten Karten wieder importiert werden, alternativ lässt sich „bild_c.ppm“ mit `$ xv` in ein anderes Bildformat überführen.

Mathematische Operationen mit Rasterdaten können vor allem mit

```
$ r.mapcalc
```

durchgeführt werden. Zu diesem Modul existiert ein eigenes Handbuch (siehe Literaturliste).

Allgemein erfolgt eine Berechnung folgendermaßen:

```
<neuekarte> = <karte1> <math.  Operation> <karte2> ...
```

Eine Berechnung des „normalized difference vegetation index“ stellt sich beispielsweise innerhalb von r.mapcalc so dar (in GRASS 4.x mit 100 multiplizieren, um Kommawerte indirekt speichern zu können):

```
mapcalc> ndvi2 = 255.0 / 90.0 * atan ((tm4 - tm3) / (tm4 + tm3))
```

Es gibt einen alternativen Algorithmus:

```
mapcalc> ndvi = 1.0 * (tm4 - tm3) / (tm4 + tm3)
```

Ab GRASS 5.0.x ist darauf zu achten, dass nur Fließkommarasterdaten erzeugt werden, wenn auch Fließkommazahlen bzw. -karten in der Rechnung vorkommen. Die zweite NDVI-Formel ist also bei GRASS 5 mit 1.0 zu multiplizieren, um die Karte „ndvi“ mit Nachkommastellen zu erhalten. Da üblicherweise alle Satellitenkanäle als Integer-Karten (ganzzahlig) gespeichert sind, würde ohne den Faktor 1.0 das Ergebnis auch als Integer-Karte abgelegt. Sind Integerzahlen dabei und sollen dennoch (z.B. aufgrund der Rechenoperationen) die prinzipiell entstehenden Fließkommaergebnisse auch so gespeichert werden, so ist den Integerzahlen ein Punkt (amerikanisches Komma) nachzustellen. Ein Beispiel (Annahme: „altekarte“ sei eine Rasterkarte im „float“-Format):

```
mapcalc> floatmap = altekarte + 120.
```

Die Dateien `ndvi`, `ndvi2` bzw. `floatmap` stellen die neue Rasterdatei dar, `tm3` und `tm4` sind LANDSAT-TM-Kanäle, die als Rasterdateien vorliegen müssen. Nicht vergessen werden darf, dass Dateinamen nicht nur aus Zahlen bestehen sollten, da sonst diese Zahl als Wert interpretiert wird (es sei denn, sie steht in Anführungsstrichen). Noch einmal der Hinweis: **Jede GRASS-Rasterdatei sollte daher immer mindestens einen Buchstaben im Namen aufweisen.**

Die Unterschiede zwischen einer Verschneidung mit r.patch bzw. r.mapcalc zeigt die Abbildung 15. Bei r.patch läuft die Verschneidung auf der Basis kleinster Geometrien, bei r.mapcalc entsprechend der eingegebenen Formel. r.mapcalc bietet damit wesentlich mehr Möglichkeiten (vgl. auch Abschnitt 6.16).

Eine gute Möglichkeit, miteinander verschnittene Flächen anschließend auf Plausibilität zu prüfen, bietet das Modul:

```
$ r.cross
```

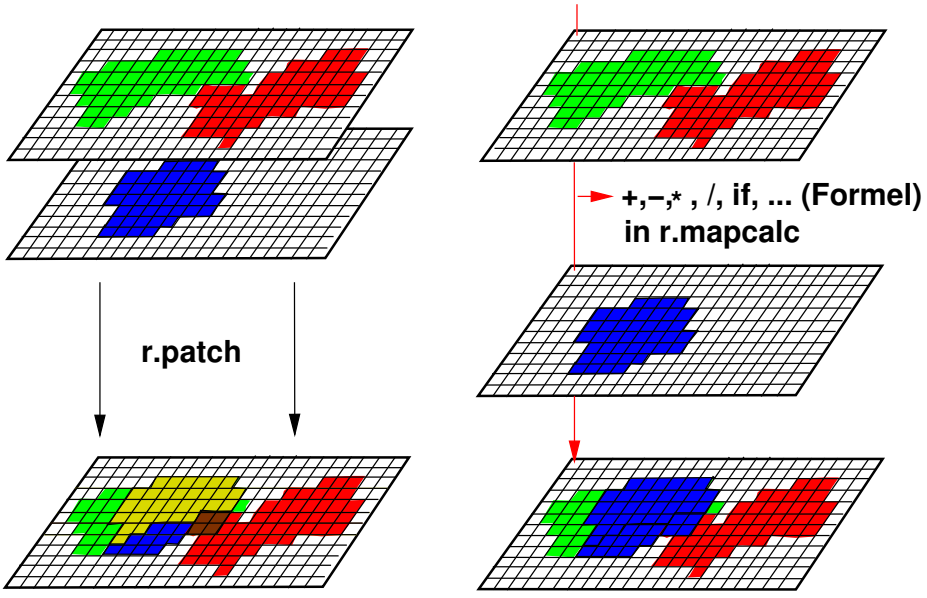


Abbildung 15: Auswirkungen von Rasterdaten-Verschneidung mit r.patch bzw. r.mapcalc

Als erste Karten werden die ursprünglichen Eingangskarten angegeben, die zum Verschneiden benutzt wurden (also mindestens zwei). Sind alle Eingangskarten eingegeben, wird einmal zusätzlich „return“ gedrückt, um diesen Abfragemodus zu beenden. Nun fragt das Modul nach einer neu zu erstellenden Kontrollkarte, in der die Vergleichsdaten gespeichert werden sollen.
Diese Karte kann man sich anschauen und z.B. mit `$ d.what.rast` abfragen. So kann die Berechnung der Flächen beim Verschneiden mit den Werten in der Resultatskarte verglichen werden.
Soll nur eine Tabelle der Category Label (Attribute) erstellt werden, hilft
`$ r.cats`
angewendet auf die Kontrollkarte aus r.cross weiter.

6.13 Interpolation von Rasterdaten

GRASS stellt Interpolationsmodule für zwei Anwendungsfälle zur Verfügung:

1. Die Rasterdatei ist lückenhaft und weist Nullwerte auf.
2. Die Auflösung der Rasterdatei soll verändert werden.

Die Interpolation verläuft für den *ersten Fall* folgendermaßen: Das Modul
`$ r.surf.idw2`
berechnet über eine Filterung mit einer bewegten Gaußschen Glockenkurve, die die Gewichtung eines Rasterwertes in Bezug zu seiner Distanz berücksichtigt (die Anzahl der in die Rechnung einbezogenen nächsten Nachbarn ist als „number of interpolation points“ wählbar, 12 Punkte sind der Standardwert). In einem wandernden „Fenster“ werden also die neuen Werte aus den Rasterzellenwerten der nächsten Nachbarn gewichtet nach ihrer Entfernung berechnet. Dabei kann die Auflösung der neu zu erstellenden Rasterdatei mit `$ g.region` (Menüpunkt 1, GRID RESOLUTION) voreingestellt, d.h. der zweite Anwendungsfall kann gleich mit einbezogen werden. Arbeiten Sie im Längen-/Breitengrad-System, nehmen Sie `$ r.surf.idw` statt `$ r.surf.idw2`.

Weist die Rasterfläche größere Lücken auf, ist zu überlegen, mit Splines zu interpolieren. Bei korrekter Steuerung des Algorithmus sind die Ergebnisse wesentlich besser als nach der IDW-Methode. GRASS bietet die Splines-Interpolation im Punktformat an. Dazu ist die Rasterkarte also mit `$ r.to.sites` in das Punktformat zu konvertieren. Nun wird die neue Auflösung mit `$ g.region` gewählt. Die Interpolation erfolgt dann mit dem Modul
`$ s.surf.rst`
Als Ergebnis erhalten Sie die interpolierte Fläche wieder im Rasterformat. Zur Überprüfung der Qualität der Interpolation kann optional sogar eine Karte der lokalen Abweichungen erzeugt werden. Für eine genauere Beschreibung vgl. auch Abschnitt 7.7 bzw. MITASOVA ET AL. 1993a/b, MITAS ET AL. 1999 und NETELER & MITASOVA 2002.
Je nach Datenstruktur müssen Sie sich für eine geeignete Methode entscheiden.

Beim *zweiten Anwendungsfall*, der Auflösungsverbesserung, können Sie entweder dieselbe Interpolationsmethode oder eine Spline-Interpolation durchführen. Für die Gewichtungsmethode wird zunächst die Auflösung, wie eben beschrieben, mit `$ g.region` neu festgelegt. Dann führt man die Auflösungsveränderung mit dem Modul `$ r.resample` durch (arbeitet wie r.surf.idw[2] mit *nearest neighbor resampling*).
Es gibt zwei alternative Algorithmen zur IDW-Methode: Die bilineare Interpolation und „Nächster Nachbar“ (*nearest neighbor*). Die bilineare Interpolation wird mit

```
$ r.bilinear
```

durchgeführt. Insbesondere die bilineare Interpolation, die zwar sehr schnell ist, eignet sich allerdings nur für geringe Auflösungsverbesserungen. Bei einer großen Änderung der Auflösung wird eine „Quaderstruktur“ im Ergebnis sichtbar.
Wollen Sie eine große Auflösungsveränderung durchführen, bietet sich eher das Splines-Verfahren an. Hier wird die Zielauflösung im Modul angegeben, sie ist also nicht vorher mit `$ g.region` einzustellen. Das Resampling-Modul mit der Spline-Methode heißt

```
$ r.resamp.rst
```

Neben der Interpolation kann dieses Modul optional auch noch geomorphologische Parameter wie Hangneigung, Hangexposition, und Profilkrümmungen (vertikal, horizontal und Mittelwert) berechnen. Zu Details der Splineinterpolation sollten die Aufsätze von MITAS ET AL. 1999 und MITASOVA ET AL. 1993a und 1993b gelesen werden[3].

6.14 Isolinienberechnung aus Höhenmodellen

Eine interessante Möglichkeit stellt die automatische Erzeugung von Isolinien (z.B. Vektorhöhenlinien aus Rasterhöhenmodellen) dar. Das benötigte Höhenmodell sollte bei möglichst hoher Auflösung interpoliert werden, um die „Treppenbildung“ der Linienvektoren zu minimieren. Das Verfahren ist im vorigen Abschnitt beschrieben.
Das Modul zur Erzeugung der Vektoren heißt:

```
$ r.contour
```

Die einfachste Bedienungsweise besteht darin, GRASS die Ermittlung der Minimal- und Maximalwerte zu überlassen und nur die Schrittweite (step) anzugeben, für die die Vektorlinien erzeugt werden sollen (Höhenstufen):

```
$ r.contour input=hoehenmodell step=Schrittweite output=hoehenliniendatei
```

Das Ergebnis liegt dann in der Vektordatei „hoehenliniendatei“ vor. Diese Datei muss noch mit `$ v.support` „behandelt“ werden, damit GRASS die internen Verknüpfungen ordnen kann. Der Aufruf erfolgt über:

```
$ v.support map=hoehenliniendatei option=build
```

[3] Im Internet online lesbar unter: `http://skagit.meas.ncsu.edu/~helena/index.html`

Die sinnvolle Dichte der Höhenlinien (und damit die Äquidistanz A) ist verständlicherweise von der Hangneigung α und dem Kartenmaßstab abhängig. Nach IMHOF 1965 (in HAKE ET AL. 1994, S. 382) berechnet sich folgendermaßen:

$$A = n * log\, n * tan\alpha \quad mit \tag{6.1}$$

$$n = \sqrt{\frac{M}{100} + 1} \tag{6.2}$$

Für M ist die Maßstabszahl einzusetzen, für α die Hangneigung in Grad, die für den vorliegenden Relieftyp gültig ist: Typische Hangneigungen sind für Gebirge: $\alpha = 45°$, für das Berg- und Hügelland $\alpha = 25°$ und für das Flachland $\alpha = 10°$. Entsprechend ist α zu wählen. Beispielsweise ergibt sich als Äquidistanz A [in m] im Berg- und Hügelland bei einem Maßstab von 1:50000 eine ideale Äquidistanz von rund 15m. Dieser Wert kann als *step* in r.contour eingesetzt werden.
Um die mittlere Hangneigung im Projektgebiet zu ermitteln, können Sie mit `$ r.slope.aspect` eine Hangneigungskarte berechnen und anschließend mit `$ r.univar` die Kartenstatistik abfragen (insb. das arithmetische Mittel).

6.15 Besondere Hinweise zu GRASS 5.0.x

GRASS 5.0.x besitzt ein neues Rasterdatenformat, es werden nun zusätzlich zu Ganzzahlen auch Fließkommazahlen unterstützt. Damit können beispielsweise Höhendaten sehr genau gespeichert werden, der bisherige „Trick“, Höhenangaben in Zentimetern statt in Metern anzugeben, muss nicht mehr angewendet werden.

Weiterhin werden im Rasterformat nun „0“ (ZERO) und „nichts“ (no data, NULL) unterschieden. Mit dem neuen Modul `$ r.null` kann ein Wert oder Wertebereich einer existierenden Karte nachträglich auf „nichts“ gesetzt werden:

```
$ r.null map=karte setnull=-9999
```

Sollen in einer Karte die „no data“ Flächen auf einen anderen Wert gesetzt werden, können Sie ebenfalls r.null benutzen. Ein Beispiel: hier werden alle Zellen mit NULL-Attribut auf „22“ gesetzt:

```
$ r.null map=karte null=22
```

Werden Daten mit `$ r.in.ascii` importiert, lässt sich der „Nichts“-Wert sofort angeben, z.B. kann der „No value“-Wert von ARC/INFO (-9999) so für GRASS ganz einfach über den „setnull“-Parameter umkodiert werden.
Sollen bei einer Rasterkarte die NULL-Werte entfernt, also in gültige Werte umgewandelt werden, verwenden Sie `$ r.support` (Nach Angabe des Kartennamens per „return“ bis zur Frage: „Do you want to create/reset null file for [map] so that all cell values are considered valid data?“: yes, weitere Fragen: mit „return“ quittieren).

Wenn Sie eine GRASS 4.x-Datenbank in GRASS 5.0.x verwenden wollen, sollten sie auf jede Vektorkarte `$ v.support` („Build“) und auf jede Rasterkarte `$ r.support` („update stats...“) anwenden, damit die interne Statistik aktualisiert wird. Bei Rasterkarten sollte auch der Befehl `$ r.null` benutzt werden, um die NULL-Datei zu erzeugen.

Speziell bei `$ r.mapcalc` gilt es, eine Besonderheit zu beachten: Möchte man eine Ergebniskarte im Fließkommaformat haben und befinden sich aber auch ganzzahlige Werte (Integer) in der Rechnung, so müssen die Integerwerte gefolgt von einem Dezimalpunkt (z.B. 123. statt 123) angegeben werden. Ein Beispiel:

```
$ mapcalc> neuekarte = altefliesskommakarte + 123
```

(„neuekarte“ wird im Integerformat erzeugt)

```
$ mapcalc> neuekarte = altefliesskommakarte + 123.
```

(„neuekarte“ wird im Fließkommaformat erzeugt)

Soll eine Integerkarte (ganzzahlige Rasterzellenwerte) in eine Fließkommakarte umgewandelt werden, muss sie ganz einfach mit 1.0 multipliziert werden:

```
$ mapcalc> neuekarte = altefliesskommakarte * 1.0
```

Das neue Modul `$ r.quant` dient zur Konvertierung einer Fließkommakarte in eine Integerkarte (wahlweise durch Runden oder Abschneiden).

6.16 Kartenalgebra mit r.mapcalc

Kartenverschneidungen auf Rasterebene bieten eine enorme Bandbreite an Analysemöglichkeiten. In GRASS kann mit dem r.mapcalc-Modul Kartenalgebra durchgeführt werden. Es arbeitet auch mit der „moving window“-Technologie, dabei bewegt sich ein Fenster definierbarer Größe suk-

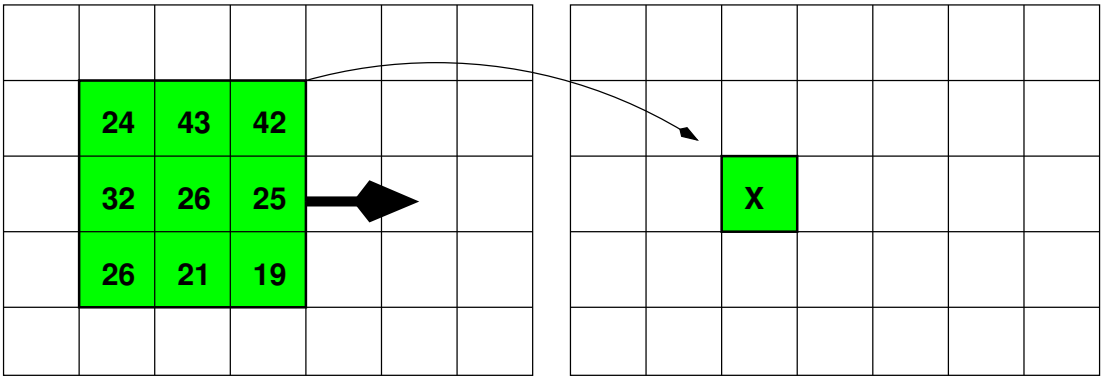


Abbildung 16: „Moving-window“-Methode bei Kartenalgebra im Rasterformat: Der Rasterzellenwert X in der neuen Karte berechnet sich hier aus einer 3x3-Matrix der alten Karte.

zessive durch alle Spalten und Reihen und verrechnet zellenweise Rasterkarten miteinander (vgl. Abb. 16). Ab GRASS 5.0.x wird zwischen NULL („no data“, also „nichts“) und 0 (ZERO) unterschieden.

Prinzipiell wird in `$ r.mapcalc` auf folgende Art gerechnet:

```
neuekarte = Wert/karte1 <formel> [karte2, ...]
```

Links vor dem Gleichzeichen steht der Name der zu erzeugenden Rasterkarte, rechts die Berechnungsformel. Das Modul kann also „intuitiv“ benutzt werden.

Es gibt folgende **Operatoren** in r.mapcalc:

%	modulus (Rest nach Division)
/	Division
*	Multiplikation
+	Addition
–	Subtraktion
==	logische Äquivalenz
!=	logische Nicht-Äquivalenz
>	größer als
<	kleiner als
>=	größer gleich
<=	kleiner gleich
&&	logisches UND
||	logisches ODER
#	teilt Farbrasterkarte in R, G, B-Anteile auf (3 neue Dateien)[4]

Folgende **Funktionen** stehen in r.mapcalc zur Verfügung:

abs(x)	absoluter Wert von x
atan(x)	Arcus-Tangens von x (x in Grad)
cos(x)	Cosinus von x (x in Grad)
eval([x,y,...,]z)	evaluiert die Werte des angegebenen Ausdrucks, Ergebnis wird in z gespeichert
exp(x)	Exponentialfunktion von x
exp(x,y)	x hoch y
x^y	Alternative für x hoch y
float(x)	wandelt x in Fließkommazahl um
if	Entscheidungsoperator
if(x)	1, wenn x nicht 0, sonst 0
if(x,a)	a, wenn x nicht 0, sonst 0

[4] Berechnung nach Gewichtung: 0.18*red + 0.81*green + 0.01*blue, vgl. Modulanleitung zu r.mapcalc (CIE-Formel)

if(x,a,b)	a, wenn x nicht 0, sonst b
if(x,a,b,c)	a, wenn x > 0, b wenn x gleich 0, c wenn x < 0
int(x)	wandelt x in Integerzahl [schneidet ab]
isnull(x)	ja, wenn Karte gleich „no data“ (NULL)
log(x)	natürlicher log von x
log(x,b)	log von x zur Basis b
max(x,y[,z...])	ermittelt größten der angegebenen Werte
median(x,y[,z...])	ermittelt Medianwert der angegebenen Werte
min(x,y[,z...])	ermittelt kleinsten der angegebenen Werte
mode(x,y[,z...])	ermittelt Moduswert der angegebenen Werte
rand(low,high)	erzeugt Zufallszahlen im Wert zwischen *low* und *high*
round(x)	rundet x zur nächsten Integerzahl
sin(x)	Sinus von x (x in Grad)
sqrt(x)	Quadratwurzel von x
tan(x)	Tangens von x (x in Grad)

Zusätzlich gibt es einige interne **Variablen** in r.mapcalc, die Werte beziehen sich auf das „moving window“, das die Berechnung durchführt:

x()	Variable für die laufende x-Koordinate
y()	Variable für die laufende y-Koordinate
col()	Variable für laufende Spaltennummer
row()	Variable für laufende Zeilennummer
nsres()	Variable für aktuelle Nord-Süd-Auflösung
ewres()	Variable für aktuelle Ost-West-Auflösung

Der Wert „keine Daten“ (NULL, „no data“) wird mit „null()“ angegeben. NULL ist von 0 (ZERO) verschieden.

Einfache Beispiele für die Grundrechenarten: Auf einfache Art und Weise können die geographisch korrespondierenden Zellenwerte von Karten miteinander addiert werden.

Beispiel 1: Addiere Gebäudestrukturen (als Gebäudehöhen in Karte „gebaeude“ gespeichert) zu einem Höhenmodell, so dass die Gebäude im Gelände „stehen“:

```
dgm_mit_haeusern = gebaeude + hoehenmodell
```

Beispiel 2: Berechung eines gewichteten Mittels aus zwei Karten (Dezimalpunkte, damit „neuekarte“ als Fließkommakarte erzeugt wird):

```
neuekarte = (5.  * karte1 + 3.  * karte2)/8.2
```

Einfache Beispiele für if-Bedingungen: („karte“ ist ein Kartenname, alle anderen Zeichen können Kartennamen *oder* Werte sein, die Rechnung erfolgt pixelweise):

1. if karte = a then b else c

in r.mapcalc: `neuekarte = if((karte == a),b,c)`

2. if karte <> a then b else c
in r.mapcalc: `neuekarte = if((karte != a),b,c)`

3. if karte >= a then b else c
in r.mapcalc: `neuekarte = if((karte >= a),b,c)`

4. if a <= karte <= b then c else d
in r.mapcalc: `neuekarte = if(((karte >= a) && (karte <= b)),c,d)`

Hier nun weitere **Beispiele für komplexere Datenselektionen** in Rasterkarten:

a) Selektiere aus „karte“ die Werte 1 und 2 und speichere in neuer Karte unter Beibehaltung der selektierten Werte, 0 sonst:

```
neuekarte = if((karte==1),1,0) + if((karte==2),2,0)
```

b) Selektiere aus „karte“ die Werte 1 und 2 und speichere in neuer Karte als Binärkarte ab (nur Werte 0 und 1 (also „ja“ und „nein“)):

```
neuekarte = if((karte==1),1,0) || if((karte==2),2,0)
```

c) Berücksichtigung von NULL-Werten („no data“, nichts):

i. Umwandlung ausschließlich von NULL-Werten in einen neuen Wert:

if karte=NULL then 3 else karte (wenn „no data“, dann 3 sonst Pixelwert übernehmen)
in r.mapcalc: `neuekarte = if(isnull(karte),3,karte)`

ii. Umwandlung aller Zellenwerte kleiner 5 in der Karte in den NULL-Wert, alle größeren in 5:
in r.mapcalc: `neuekarte = if(karte<5,null(),5)`

Eine weitere interessante Möglichkeit bietet die **Angabe relativer Koordinaten**, die für das „wandernde Fenster“ gültig sind. Damit können auch Nachbarzellen mit in die Rechnung integriert bzw. ein größeres/unsymmetrisches „moving window“ als die übliche 3x3-Matrix definiert werden. Beispiel:

```
neuekarte = altekarte[1,1] + altekarte[-1,-1]
```

benutzt nur die Zellen oben rechts [1,1] und unten links [-1,-1] in einer 3x3-Matrix zur Berechnung der neuen Karte. Diese Möglichkeit lässt sich auch bei verschiedenen Eingangskarten einsetzen. Mit „row()“ und „col()“ können die aktuellen Zeilen- und Spaltenwerte in der Formel integriert

werden, mit „x()“ und „y()“ die laufenden Koordinaten des „moving window“.

Zur einfacheren Editierung (Verwendung der Cursortasten) kann r.mapcalc auch kommandozeilenorientiert benutzt werden. Dabei ist die Formel in Anführungsstriche zu setzen. Ein Beispiel für die Erzeugung eines **Bildmosaiks**: wenn Koordinaten in x- und y-Richtung kleiner als angegebener Wert, dann LANDSAT-Kanal 1 (tm1) ausgeben, sonst an diesem Pixel die Topographische Karte (tk25) darstellen:

```
$ r.mapcalc 'neuekarte=if((x()<3571000 && y()<5767000),tm1,tk25)'
```

Hier wurde von Variablen Gebrauch gemacht, die die laufenden Koordinaten des „moving windows“ speichern. Problematisch ist hier nur, dass die Farben der Originalkarten nicht erhalten bleiben. Auf ähnliche Weise kann auch schnell ein Rechteck mit definierten Randkoordinaten und bestimmten Zellenwerten erzeugt werden.

Eine praktische Funktion ist „eval()“. Damit lassen sich Zwischenwerte speichern, ohne neue Karten erzeugen zu müssen und sich so mehrere Schritte in eine Formel bringen. Die einzelnen Formeln werden innerhalb der „eval()“-Umgebung mit Komma getrennt. Die letzte Anweisung wird hier immer als Ergebnis gespeichert. Interessant ist diese Funktion beispielsweise bei Abfragen von Fließkommakarten, wenn nicht exakte Bereiche definiert werden sollen. In diesem Beispiel werden die Kartenwerte erst gerundet, in „a“ zwischengespeichert und dann über eine Bedingung abgefragt:

(wenn gerundete Zellenwerte der Karte =3, dann 3 erhalten, sonst NULL setzen)

```
neuekarte = eval( a=round(karte), if(a==3,3,null()) )
```

Als komplexeres Beispiel soll noch einmal eine Bereichsselektion in einer Fließkommakarte gezeigt werden, bei der die NULL-Werte („no data“) erhalten bleiben, Speicherung der Zwischenwerte in die Variablen „zw1“ und „zw2“:

if a <= karte <= b then c else d mit Erhalt von NULL *(in einer Befehlszeile eingeben)*:

```
neuekarte = eval (zw1=round(karte), zw2=if(((zw1 >= a) && (zw2 <=
b)),c,d),
   if(isnull(karte), karte, zw2))
```

Zur Veranschaulichung noch einmal als Zahlenbeispiel:

if 4 <= round(karte) <= 5 then 20 else 66 mit Erhalt von NULL *(in einer Befehlszeile eingeben)*:

```
neuekarte = eval (zw1=round(karte), zw2=if(((zw1 >= 4) && (zw1 <=
5)),20,66),
   if(isnull(karte),karte,zw2))
```

Es muss selbstverständlich nicht vor Berechnungen bei Fließkommakarten gerundet werden, Sie können auch Bereiche oder explizite Werte angeben.

Beispiel: Erzeugung einer in Nordwest-Richtung geneigten Ebene mit Starthöhe 100m:

```
neuekarte = 100 + row() + col()
```

Weitere Beispiele zu r.mapcalc finden sich bei ALBRECHT 1992 und dem Tutorial zu r.mapcalc von SHAPIRO und WESTERVELT 1992.

6.17 Zuweisen von Attributen bei Rasterkarten

Eine häufiger benötigte Zuweisung von Attributen ist die Zuweisung der Flächenwerte als Rasterzellenwert. Hier wird mittels r.stats eine Zuweisungsdatei aus der Rasterkarten-Flächenstatistik (für jede Legendeneinheit, Option: -a) erstellt und durch „UNIX-Piping“ an das Modul r.reclass zur Erstellung einer neuen Karte übergeben (GRASS 4.x):

```
$ r.stats -a fs== in=karte | r.reclass in=karte out=neuekarte
```

In GRASS 5.0.x müssen, da das Ergebnis von r.stats im Fließkommaformat angezeigt wird, aber r.reclass nur mit Integerwerten reklassifizieren kann, die Werte über awk konvertiert werden (mit anderem Feldtrennzeichen, in einer Zeile einzugeben):

```
$ r.stats -an in=karte | awk '{printf "%d=%d\n", $1, $2}' |
   r.reclass in=karte out=neuekarte
```

Sollte die Meldung erscheinen „WARNING: can't read range file for [karte]“, ist die Kartenstatistik neu zu berechnen mit `$ r.support` („Edit header“: n, „Update stats“: y, weitere Fragen: „return“).
Die von r.stats erstellte und beim „UNIX-Piping“ and r.reclass übergebene Zuweisungstabelle für die neue Karte kann beispielsweise folgendermaßen aussehen (wenn mit -n als Option angegeben wurde gilt der Stern $*$ für alle NULL („no data“) Flächen):

```
1=81182.908163
2=1406.802721
3=7.993197
4=55037.159864
5=670621.258503
6=19.982993
*=660473.894557
```

Die Originalfarben können Sie mit `$ r.colors` (Option: rast, vgl. Abschnitt 6.7) kopieren.

Alternativ ist denkbar, dass die Gesamtflächen in Quadratmeter zusätzlich auch als erklärendes Textattribut zugewiesen werden sollen (um bei Abfragen mit d.what.rast etc. klare Aussagen zu bekommen). Bitte beachten Sie die Leerzeichen (Befehl in einer Zeile eingeben):

```
$ r.stats -a karte | awk '{printf "%d=%d Flaeche: %dm^2\n",$1, $1, $1 }' |
  r.reclass in=karte out=neuekarte
```

Das „awk“-Programm übernimmt hier die Erzeugung der Klassifizierungsregeln:

```
1=1 Fläche: 81182.908163m^2
2=2 Fläche: 1406.802721m^2
3=3 Fläche: 7.993197m^2
4=4 Fläche: 55037.159864m^2
5=5 Fläche: 670621.258503m^2
6=6 Fläche: 19.982993m^2
*=* Fläche: 660473.894557m^2
```

Der Nachteil dieser Zuweisung ist allerdings, dass immer die Gesamtfläche einer Legendeneinheit zugewiesen wird. Sollen objektweise die Flächen einzeln berechnet werden, können Sie mit
`$ r.clump`
zunächst eine neue Karte erzeugen, in der jede zusammenhängende Fläche mit einer laufenden Nummer versehen wird. Anschließend benutzen Sie obigen Befehl (mit oder ohne „awk“) unter Verwendung dieser neu erstellten Karte, um die Flächengrößen den Einzelflächen als Attribut zuzuweisen.

6.18 Speichern und Abfragen von Metainformationen bei Rasterkarten

Als Metainformationen werden Informationen über Datenqualität, -beschaffenheit, -genauigkeit, Hersteller usw. bezeichnet. Für die GIS-Arbeit ist eine Datendokumentation von höchster Bedeutung, um immer den Zustand eines Datensatzes beurteilen zu können. Das ist speziell bei Projekten, die sich über einen längeren Zeitraum erstrecken, die von mehreren Personen durchgeführt werden oder bei der Weitergabe digitaler Daten relevant. GRASS bietet als Möglichkeit der Dokumentation an, eine „History-Datei“ mitzuführen. Bei der automatischen Erzeugung neuer Karten durch Berechnungen mit GRASS-Modulen wird der zugrundeliegende Rechenschritt gleich automatisch in dieser Beschreibungsdatei eingetragen. Es kann jedoch wichtig sein, Zusatzinformationen abzuspeichern.
Die „History-Datei“ können Sie bearbeiten, indem Sie `$ r.support` aufrufen. Nach Angabe des Kartennamens geht es bis zur Frage: „Edit the history file for [karte]?“ mit der „return“-Taste weiter. Hier ist dann mit „yes“ zu antworten. Nun erscheint eine Eingabemaske für Metainformationen. Sie sollten speziell die Felder „Data source“ und „Data Description“ aktualisieren. Die

Datenbeschreibung kann auf der nächsten Seite fortgesetzt werden („ESC“-„RETURN“). Bei in GRASS berechneten Karten sind hier teilweise schon Parameter, die der Berechnung zugrunde lagen, gespeichert. Mit „ESC“-„RETURN“ wird auch diese zweite Eingabemaske verlassen. Weitere Fragen können mit „return“ übergangen werden.

Möchten Sie die Metainformationen zu einer Rasterkarte abfragen, benutzen Sie

```
$ r.info
```

Es erfolgt damit die Ausgabe der Datenbeschreibung.

7 Vektordatenverarbeitung

Vektordaten sind wichtige Kartenbestandteile, die in Form von Polygonen („areas“ im GRASS-Vokabular), Linien („lines“) und Punkten („sites“) auftreten können. Im Allgemeinen ist die digitale Einspeicherung von Karten in einigermaßen vertretbarem Zeitaufwand nur über den Weg der Digitalisierung von Flächenumrissen, linienhaften Verläufen von Straßen, Flüssen etc. und, wenn keine geographische Ausdehnung benötigt wird, punkthaften Attributen möglich. GRASS verwaltet Vektoren topologisch, als wesentliche Regel für die Praxis gilt, dass gleiche Flächengrenzen als eine Grenzlinie eingegeben werden müssen.
In näherer Zukunft, wenn die Mustererkennung weitere Fortschritte gemacht hat, lässt sich die Prozedur der Digitalisierung auch im handelsüblichen oder sogar freien GIS vielleicht einmal partiell automatisieren. Eventuell wird digitales Kartenmaterial auch einmal erschwinglicher. In diesem Fall könnten die gewünschten Informationsebenen direkt selektiert werden. Doch bis dahin ist die manuelle Digitalisierung häufig der einzige Weg zur digitalen Karte. Das folgende Kapitel beschäftigt sich unter anderem mit der Vektorisierung von analogem Kartenmaterial.

In diesem Kapitel wird nur auf Digitalisierung mit der Maus am Bildschirm eingegangen. Die Installation von Digitalisierbrettern ist im „GRASS 4.1 Installation Guide“ beschrieben, der auch für GRASS 4.x und GRASS 5.0.x in diesem Punkt Gültigkeit besitzt.

7.1 Warum werden Karten vektorisiert?

Eine Karte kann in einem Geographischen Informationssystem nur dann zu Analysen benutzt werden, wenn die Kartenobjekte (Straßen, Gebäude, Flächen etc.) als einzelne Objekte selektierbar

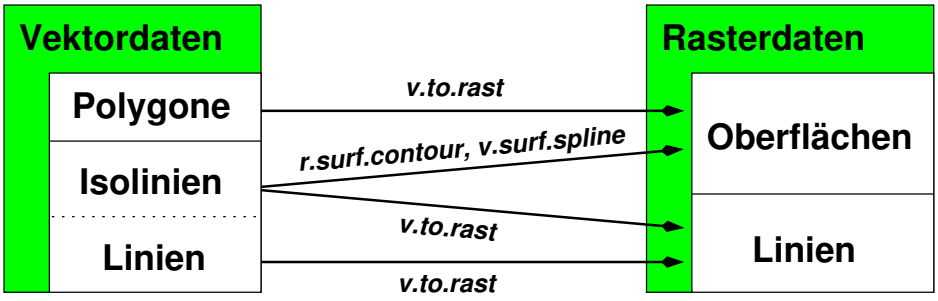


Abbildung 17: Wege zur Umwandlung von Vektor- in Rasterdaten

sind. Wurde beispielsweise eine Karte gescannt und dann in das GIS als „Bild“ (z.B. mit r.in.tiff) importiert, können die Kartenobjekte nicht selektiert werden. Alle Informationen liegen in diesem Fall als einzelne Rasterzellen vor, die nur singulär selektiert werden könnten. Zwar ließen sich homogene Flächen und Linien automatisiert vektorisieren (r.line, r.poly), es treten aber sofort Probleme auf, wenn sich Kartensignaturen überlagern (z.B. eine Straße in einem Waldstück). Eine Übersicht der Konvertiermöglichkeiten vom Vektor- in das Rasterformat zeigt Abbildung 17.
Um einzelne Objekte selektierbar zu machen, führt man eine Vektordigitalisierung durch und überträgt dabei die Objektgrenzen und Attribute in eine neue Karte. Es werden mit der (Digitalisier-) Maus die Grenzlinien um homogene Flächen und Verbindungslinien „nachgezeichnet“. Da die Arbeit im GIS geschieht, sind diese Vektoren automatisch geocodiert gespeichert.

7.2 Vektortypen im GIS

Mit Vektorlinien lassen sich linienhafte Objekte (Wege, Flüsse etc.) beschreiben. Vektorflächen stellen flächenhafte Informationen dar (beispielsweise Ackerparzellen oder Wasserflächen). Vektorlinien bzw. die Grenzlinien um Flächen sind üblicherweise aus Linienzügen aufgebaut. Sie haben zwischen den Linienenden Stützpunkte, die auch als „vertices“ (Singular: „vertex“) bezeichnet werden (vgl. Abb. 18). Die Linienenden sind die sogenannten „Knoten“ („nodes“). Vektorlinien und -flächen bekommen eine Attributnummer („category number“) und ein Textattribut („category label“). Diese Attribute werden bei Vektorlinien direkt verknüpft, bei Vektorflächen ist dagegen ein zusätzlicher „Labelpunkt“ innerhalb der Vektorfläche zu digitalisieren, der die Attributinformation trägt. GRASS kann intern ein Attribut pro Vektor verwalten, bei Anschluss einer externen Datenbank können es dann beliebig viele Attribute werden.

7.3 Die Vektorisierung in GRASS

Es gibt zwei Möglichkeiten der Kartenvektorisierung (häufig auch als „Digitalisierung“ bezeichnet):

- die Digitalisierung am Digitalisierbrett und
- die Digitalisierung am Bildschirm.

Beim **Digitalisieren mit einem Digitalisierbrett** wird die Karte auf ein Digitalisierbrett gelegt, die Eckkoordinaten per Mausklick angewählt und jeweils die zugehörigen Koordinaten per Tastatur im Digitalisiermodul angegeben (Kartenregistrierung). Der Vorteil der Digitalisierung am Digitalisierbrett besteht darin, dass die Übersicht auf dem Kartenblatt gewahrt bleibt. Der Nachteil ist der hohe Kaufpreis eines Digitalisierbretts und die nicht vorhandene Vergrößerungsmöglichkeiten bei schlecht lesbaren Kartensignaturen. Außerdem muss sich die Kartenvorlage in sehr gutem Zustand befinden, sie darf also nicht verzogen sein, um Lagefehlern vorzubeugen.

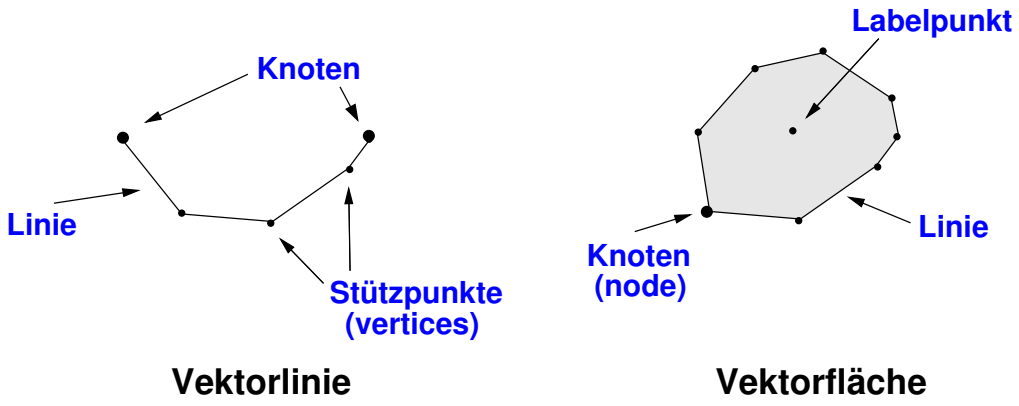


Abbildung 18: Vektortypen im GIS: Vektorlinie und Vektorfläche

Beim **Digitalisieren mit der Maus am Bildschirm** werden eine gescannte, geocodierte Rasterkarte auf dem GRASS-Monitor dargestellt und per Maus die relevanten Kartenobjekte digitalisiert. Die Kartenreferenzierung entfällt bei der Digitalisierung per Maus, da die Karte ja bereits geocodiert ist. Der Vorteil liegt darin, dass über die Änderung der ZOOM-Stufe Ausschnitte vergrößert werden können und damit die Digitalisierung sehr genau erfolgt. Neben einem Zugang zu einem Scanner sind keine Investitionen notwendig. Der Nachteil liegt in der etwas schwierigeren Orientierung auf der Karte am Bildschirm.
Prinzipiell wird im GIS auf folgende Weise digitalisiert (wichtig wegen der maßstabsabhängigen Generalisierungen in Karten):

- Linien: Vektorlinie in der Mitte der linienhaften Information (z.B. Straßenmitte)
- Flächen: Vektorlinie in der Mitte einer Flächengrenzlinie, zusätzlich Digitalisierung eines Labelpunktes in der Flächenmitte, wenn die Fläche geschlossen ist
- Gebäude: Lagetreu ist nur die Mitte eines Gebäudes, die Gebäudeausdehnung i.a. generalisiert und in ihrer Flächenbedeckung überproportional größer dargestellt als real vorhanden (Ausnahme: Katasterkarten)

7.3.1 Digitalisierregeln für topologische GIS

Für die Digitalisierung von Vektordaten aus analogem Kartenmaterial gelten in topologischen GIS wie GRASS einige Grundregeln. Sie sollten beachtet werden, um später die topologischen Funktionalitäten ausnutzen zu können. Näheres steht auch im *GRASS Programmer's Tutorial*.

- Linien sollten sich nicht ohne Knotenpunkt kreuzen,
- Linien, die einen gemeinsamen Knoten benutzen, müssen ihn exakt treffen (dazu ist die *snapping*-Funktion im Digitalisiermodul anzuwenden),
- Gemeinsame Grenzlinien von Flächen sollen nur einfach digitalisiert werden,

- Flächen müssen geschlossen sein (auch hier hilft die *snapping*-Funktion im Digitalisiermodul),
- Es ist zu überlegen, Linien und Flächen in getrennten Karten abzulegen, um Überkreuzungsprobleme zu vermeiden. Außerdem ist es einfacher, den Linien ihren Typ (Linie oder Flächengrenze) zuzuweisen.

Wenn diese Regeln beachtet werden, sollten kaum Probleme bei der Arbeit mit Vektordaten auftreten.

7.3.2 Digitalisieren von Karten

Wenn kein Digitalisierbrett vorhanden ist, aber ein Scanner in erreichbarer Nähe steht, lässt sich die Digitalisierung einer Karte problemlos durchführen. Wie eine gescannte Karte in GRASS übernommen wird, finden Sie im Abschnitt 6.6.

Als Ersatz für ein Digitalisierbrett kann die Maus eingesetzt werden. Dazu rufen Sie das Modul
`$ v.digit`
nach dem Start eines GRASS-Monitors auf. Als Digitalisierbrett wird „none“ ausgewählt. Nach der Angabe einer Vektordatei (neu oder bereits vorhanden) erscheint ein Formular mit Daten zu Person, Datum, Koordinaten etc. Verpflichtend muss der Parameter „Map's Scale“ von 1:0 auf den entsprechenden Kartenmaßstab gesetzt werden.

```
Provide the following information:

        Your organization    Geographisches Institut_____
        Todays date (mon,yr) Jan 25 99__________
        Your name            Emil_______________
        Map's name           realnutzung_____________________________
        Map's date           __________
        Map's scale          1:25000____
        Other info           Sommer 1998_____________________________
        Zone                 0____
        West edge of area    3568750_______
        South edge of area   5762774_______
        East edge of area    3574250_______
        North edge of area   5767726_______
```

Nach dem Ausfüllen dieser Seite (relevant ist: „Map's scale“) gelangt man zum Hauptmenü.

Die topographische Karte sollte nun in den Hintergrund gelegt werden, um sie als Digitalisiergrundlage verwenden zu können: Unter dem Menüpunkt „Customize“ (großes „C“) befindet sich

der Menüpunkt „Backdrop cell map“ (großes „B“), der eine Rasterkarte in den Hintergrund laden lässt.
Jetzt kann das eigentliche Digitalisiermenü mit „D“ wie „Digitize“ aufgerufen werden. Das Modul v.digit ist im Prinzip selbsterklärend, für speziellere Informationen sei auf das entsprechende *CERL-Tutorial: v.digit* verwiesen. Unter „Customize“ befindet sich u.a. ein „Color“-Menü, mit dem sich die Farben der digitalisierten Linien etc. verändern lassen. Damit können die Farben optimal sichtbar an den jeweiligen zu digitalisierenden Untergrund angepasst werden. Wichtig ist, den zu digitalisierenden Typ (mit „t“) für „Line“, „Area“, „Site“ zu wählen und gegebenenfalls das „Autolabeling“ einzuschalten. Mit dieser Funktion wird die Attributnummer *(category number)* gesetzt, der erklärende Attributtext *(category text)* kann (bisher) nicht direkt in v.digit eingetragen werden. Daher sollten Sie immer mitschreiben, welche Attributnummer welchem Objekttyp entspricht. Die Attributtexte werden erst nach der Arbeit in v.digit mit dem Modul v.support („Edit the category file“) eingetragen.
Nicht zu vergessen ist der Hinweis auf die „Zoom“-Funktion, die sogar eine genauere Digitalisierung als mit einem Brett zulässt.
Um Verwirrung vorzubeugen: Punkte lassen sich sowohl als Vektorpunkte (in v.digit) als auch im eigenen Punktformat in GRASS speichern. Auf Letzteres wird im nächsten Kapitel eingegangen. Die Punkte im Vektorformat sind üblicherweise das Resultat von Digitalisierungen, die Punkte im eigenen Sitesformat Ergebnis von Importierungen aus Dateien mit Koordinaten (Hoch-, Rechtswert) und Attributen.
Eine wichtige Grundregel im topologischen GIS besteht darin, Grenzlinien, die benachbarte Flächen trennen, nur einfach zu digitalisieren. GRASS ordnet diese Grenzlinie gleichzeitig beiden Flächen zu. Keinesfalls dürfen zwei parallele Linien digitalisiert werden.

7.3.3 Digitalisieren von Flächen

Da eine Flächendigitalisierung erfahrungsgemäß am Anfang ohne Übung etwas problematisch ist, soll hier konkret in Bezug auf das Setzen von Attributen eingegangen werden. Im Digitalisiermodul werden die Vektortypen im Menü „Digitize“ gewählt.
Mit „t“- *toggle type* können Sie zwischen

- Linien (LINE),
- Flächengrenzlinien (AREA EDGE)
- und Punkten (SITE)

selektieren. Für Flächen ist essentiell wichtig, dass sie geschlossen sind. Nur dann können die GIS-Vektortopologie aufgebaut und gesetzte Labelpunkte zugewiesen werden.
Zum vereinfachten Schließen von Flächen gibt es eine *Snapping*-Funktion, da es quasi unmöglich ist, Linienenden exakt zu treffen:

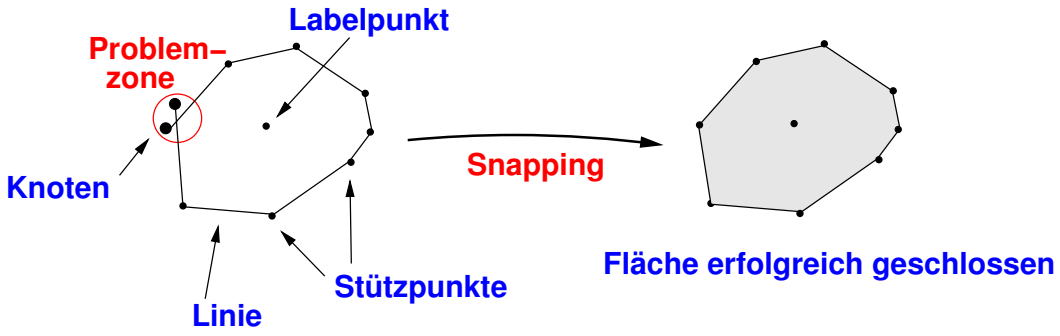


Abbildung 19: Die *Snapping*-Funktion im GIS

In v.digit ist im Hauptmenü die Option „Customize“, darin wiederum die Option „*s*“ - *Set snapping threshold*. Mit einem geeigneten Wert für „snapping threshold“ kann erreicht werden, dass beim Digitalisieren von Flächen die Linienenden richtig aufeinander „gezogen“ (neudeutsch: gesnappt) werden. Nur im Falle eines geschlossenen Linienzuges entsteht eine Vektorfläche, der auch ein Attribut zugewiesen werden kann. Prinzipiell sieht der Vorgang so aus: Es ist eine dem Kartenmaßstab angemessene „Snap-Distanz“ zu wählen. Da das Snapping unmittelbar beim Digitalisieren erfolgt, sind die Auswirkungen sofort zu sehen. Die Angabe ist in Inch zu treffen, sie wird aber im Menü sofort umgerechnet und in Karteneinheiten (Meter) angegeben. Folgende Snapdistanzen sind sinnvoll (maßstabsabhängig!):

Bei 1:5.000 - 1:10.000: Snapdistanz 1-2m (Angabe in Inch im Menü: 0.0017 bis 0.002)

Bei 1:10.000 - 1:25.000: Snapdistanz 2-5m (Angabe in Inch im Menü: 0.002 bis 0.008)

Bei 1:25.000 - 1:50.000: Snapdistanz 5-10m (Angabe in Inch im Menü: 0.008 bis 0.017)

Erscheint hier keine sinnvolle Distanz (in Meter) im Menü, wurde der Karte vermutlich eingangs beim Aufruf in v.digit kein Maßstab zugewiesen. Beenden Sie in diesem Fall v.digit und rufen es erneut mit dieser Karte auf. Im Definitionsbildschirm kann der Kartenmaßstab entsprechend gesetzt werden.

Folgende Problemfälle treten bei der Digitalisierung häufiger auf:
Wenn die Snapdistanz nicht geeignet gewählt wurde oder die ZOOM-Stufe zu klein war, kann es vorkommen, dass Linien nicht korrekt aneinander anknüpfen. Es wird von „*undershoot*“ gesprochen, wenn die Linie zu früh endet, von „*overshoot*“, wenn sie zu lang ist (vgl. Abb. 20).
Prinzipiell kann hier über „Snapping“ eine Vektor-Verknüpfung erreicht werden. „Snapping“ ist eine Funktion in v.digit (Snapdistanz dort entsprechend des Kartenmaßstabs einstellen) bzw. in v.support. Für das „*overshoot*“-Problem gibt es das GRASS-Modul `$ v.trim`. Es lässt sich beim Digitalisieren durch die Wahl der geeigneten Snapdistanz minimieren.

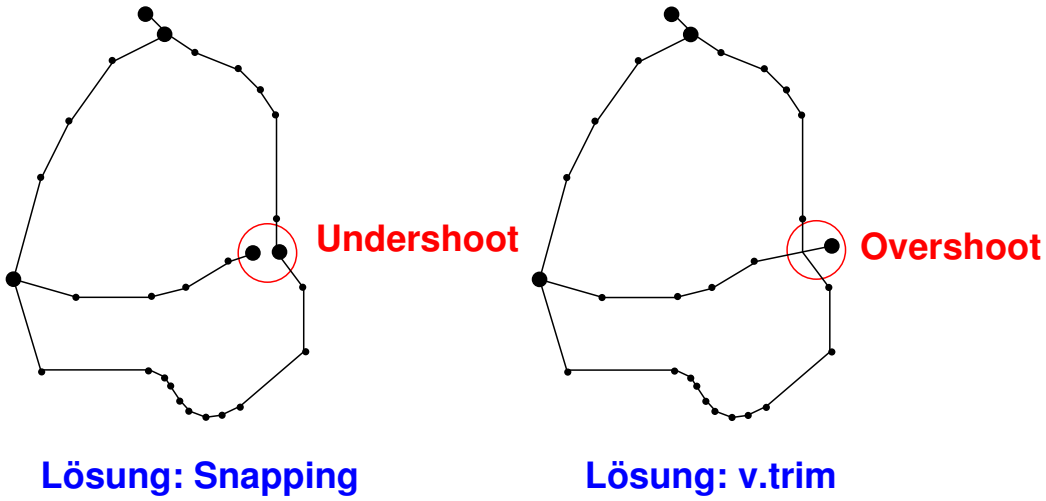


Abbildung 20: „Over-“ und „undershoots“ bei der Vektordigitalisierung

Ein Hinweis: „*Overshoots*“ werden manchmal absichtlich eingesetzt, um Flächengrenzen auf jeden Fall zu schließen. Allerdings müssen sie anschließend korrigiert werden, um keine überstehenden Linienenden zu bekommen. Bei der Digitalisierung am Digitalisierbrett wird häufig ein das Projektgebiet umschließendes Rechteck („*neatline*“) erzeugt. Der Sinn liegt darin, die innenliegenden Flächengrenzen mit gewolltem „*overshoot*“ über das Projektgebiet hinaus zu digitalisieren und später automatisiert zu korrigieren. In diesem Falle bildet das Rechteck dann einen Teil der Flächengrenzlinien.

Generell gilt, nach der Benutzung von v.digit (bzw. v.displine) immer `$ v.support` zu benutzen, um die Topologie der Vektordaten (neu) aufzubauen. Hier können Sie dann auch zu den Attributnummern gehörigen Attributtexte eintragen („Edit the category file“). Haben Sie sehr viele Attributnummern, lassen sich in der letzten Zeile der Eingabemaske die zu editierenden Zeilennummern direkt angeben. Nach Eingabe von „ESC“-„RETURN“ springt v.support zu dieser Zeile.

Nachbearbeitung von digitalisierten Karten:

Da es nicht ganz einfach ist, bei Flächendigitalisierungen immer genau die Linienendpunkte zu treffen (eine Fläche ist ja eine geschlossener Linienzug, also eine Linie, die mit beiden Enden in einem Punkt aneinanderstößt), gibt es ein GRASS-Modul, um diesem Problem entgegenzutreten:
`$ v.spag`

Eine weitere Eigenschaft des v.spag-Moduls ist, bei Kreuzungen fehlende Knoten einzufügen (gemäß der Digitalisierregeln, vgl. Abb. 21). Jedoch ist dieser Befehl mit äußerster Vorsicht zu benutzen. Am besten sollte vorher eine Kopie der Vektorkarte mit g.copy erzeugen. Bearbeitungen mit v.spag können nicht rückgängig gemacht werden.

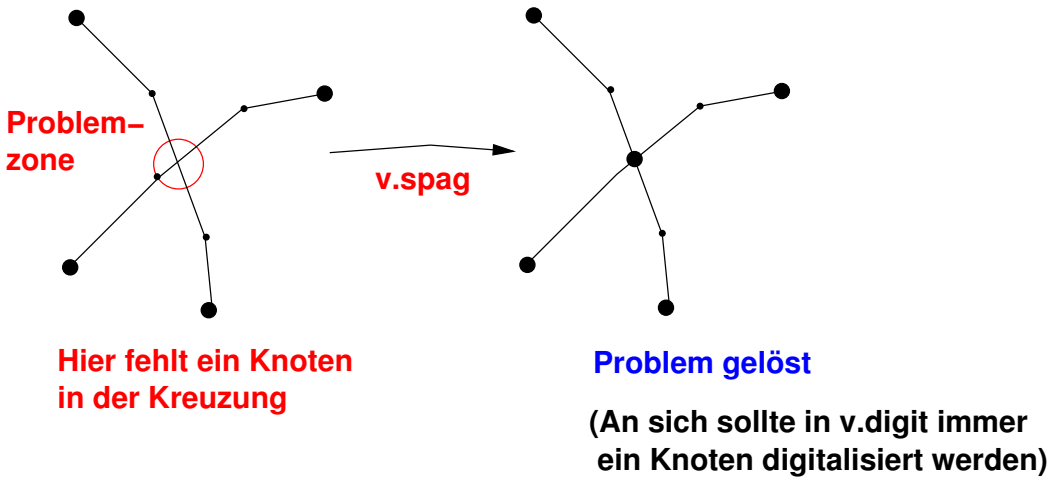


Abbildung 21: Korrektur von „Spaghetti-Digitalisierung“

Weitere Module sind `$ v.clean` und `$ v.prune`. Mit v.clean lassen sich „tote Linien“ aus einer Vektorkarte entfernen. Dabei sind „tote Linien“ Vektoren, die in v.digit als gelöscht markiert wurden. Mit dem Modul v.prune können überflüssige Knoten aus Vektorlinien entfernt werden. Das Modul ist mit Vorsicht zu genießen; wenn man zu viele Punkte entfernt, werden komplexe Flächen zu einfachen geometrischen Formen wie Dreiecken oder Rechtecken „gesäubert“.

7.3.4 Digitalisieren von Höhenlinien

Höhenlinien werden als Linienvektoren digitalisiert. GRASS bietet eine erleichternde Möglichkeit, Höheninformationen der Linien mit wenig Aufwand nach der Vektorisierung zu setzen:
Es werden zunächst die Linien ohne Attribute digitalisiert. Anschließend wechselt man mit „L“ in das „Label“-Menü. Dort ist der Menüpunkt „i“ - „Contour interval“. Hier muss zunächst das Linienintervall gewählt werden. Vorgegeben sind 5 Grundeinheiten pro Linie, i.a. ist die Grundeinheit Meter (also 5m). Mit „c“- „Label Contours“ kann nun das einfache Setzen beginnen. Generell läuft der Vorgang so ab, dass die beiden äußeren Höhenlinien einer Linienschar attributiert werden und alle dazwischenliegenden Linien automatisch entsprechend dem gesetzten Intervall ebenfalls ihre Höheninformationen bekommen. Zwischen dem ersten gesetzten Punkt (Auswahl der Linie) und dem zweiten wird während dieser Attributierung eine Gerade gezogen. Alle Linien, die von dieser Geraden geschnitten werden, erhalten ein Attribut. GRASS führt auch sofort eine Plausibilitätskontrolle durch, das Intervall und die Anzahl der Linien zwischen den markierten Linien müssen zueinander passen. Dabei muss man darauf achten, nicht zu nahe an Linien zu geraten, die nicht „gelabelt“ werden sollen, da die Gerade einen gewissen Suchradius hat. Diese sogenannten „Thresholds“ können, falls nötig, im „C“ - „Customize“-Menü eingestellt werden.
Mit der linken Maustaste wird also eine äußere Linien der Höhenlinienschar angeklickt (im „Tal“ beispielsweise) und mit der rechten Taste bestätigt. Nun gebe man die Höhe (das Attribut) dieser Linie an. Mit der Maus fährt man nun auf die andere Seite der Höhenlinienschar (auf den „Berg“),

wählt mit der linken und rechten Maustaste wieder eine Linie aus. Die beiden Linien sind nun über eine Gerade verbunden, die die dazwischenliegenden Höhenlinien schneidet. Auch hier muss nun eine Höhe angegeben werden. Alle dazwischenliegenden Linien erhalten dann automatisch ihren entsprechenden Wert. Passen das gewählte Intervall und die Anzahl der Linien nicht zusammen, gibt GRASS eine Warnung aus und unterbricht den Vorgang. Dann sollte nochmals genau nachgezählt bzw. stückweise vorgegangen werden.

7.3.5 Setzen von Attributen (Labels) in Vektorkarten

In GRASS sind, wie bereits angemerkt, Attributnummern und Attributtext zu unterscheiden. Das Setzen der „category labels“ (Attributtext) kann in GRASS mit dem Modul `$ v.support` erfolgen. Es erlaubt die Bearbeitung der Textattribute in Tabellenform. Mit der Option „Edit the category file“ erreichen Sie die Tabelle. Haben Sie sehr viele Attributnummern, können Sie in der letzten Zeile der Eingabemaske die zu editierenden Zeilennummern direkt angeben. Nach Eingabe von „ESC“-„RETURN“ springt v.support zu dieser Zeile.
Das Modul `$ v.alabel` ermöglicht das automatische Setzen von Attributnummern für alle ungelabelten Vektoren. Sie werden dabei auf einen Wert gesetzt. Alternativ kann `$ v.reclass` verwendet werden, mit dem eine Reklassifizierung über die Attribute möglich ist. Manuell lassen sich Vektoren mit `$ v.digit` editieren.
Ist einmal in einer Vektorkarte der Maßstab nicht korrekt gesetzt, kann dafür `$ v.digit` benutzt werden. Sie können den korrekten Maßstab im Datenformular zur jeweiligen Karte eintragen, nachdem Sie die zu verändernde Vektorkarte ausgewählt haben. Anschließend verlassen Sie v.digit wieder. Der Maßstab ist für viele Vektorberechnungen relevant und sollte daher korrekt sein.

7.4 Import und Export von Vektordaten

Vektordaten liegen sehr häufig in einem ESRI-Format vor: *SHAPE-Format, E00-Format* oder das ASCII-„*ungenerate*“-Format. Der Import von SHAPE und E00 ist im Anhang A.4 beschrieben. Vektordaten können aus GRASS im „*ungenerate*“-Format auch exportiert werden.

Ein weiteres Format ist das *ASCII-Vektorformat*. Mit dem Modul
`$ v.out.ascii` werden Vektordaten im ASCII-Format exportiert. Die Exportdatei findet sich im Verzeichnis
`$LOCATION/dig_ascii`
Die zugehörigen „categories“ sind bereits im ASCII-Format, die entsprechenden Dateien in den Verzeichnissen
`$LOCATION/dig_cats` und
`$LOCATION/dig_att`
können also direkt kopiert werden. Im Verzeichnis `$LOCATION/dig_cats` werden die Text-Label abgelegt und in `$LOCATION/dig_att` die Punkte, an denen die Label für die areas (Polygone)

platziert sind.

Im- und exportiert werden kann das *DXF-Format* (`$ v.in.dxf`). Allerdings gibt es manchmal Probleme mit dem DXF-Austausch, da verschiedene Formatversionen existieren.
Das zweite Modul `$ v.in.dxf3d` arbeitet bei Linienvektoren korrekt, bei Polygonen allerdings (derzeit) nicht. Die Lösung besteht im Anwenden von `$ v.line2area` nach dem DXF-Import. Anschließend folgt `$ v.alabel`, das mit laufender Nummer die Vektoren „labelt“. Die Attribute können nachher mit `$ v.support` zugefügt werden.
Ein weiteres Vektor-Austauschformat ist das *U.S. DLG-Format*, das in verschiedenen Versionen vorliegt. In Europa ist es eher unüblich.
Ganz wichtig ist generell der Aufruf von `$ v.support` nach einer Importierung, Bearbeitung etc. von Vektordaten. Damit werden die interne Struktur überarbeitet (Aufbau der Topologie) und Probleme bei der Vektordatenbearbeitung vermieden. Vereinfacht kann v.support per Kommandozeile aufgerufen werden:

```
$ v.support map=vektorkarte option=build
```

7.5 Zoomen in Vektorkarten

In Vektorkarten können Ausschnitte entweder über das Zoomen von Rasterdateien vergrößert werden oder direkt über die Modifikation der Koordinaten in `$ g.region` (Menüpunkt 1). Danach ist unbedingt `$ d.erase` aufzurufen, um die Änderungen auch umzusetzen. Die anschließende Kartenneuausgabe mit

```
$ d.vect vektordatei
```

zeigt dann den gewünschten Ausschnitt.
Ein komfortables, mausgesteuertes Zoommodul ist `$ d.zoom`.
Alternativ kann die Karte in `$ v.digit` geladen und dort betrachtet werden. Dieses Modul beinhaltet umfangreiche Zoom- und Panning-Funktionen (Verschieben des Bildmittelpunktes bei gleichbleibender Zoomtiefe).

7.6 Vektordaten-Umwandlung zu Rasterdaten

Um eine Umwandlung vom Vektor- in das Rasterformat durchzuführen, müssen die Vektoren zuerst ein Label, also eine Wertzuweisung bzw. ein Attribut erhalten. Dafür können, sofern es nicht auf genau festgelegte Werte ankommt, alle Vektoren automatisch mit derselben Information belegt werden. Im Modul

```
$ v.digit
```

besteht hierfür die Möglichkeit. Zuerst wird das Digitalisiermedium (Digitalisierbrett oder Maus) angegeben, dann der Name der zu bearbeitenden Vektorkarte. Dann sehen Sie eine Parameterliste

für die Karte, in der vor allem der Maßstab korrekt gesetzt sein muss. Nach Verlassen dieser Maske erhalten Sie das v.digit-Menü.
Nun ist der Menüpunkt „L“ wie „Label“ zu wählen, dann „B“ wie „Bulk remaining labels“ (belegt alle unattributierten Vektoren mit einem Wert). So werden alle Vektoren auf einen definierbaren Einheitswert gesetzt. Sie verändern ihre Farbe im GRASS-Monitor (zur optischen Kontrolle).

Ein Problem kann in v.digit auftreten – manchmal lassen sich nicht alle Vektoren automatisch „labeln“ (erkennbar an der Uneinheitlichkeit der Farben). Dann hilft folgender Trick: Man digitalisiert mit der Maus einen kleinen Linienvektor an eine freie Stelle, führt die Prozedur „Label“ – „Bulk remaining labels“ (diesmal erfolgreich) durch und löscht die soeben neu angelegte Linie wieder.
Nun ist `$ v.support` zu benutzen, um die Vektortopologie neu aufzubauen (Option: „build“).
Die Umwandlung der Vektoren in Rasterdaten erfolgt anschließend mit `$ v.to.rast`, das Modul ist menügesteuert. Es werden nur Vektoren mit Attribut konvertiert. Anschließend sollte `$ r.support` auf die neue Rasterkarte angewendet werden („Edit header“: no, „update stats...“: yes, weitere Fragen: „return“).

7.7 Umwandlung von Vektorhöhenlinien in ein Rasterhöhenmodell

Digitalisierte Höhenlinien (Linienvektoren) lassen sich auf einfache Weise in ein Rasterhöhenmodell umwandeln. Zunächst müssen die Vektoren, die die Höheninformation als Label tragen, in das Rasterformat umgewandelt werden. Die Höheninformationen können dazu in `$ v.digit` unter „L“ wie „Label“ in wenigen Durchgängen mit „c“ wie „Label contours“ gesetzt werden (Intervall mit „i“ setzen). Es bestehen ab GRASS 5.0.x zwei Möglichkeiten, aus Vektorhöhenlinien in ein Rasterhöhenmodell zu erzeugen:

- Umwandlung der Vektorhöhenlinien in Rasterhöhenlinien, dann Interpolation (Methode: IDW (*inverted distance weighted*))
- Direkte Interpolation von Vektorhöhenlinien in Rasterhöhenfläche (Methode: RST (*regularized splines with tension*))

Die Wahl der geeigneten Interpolationsmethode ist von der Aufgabenstellung abhängig.

Variante 1: IDW-Methode

Vor der Umwandlung der Vektorhöhenlinien in Rasterhöhenlinien ist die gewünschte Rasterauflösung vorher mit `$ g.region`, Menüpunkt 1 (GRID RESOLUTION), einzustellen. Dann folgt die Umwandlung in das Rasterformat mit:

```
$ v.to.rast
```

Anschließend sollte wie üblich `$ r.support` auf die neue Rasterkarte angewendet werden („Edit header“: no, „update stats...“: yes, weitere Fragen: „return“). Die Rasterdatei enthält nun die

gerasterten Linien ohne Zwischenwerte. Um eine geschlossene Oberfläche zu erzeugen, müssen also eben diese Zwischenwerte interpoliert werden. Dazu dient das Modul:

```
$ r.surf.contour
```

Als erste Angabe wird die gerasterte Liniendatei angegeben, anschließend ein Name für das zu berechnende Höhenmodell. Nach einiger Zeit liegt das vollständige Rastermodell vor. Hinweis: Punkthöhen sollten sich nur in Form von Bergspitzen oder Tiefpunkten bei Senken in der Karte vorhanden sein, also innerhalb einer geschlossenen Höhenlinie. Punkte zwischen Höhenlinien können Probleme verursachen.

Variante 2: RST-Methode

Die Spline-Interpolation führt häufig zu besseren Interpolationsergebnissen. Hier wird die Vektorkarte direkt angegeben, intern werden die Linien in das Punktformat umgewandelt und interpoliert (derselbe Algorithmus wie s.surf.rst). Das Modul wird folgendermaßen aufgerufen:

```
$ v.surf.rst
```

Nun ist die Vektorkarte mit den Isolinien anzugeben. Die Frage „Use category data instead of attribute?“ ist üblicherweise mit „no“ zu beantworten. Dann wird angegeben, ob Vektorlinien mit „0“-Attribut gültig sind oder „no data“ entsprechen sollen (typisch: „no“). Die „Maximum distance between points“ kann vom Programm ermittelt werden („return“), ebenso „Minimum distance between points“. Nun lässt sich optional eine Punktdatei erzeugen, die die lokalen Abweichungen der Interpolation an den Stützstellen zeigt (wenn nicht: „return“). Dann wird die zu erzeugende Rasterkarte benannt: „Output z-file (elevation)“. Optional können statt der Höhenwerte auch die Partialableitungen berechnet werden. Nun werden die optional berechenbaren Hangneigungs- und Expositionskarten sowie Profilkrümmungskarten abgefragt (wenn nicht: „return“). Es folgt die optionale Maskenkarte (alternativ zu r.mask). Dann kann ein optionaler Multiplikator für die Werte der Vektorlinien angegeben werden. Anschließend folgen die Einstellungen zur Spline-Interpolation: „Tension“ steuert die Behandlung von Extremwerten, „Smoothing“ die Übergänge bei hoher Reliefenergie. Die weiteren Parameter steuern die Segmentierung (in den seltensten Fällen kann die Interpolation für das Projektgebiet in einem Schritt erfolgen), es kann eine entsprechende Segmentierungskarte sowie eine Karte der Segmentüberlagerung im Vektorformat erzeugt werden (wenn nicht: „return“). Die Anwort auf die Frage für den Normalisierungsparameter „Use dnorm independent tension?“ ist von den Eingangsdaten abhängig. Zeigen sich beispielsweise „Krater“ in der erzeugten Rasteroberfläche, wurde die Spannung der Splines (tension) falsch gewählt bzw. der Parameter „-t“ (dnorm) fälschlicherweise benutzt.

Es ist empfehlenswert, die entsprechenden Aufsätze von MITASOVA ET AL. (1993a/b, 1999) zu lesen, um die Spline-Interpolation richtig zu steuern.

7.8 Abfragen von Vektorinformationen

Graphisch können Vektorinformationen sehr einfach mit der Maus im GRASS-Monitor abgefragt werden. Mit `$ d.vect` lassen sich Vektordaten anzeigen, mit `$ v.area` per Maus Flächenausdehnungen abfragen. Optional wird die jeweils gewählte Fläche farbig markiert. Soll statt der Flächeninformation das individuelle Attribut des Vektors ermittelt werden, ist stattdessen `$ d.what.vect` oder `$ v.what` zu benutzen. Mit `$ d.vect.area` können Sie sich Flächen von einer oder mehreren Kategorien anzeigen lassen. Es erweist sich als praktisch, immer den Kartennamen als Parameter beim Modulaufruf anzugeben.

7.9 Verschneiden von Flächen, Vektorextraktion

Eine Überlagerung von zwei oder mehreren Vektorkarten kann mit dem Modul `$ v.patch` erreicht werden. Im Ergebnis tauchen dann alle Geometrien auf, die in den Karten enthalten sind.

Mit dem Modul `$ v.cutter` können dagegen Teilflächen ausgeschnitten werden. So lässt sich beispielsweise mit `$ v.digit` eine Fläche mit der Kategorie 1 digitalisieren, die als Schablone für die eigentliche Vektorkarte dient. Das Modul `$ v.cutter` produziert eine neue Karte, in der nur noch Vektordaten enthalten sind, die in der Schablonenfläche liegen. Die Abbildung 22 zeigt die Unterschiede zwischen v.patch und v.cutter.
Sollen dagegen einzelne Informationen aus einer Vektorkarte in eine neue Karte extrahiert werden, erstellt `$ v.extract` unter Angabe der gewünschten Kategorie(n) diese Selektionskarte.

Probleme können sich generell im GIS beim Verschneiden ergeben, wenn Polygone, die an sich übereinander liegen sollten, teilweise in den Grenzlinien geringfügig (durch Digitalisierfehler) voneinander abweichen. Dann ergeben sich Kleinstflächen („sliver polygons“), die das Verschneidungs-

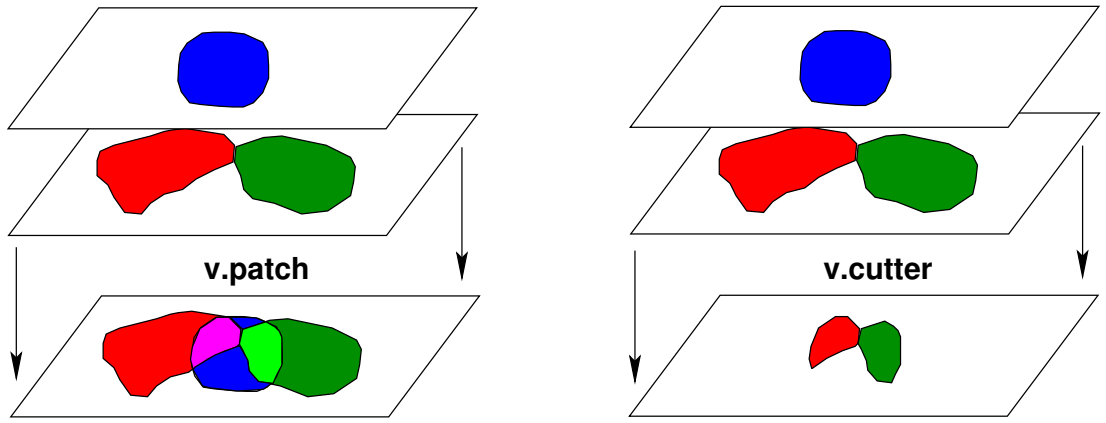


Abbildung 22: Auswirkungen der Verschneidung von Vektorkarten mit v.patch bzw. v.cutter

ergebnis verschlechtern oder sogar unbrauchbar machen können. Entsprechend korrekt sollte also digitalisiert werden.

7.10 Interpolation von Rasteroberflächen aus Vektordaten

GRASS bietet Ihnen unterschiedliche Möglichkeiten, Rasteroberflächen aus Vektorkarten zu gewinnen. Um beispielsweise eine Höhenrasterkarte aus einer Karte mit Vektorhöhenlinien zu erzeugen, gibt es zwei Möglichkeiten in GRASS. Sie können

- direkt ein Höhenmodell aus den Vektordaten interpolieren oder
- die Vektordaten erst in Rasterlinien umwandeln und dann im Rasterformat interpolieren.

Die Wahl hängt davon ab, welchen Interpolationsalgorithmus Sie verwenden möchten. Ab GRASS 5.0.x können Sie mit „regularised splines with tension“ direkt aus Vektorlinien eine Rasteroberfläche interpolieren:

```
$ v.surf.rst
```

Da die Spline-Interpolation von einigen Parametern abhängig ist, sollten Sie MITAS ET AL. (1999), MITASOVA ET AL. (1993a) und MITASOVA ET AL. (1993b) sowie die Anleitung zum Modul lesen. Die Spline-Interpolation liefert bei korrekter Steuerung des Moduls hervorragende Ergebnisse.
Alternativ können Sie die Vektorlinien auch direkt in das Rasterformat umwandeln (vgl. nächsten Abschnitt) und dann mit dem Raster-Interpolationsmodul

```
$ r.surf.idw2
```

die flächenhafte Interpolation durchführen. Genaues finden Sie oben im Abschnitt 6.13.

7.11 Direkte Vektordaten-Wandlung zu Rasterdaten

Eine direkte Konvertierung von Vektordaten in das Rasterformat kann für eine Weiterverarbeitung (Verschneidungen im Rastermodell, Kartenalgebra) interessant sein.
Um eine Umwandlung vom Vektor- in das Rasterformat durchführen zu können, müssen die Vektoren jeweils ein Attribut (Label) besitzen. Das ist der Regelfall.
Sind nur die Flächenausdehnungen für die spätere Rasterverarbeitung relevant, nicht aber die Attribute, können alle Vektoren auch automatisiert mit der selben Information belegt werden. Im Modul

```
$ v.digit
```

besteht hierfür die Möglichkeit. Der Menüpunkt „L“ wie „Label“ ist zu wählen, dann „B“ wie „Bulk remaining labels“. So werden alle Vektoren auf einen definierbaren Einheitswert gesetzt. Sie verändern ihre Farbe im GRASS-Monitor (zur optischen Kontrolle).

Ein Problem kann auftreten - manchmal lassen sich nicht alle Vektoren automatisch „labeln“ (erkennbar an der Uneinheitlichkeit der Farben). Dann hilft folgender Trick: Man digitalisiert mit

der Maus einen kleinen Linienvektor an eine freie Stelle, führt die Prozedur „Label“ – „Bulk remaining labels“ (diesmal erfolgreich) durch und löscht die soeben neu angelegte Linie wieder. Nach Verlassen von v.digit ist

```
$ v.support
```

anzuwenden (Option: „build“), um die Topologie aufzubauen.

Die Umwandlung in das Rasterformat erfolgt anschließend mit `$ v.to.rast`, das Modul ist menügesteuert. Das Modul konvertiert nur Vektoren mit Attribut. Da die erzielte Genauigkeit von der Rasterzellenauflösung abhängt, können Sie diese vorher mit

```
$ g.region
```

passend wählen, sofern Sie nicht an eine bestimmte Auflösung gebunden sind. Anschließend sollten Sie `$ r.support` auf die neue erzeugten Rasterdaten anwenden und die Datenstatistik berechnen.

Eine Besonderheit ergibt sich, wenn Sie Flächen vorliegen haben, aber im Vektor- oder Rasterformat nur noch die Grenzlinien bekommen wollen. Hier ist erforderlich, den Vektortyp zu ändern. In GRASS kann die Umwandlung ohne großen Aufwand durchgeführt werden. Zuerst ist die Vektorflächenkarte in das GRASS-ASCII-Vektorformat zu exportieren (normalerweise sind Vektoren im GRASS-Binär-Vektorformat gespeichert):

```
$ v.out.ascii
```

Die Attribute bleiben dabei erhalten, die Exportdatei wird in `$LOCATION/dig_ascii/` gespeichert. Nun benötigen Sie einen Textedit mit Austauschfunktion („Suchen“, „Ersetzen“) wie z.B. textedit, joe oder einen Office-Editor. In der Datei sind alle Vektoren mit Attribut und Typ anhand ihrer Stützpunkte aufgelistet. Ersetzen Sie alle „A“ durch „L“ (großgeschrieben, kleingeschriebene Typbuchstaben markieren einen Vektor als gelöscht). Damit haben Sie den Vektortyp geändert. Mit

```
$ v.in.ascii
```

wird die geänderte Karte wieder in das GRASS-Binär-Vektorformat importiert. Anschließend ist

```
$ v.support
```

anzuwenden (Option: „build“), um die Topologie aufzubauen. Um hier keine Warnungen über nicht zuweisbare Attribute zu bekommen, können Sie vorher, sofern nicht benötigt, die Attributtabelle in `$LOCATION/dig_att/` löschen. Die Attribute lassen sich deshalb nicht korrekt zuweisen, da bei Flächen die attributtragenden Labelpunkte in der Fläche liegen, bei Linienvektoren aber direkt an die Linien gekoppelt sein müssen. Und diese Struktur liegt ja nicht vor. Nun haben Sie eine Karte, die die Flächengrenzen beschreibt.

Wollen Sie diese Grenzen im Rasterformat vorliegen haben, ist in v.digit (s.o.) ein Einheitsattribut vor der Umwandlung mit v.to.rast zu vergeben. Sie können beispielsweise, wenn Sie vor oder nach `$ v.in.ascii` die Attributtabelle gelöscht haben, das Einheitsattribut „1“ vergeben, damit alle Rasterlinien später diesen Wert haben. Die Flächeninhalte sind entprechend „0“, also nicht ausgefüllt.

7.12 Automatische Vektorisierung von Rasterdaten

Häufig ist auch genau der andere Weg wichtig: aus Rasterdaten (Linien oder Flächen) sollen Vektordaten erstellt werden. Das kann einerseits manuell mit dem Modul v.digit erfolgen, jedoch gibt es eine automatisierte Alternative. GRASS bietet zwei interessante Module für die automatische Vektorisierung:

- für Linienvektoren (lines): `$ r.line`
- für Isolinienvektoren (lines): `$ r.contour`
- für Vektorflächen (Polygone bzw. areas): `$ r.poly`

Linien und Flächen sind getrennt zu behandeln.

Um eine Linienkarte zu vektorisieren, müssen die Linien in der Rasterdatei „verdünnt“ werden. Das geschieht mit dem Modul:
`$ r.thin`
Anschließend wandelt
`$ r.line`
alle Linien in der Rasterdatei in eine neue Vektordatei um.

Flächen können direkt vektorisiert werden. Das Modul
`$ r.poly`
sorgt für die entsprechende Umwandlung. Die Vektorattribute werden übertragen.
In beiden Fällen ist anschließend `$ v.support` auf die neue Vektordatei anzuwenden. Zum Setzen oder Überprüfen der Label kann `$ v.digit` herangezogen werden.

Eine geographisch interessante Anwendung ist die automatische Digitalisierung von Höhenlinien. Die Erzeugung der Vektorlinien aus einem Höhenmodell erfolgt mit
`$ r.contour`
Sie brauchen miminal nur die Linienabstände in Höhenmetern anzugeben (z.B. 10m-Intervalle), alle anderen Werte werden automatisch ermittelt.

Ein Hinweis für gescannte topographische Karten mit „bereinigter“ Farbtabelle (weniger als 20 Farben): Mit dem Modul
`$ r.reclass`
können alle Farbinformationen bis auf die Farbe der Höhenlinien ausgeblendet werden (dazu muss bei gescannten Karten die Farbtabelle mit Bildbearbeitungssoftware wie xv aufbereitet sein). Danach kann die Selektion über die Klassifizierung erfolgen. Eine Farbtabelle mit beispielhaften Werten für digitale Karten befindet sich im Anhang A.8.2.

Es ist auch möglich, die äußere Grenze von Punkten (Umhüllungskurve, *convex hull*) automatisch als Vektorgrenzen zu erzeugen. Dazu wird eine Rasterkarte, die nur einzelne Punkte enthält, in eine Vektorpunktkarte umgewandelt. Dieses Verfahren ist nur über eine Zwischenumwandlung in das Punktformat möglich:

```
$ r.to.sites -a input=rasterkarte output=punktkarte
```

Diese Punktkarte kann nun in Vektorpunkte umgewandelt werden mit:

```
$ s.to.vect input=punktkarte output=vektorkarte
```

Der umschließende Vektorlinienzug wird mit

```
$ v.geom in=vektorkarte output=vektorgrenze operation=hull
```

erstellt. So können Sie beispielsweise mit GPS punktweise eingemessene Ackerschlaggrenzen in eine Vektorfläche konvertieren.

8 Punktdatenverarbeitung

In Umweltstudien stehen häufig nur punktuelle Daten einzelner Meßstationen zur Verfügung wie beispielsweise klimatische Messungen, einzelne Höhenpunkte in einem Gelände oder Bohrlochinformationen. GRASS bietet Module zur geostatistischen Analyse dieser punkthaften Daten an sowie Konvertierungsroutinen zur Übertragung der punkthaften Geodaten in das Vektor- bzw. Rasterformat. Bei der Umwandlung in das Rasterformat sind auch verschiedene Interpolationsmethoden zur Oberflächengenerierung vorhanden. So lassen sich beispielsweise aus einzelnen Höhenpunkten digitale Höhenmodelle (DGM) interpolieren. Als Nebenprodukte können bei Höhendaten auch Hangneigungen, Exposition und Profilkrümmungen der Hänge mitberechnet werden.
Punkthafte Daten werden in GRASS als „Sites“ bezeichnet. Sie liegen entweder als Punkte im eigenen „Sitesformat“ (z.B. nach einem Import aus einer Tabelle) vor oder im „Vektorformat“ nach Digitalisierung von einer Karte.

8.1 Manuelles Setzen von Punkten

Um punkthafte Daten manuell mit jeweiligem Attribut zu digitalisieren, können Sie das Modul `$ v.digit` benutzen. Es entsteht eine Vektorkarte mit Punktinformationen, die später mit `$ v.to.sites` in das „Sitesformat“ überführt wird. Geben Sie zu Beginn der Digitalisierung in v.digit einen neuen Dateinamen an und korrigieren Sie im nachfolgenden Startformular den Wert für „map scale“ vom Verhältnis 1:0 auf den entsprechenden Kartenmaßstab. Da im Allgemeinen von einer Karte digitalisiert wird, kann eine Vektor- oder Rasterkarte in den Hintergrund gelegt werden („C“ - Customize, „O“ - Vektorkarte hinterlegen bzw. „B“ - Rasterkarte hinterlegen).

Mit „t“ wird der Vektortyp auf „site“ verändert, mit „D“ das Digitalisieren einzelner Punkte (Sites) begonnen. Im zweiten Schritt werden mit „L“ die Label vergeben, also beispielsweise beim Digitalisieren von Messstationen die Niederschlagshöhen an jeder Station. Dazu wird ein Wert als „Category“ angegeben und die zugehörige Station angeklickt. Die Sites-Kreuze wechseln nach dem Labeln ihre Farbe.
Die digitalisierten Sites sind, wie bereits angemerkt, allerdings noch als Vektorpunkte gespeichert. Sie müssen nun in „echte“ Sites konvertiert werden. Dazu eignet sich das Modul:

```
$ v.to.sites
```

Es wird mit der erstellten Vektordatei aufgerufen:

```
$ v.to.sites input=vektordatei output=sitesdatei
```

Dieser Weg der Punktdigitalisierung ist eher selten, häufiger werden Punktdaten aus Dateien importiert.

8.2 Bearbeitung digitaler Höhenmodelle

Für viele Anwendungen im GIS-Bereich, insbesondere bei Simulationen (Kaltlufthaushalt, Erosion etc.) werden digitale, rasterorientierte Höhenmodelle benötigt. In diesem Abschnitt soll der Aufbau eines entsprechenden Modells vorgestellt werden.

Benötigt wird eine Datei, in der die Höheninformationen lückenlos in der gewünschten Rasterweite vorhanden sind. Sind Lücken vorhanden, können fehlende Werte in GRASS nach dem Import der Daten mit

`$ s.surf.idw` und `$ s.surf.rst`

berechnet werden.

8.2.1 Import und Konvertierung von Höhendaten im xyz-ASCII-Format

Sind die Höhendaten in einer externen Datenbank enthalten (z.B. in dBase oder Excel), lässt sich dort folgende Anordnung für eine Datei, die dann in GRASS importiert werden soll, erstellen:

```
Rechtswert  Hochwert  Höhe
```

Beispielsweise (12.5m-Raster):

```
3246000    5877000    23.0
3246012.5  5877012.5  24.4
3246025    5877025    22.3
```

Diese Höhendatei kann direkt mit `$ s.in.ascii` eingelesen werden („Attribute field to import represents elevation heights“: yes). Das Modul fragt den Namen der neu zu erstellenden Höhendatei und den Namen der Quelldatei (ASCII) ab, die im aktuellen Verzeichnis liegen muss. In GRASS 4.x ist das Modul s.in.ascii.dem zu benutzen.

Es bestehen verschiedene Möglichkeiten, Punktdaten in das Raster- oder Vektorformat umzuwandeln. Eine Übersicht zeigt Abbildung 23. Die Möglichkeiten werden im Folgenden vorgestellt.

(a) Die direkte Umwandlung der Punktdaten in das Rasterformat **ohne Interpolation** erfolgt mit `$ s.to.rast`, bei der die Namen für die soeben importierte Sitesdatei und die zu erstellende Rasterdatei anzugeben sind.

(b) Müssen fehlende Höhenwerte interpoliert werden, so kann, wie im folgenden Abschnitt beschrieben, verfahren werden.

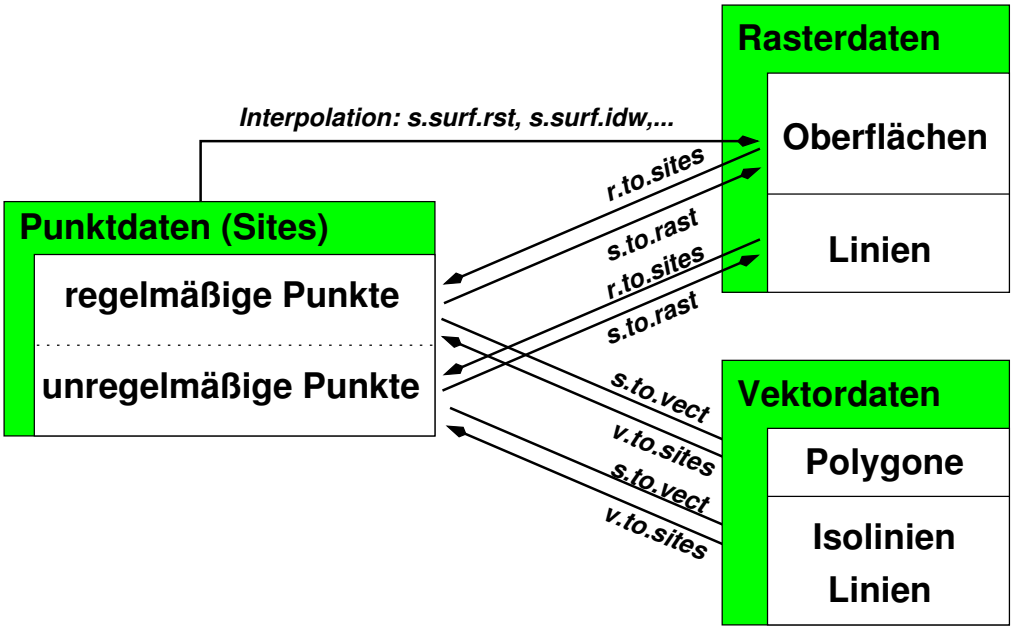


Abbildung 23: Wege zur Umwandlung von Punktdaten in Vektor- oder Rasterdaten

8.2.2 Interpolation von Höhenmodellen

Sogar aus einzeln gesetzten Höheninformationen lässt sich in GRASS ein Rasterhöhenmodell interpolieren. Dazu importiert man über `$ s.in.ascii` die einzelnen Höhenpunkte. Die Anordnung in der benötigten ASCII-Datei sieht auch hier folgendermaßen aus:

`Eastcoordinate Northcoordinate Category` (getrennt durch Leerzeichen), also:

`Rechtswert Hochwert Höhe`

oder digitalisiert wie im vorigen Abschnitt beschrieben, manuell einzelne Punkte.

Aus diesen Höheninformationen errechnet GRASS unter Angabe der Anzahl der gesetzten Punkte (Sites) eine vollständige Rasterkarte. Die gewünschte Auflösung wird mit `$ g.region`, Menüpunkt 1 eingestellt (GRID RESOLUTION).

Eine Interpolation kann mit mehreren Methoden durchgeführt werden. Die zu wählende Methode hängt von den Eingangsdaten ab. Sofern nicht anders vermerkt, gelten die Verfahren für unregelmäßig verteilte Höhenpunkte sowie regelmäßige Gitter:

- IDW (inverse distance weighted): entfernungsgewichtete Berücksichtigung vorhandener Punkte
- Splines (regularized splines with tension): Berechnung fehlender Punkte anhand von Splines, die durch die vorhandenen Punkte gelegt werden
- Bilinear: Gradientenberechnung aus vier umgebenden Punkten, nur für regelmäßige Punktgitter

- Kubische Faltung (cubic convolution): Berechnung aus acht umgebenden Punkten, nur für regelmäßige Punktgitter

- Nächster Nachbar (nearest neighbor): wie IDW

Zur Verwendung der IDW-Methode wird das Modul `$ s.surf.idw` aufgerufen. Die erste Angabe ist die Sitesdatei, die die einzelnen Höhenpunkte enthält. Anschließend benennen Sie die zu erzeugende Rasterdatei. Optional kann die Anzahl der zu berücksichtigen nächsten Nachbarn angegeben werden, standardmäßig sind es zwölf Nachbarpunkte.

Die RST-Methode ist in `$ s.surf.rst` implementiert. Da die Verwendung von Splines einerseits sehr gute Ergebnisse liefert, andererseits die Steuerung des Moduls aber auch recht komplex ist, sollten Sie unbedingt die Modulbeschreibung lesen (online oder mit g.manual, vgl. auch Abschnitt 7.7)

Die Methoden „bilineare Interpolation“ und „kubische Faltung“ sind im Punktformat nicht direkt durchführbar, die Punktdaten müssen vorher mit `$ s.to.rast` in Rasterpunkte umgewandelt werden. Vorher ist die Punktweite als identische Rasterweite mit `$ g.region` einzustellen. Dann werden die regelmäßig verteilten Punkte in eine Rasterkarte gleicher Auflösung umgewandelt. Nun ist die neue Rasterauflösung wieder mit `$ g.region` einzustellen. Anschließend erfolgt die Interpolation auf die höhere Auflösung mit `$ r.bilinear`. Eine kubische Faltung kann mit `$ r.mapcalc` programmiert werden.
Der Interpolationsvorgang nimmt je nach Auflösung, Rechnergeschwindigkeit und gewählter Methode einige Zeit in Anspruch. In NETELER & MITASOVA 2002 werden die verschiedenen Interpolationsmethoden ausführlich und mit theoretischem Hintergrund beleuchtet.
Zur besseren Anschauung kann das erzeugte Modell mit dem Modul `$ d.3d` oder `$ nviz` dreidimensional betrachtet werden (dafür die soeben erzeugte Rasterdatei als „rasterfile to be displayed (color)“ sowie auch als „rasterfile for elevation“ angeben). Näheres zu diesem Thema steht im nächsten Abschnitt. Weitere Umwandlungen, nämlich die Erzeugung von Vektorhöhenlinien, finden Sie im Abschnitt 6.14, die Erzeugung eines Höhenmodells aus Vektorhöhenlinien im Abschnitt 7.7.

8.2.3 Export von Höhendaten

Höhendaten können aus dem Rasterformat oder aus dem Sites-Format exportiert werden. Im Rasterformat kann man mit dem folgenden Befehl erreichen, dass zu jeder Zelleninformation die Koordinaten ausgegeben werden (xyz-Format):

```
$ r.stats -1 -g input=rasterdatei > hoehendaten.txt
```

Die Struktur der ASCII-Datei hoehendaten.txt ist:
`Rechtswert Hochwert Zellenwert`
Diese Datei könnte dann über `$ s.in.ascii` wieder importiert werden.

Wollen Sie Daten aus dem Sites-Format exportieren, kommt das GRASS-Modul `$ s.out.ascii` zur Anwendung. Alternativ können die Punktdaten natürlich auch in das Rasterformat konvertiert und beispielsweise mit `$ r.out.arc` in das ARC-GRID-Format exportiert werden.

8.2.4 Dreidimensionale Betrachtung

Das Darstellungsmodul `$ d.3d` erlaubt die Umrechnung von Rasterdaten (v.a. Höhendaten) in eine 2.5-dimensionale Landschaft. Es besteht die Möglichkeit, sowohl Drahtmodelle in beliebigen Blickrichtungen als auch Überlagerungen mit gescannten Karten bzw. Schummerungen (diese werden in Expositionsrechnungen mit `$ r.slope.aspect` erzeugt) zu erstellen. Hinweis: d.3d ist ab GRASS 5.0.x von „nviz“ abgelöst, das multiple Raster-, Vektor- und Punktdaten verarbeiten und darstellen kann (vgl. Abschnitt 10.4).
Die erste Angabe ist die Rasterdatei, die dem Modell überlagert werden soll (z.B. eine gescannte topographische Karte, ein Satellitenbild oder eine Schummerungs-Rasterdatei). Als zweites wird die Rasterdatei mit Höhendaten angegeben.
Ein Beispiel ist in Abbildung 24 dargestellt (Schummerung wurde überlagert). Das Programmformular von `$ d.3d` wurde dafür auf diese Weise ausgefüllt:

```
-------------------------------------------------------------------------------
          VIEWING REGION        | RUN? Y/N            Y_
          N: 5678               | Erase Color         black___ Hintergrundfarbe
 W: 3566  ---|--- E:3677        | Vertical Exaggerat. 2______  Überhöhung
          S: 5566               | Field of View (deg) 30.00__  Blickwinkel
                                | Lines Only? Y/N     n_       n, wenn Karte etc.
                                                               über Modell gelegt
                                                               werden soll
       VIEW COORDINATES:        |Line Color           color__  Grid un-/sichtbar
Eye Position    Center of view  |Line Frequency       1____    Grid-Dichte
 ...50<- Northing (y) -> ...00  |Resolution           10.00__  Auflösung
 ...50<- Easting  (x) -> ...00  |Plot zero elev?      N_       Nullwerte darst.
 ...26<- Height   (z) -> ...00  |Box color            none___  3D-Box Farbe
                                |Average elevs?       N_       Mittelwerte
-------------------------------------------------------------------------------
Eye -----                       | Colors: red orange yellow green blue
                      N         |   indigo violet brown gray white black
        /MAP----------/         |
       /      X      /          | Special 'colors':
     W/_____________/E          |   'None' available for 'Erase Color'
           S                    |   'color' available for 'Line Color'
-------------------------------------------------------------------------------
```

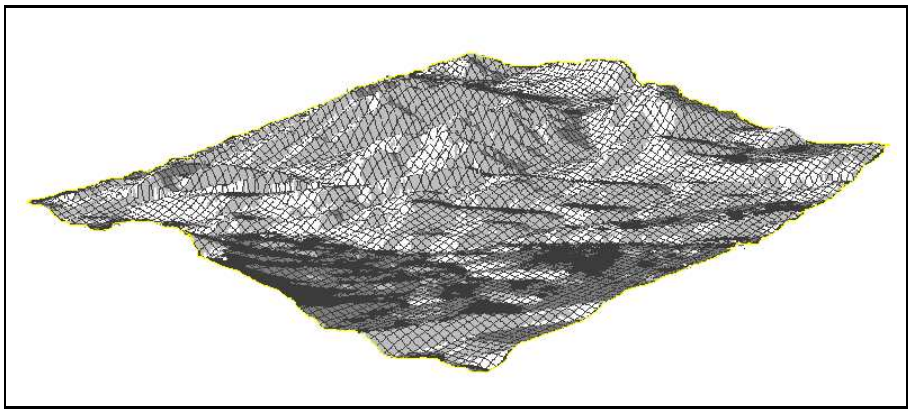


Abbildung 24: Dreidimensionale Ansicht eines Höhenmodells mit Schummerung

Einige Anmerkungen dazu: „Lines Only“ gibt an, ob nur ein Drahtmodell oder eine geschlossene Oberfläche gezeigt werden soll. Wenn „Lines Color“ auf „color“ steht, wird kein Gitterraster überlagert. Soll ein Gitter überlagert werden, ist hier eine Farbe anzugeben, die Gitterweite kann mit „Line Frequency“ gewählt werden. Die Augenposition ist in der Kommandozeilenversion von `$ d.3d` recht schwer einzustellen. In TclTkGRASS lassen sich dagegen sowohl die Position des Betrachters (*Eye Position*) als auch das Blickziel (*Center of View*) mit der Maus aus der Karte selektieren.

Die manuelle Bedienung von d.3d mag etwas antiquiert erscheinen (dafür gibt es nun NVIZ als Alternative), der große Vorteil ist aber die Scriptsteuerbarkeit. Sie können damit beispielsweise bei der Verwendung von GRASS als „Online-GIS“ räumliche Ansichten produzieren, die dann automatisch im Webbrowser Ihrer Server-Besucher erscheinen. Lesen Sie bitte dazu auch Abschnitt 9.3 über die Benutzung des „CELL-Treibers“.

8.3 Berechnung von Thiessen-Polygonen

Als weiteres Beispiel für eine Interpolationsanwendung soll ein hydrologisches Thema folgen: Dort wird zur Untersuchung der Grundwasserneubildung der Gebietsniederschlag betrachtet. Dabei werden die Messungen einzelner Stationen, die repräsentativ für ein Gebiet stehen, über die Fläche interpoliert, um die Niederschläge der Teilgebiete zu bestimmen. Das hierfür eingesetzte Rechenverfahren nach Thiessen, die sogenannten „Thiessen-Polygone“, lässt sich auch in GRASS durchführen. Dabei werden zur Gebietsabgrenzung die Mittelsenkrechten der Verbindungsgeraden zwischen den einzelnen Stationen errichtet. Die umgrenzten Flächen beinhalten also jeweils eine Messstation.

Im Abschnitt 8.1 „Manuelles Setzen von Punkten“ ist ein Weg zum Digitalisieren dieser Messstationen mit der Niederschlagshöhe als Attribut (v.digit-Variante) beschrieben.

Um möglichst glatte Ränder an den Flächengrenzen zu bekommen, können Sie die Auflösung in

```
$ g.region
```

(Menüpunkt 1) vor der Interpolation heraufsetzen. Die Berechnung der Mittelsenkrechten erfolgt über das Sites-Interpolationsmodul mit

```
$ s.surf.idw in=sitesdatei out=thiessen npoints=1
```

Die berechneten Polygone können mit

```
$ d.rast thiessen
$ d.site.labels sitesdatei
```

angesehen werden (der zweite Befehl blendet die Zahlenwerte ein). Sollen die Polygone vektorisiert werden, erfolgt diese Umwandlung mit `$ r.poly` in das Vektorformat.
Alternativ kann das Modul `$ s.voronoi` verwendet werden (vgl. zur Theorie die Erläuterungen in BILL 1996).

8.4 Besonderheiten von GRASS 5.0.x

GRASS 5.0.x besitzt ein neues Punktdatenformat (Sitesformat): Es werden ein oder mehrere Attribute pro Punkt mit beliebig vielen Dimensionen akzeptiert. Neben den Koordinatenangaben gibt es ein Integer-Feld „category“ (mit dem Zeichen # als Präfix), optionale Dezimalattribute (mit % als Präfix) und optionale Zeichenketten (mit `@` als Präfix, bei Leerzeichenverwendung mit `" "` geschachtelt). Getrennt werden die Felder durch Leerzeichen. Der allgemeine Aufbau sieht so aus:

```
easting|northing|[z|[d4|]...][#category_int] [ [@attr_text OR %flt] ... ]
```

Zwischen den Pipezeichen `|` können beliebig viele Dimensionen angegeben werden (also über die üblichen Raumdimensionen hinaus), jedoch erst dann dürfen die optionale Kategorienummer (maximal eine Integerzahl) und weitere optionale Dezimal- bzw. Zeichenkettenattribute (beliebig viele, getrennt durch Leerzeichen) folgen.

Da `$ s.in.ascii` nur das simple „xyz-Format“ akzeptiert, empfiehlt es sich, die Punktdatei mit einem Texteditor direkt anzulegen oder eine Tabelle im DBF-Format mit `$ s.in.dbf` einzulesen. Beim Erzeugen einer Punktdatei mit einem Texteditor ist die Datei in der GRASS-Datenbank im Pfad `$LOCATION/site_lists/dateiname` zu speichern (gegebenenfalls Verzeichnis `$LOCATION/site_lists/` vorher erzeugen). Dann können auch beliebig viele Dimensionen und Attribute gespeichert werden.

Ein Beispiel: Ein Bodenstandort (Nr. 1) mit Koordinaten, Bodentyp und pH-Wert wird folgendermaßen gespeichert (statt `$ s.in.ascii` zu benutzen):

```
3567000|5787000|#1 %4.2 @"Sandboden"
```

Diese Textdatei wird dann gespeichert, beispielsweise als

```
$LOCATION/site_lists/standortkarte.
```

Ein weiteres Beispiel (Koordinaten und weitere Informationen zu einer U.S.-amerikanischen Stadt):

```
739865.8|4279785.5|#2965 %396685 %10941 %2.473222 @"St. Louis" @MO @city
```

9 Kartenausgabe

Es gibt mehrere Möglichkeiten, Karten im Zusammenhang mit GRASS zu produzieren: GRASS bietet ein entsprechendes Modul (ps.map) an, um Karten im Postscriptformat herzustellen, andererseits können Sie Karten (Raster- und Vektorformat) auch exportieren und, gegebenfalls, in einem externen Programm verarbeiten („GMT“, „xfig“ oder andere Software).
Das GRASS-Modul ps.map arbeitet entweder interaktiv oder scriptgesteuert (zur automatischen Kartenherstellung). Auch wenn ps.map ständig erweitert wird, fehlt doch ein grundsätzliches Element zur einfachen Kartengestaltung : eine graphische Benutzeroberfläche. Wesentlich komfortabler ist jedoch das freie „xfig“-Programm[1]. Mit xfig können Sie aus GRASS exportierte Karten graphisch verarbeiten und ansprechende Karten herstellen, vergleichbar beispielsweise mit dem für MS-Windows erhältlichen Programm „CorelDraw“. Das „xfig“ liegt den meisten Linux-Distributionen bei, läuft aber auf allen UNIX-Plattformen.
Eine interessante Art ps.map mit xfig zu verbinden besteht darin, eine Karte mit ps.map zu erstellen, und sie dann mit dem Programm „pstoedit“[2] in eine xfig-Datei umzuwandeln. Man kann dann alle Vektorelemente der Karte in xfig noch bearbeiten, was nicht geht, wenn man die Karte als Rasterbild exportiert und in xfig wieder importiert.
Es gibt noch zwei weitere Möglichkeiten Karten zu erstellen: der CELL-Treiber und der HTMLMAP-Treiber. Der erste ist dazu gedacht, Karten (mit Raster-, Vektor- und Punktdaten) als hochauflösende Rasterbilder zu exportieren. Der HTMLMAP-Treiber, wie der Name schon andeutet, ermöglicht es, Karten für den Gebrauch in WWW-Seiten zu erstellen, inklusive interaktive Abfrage.

9.1 Kartenausdruck mit ps.map

Die Kartenherstellung als Postscriptdatei lässt sich mit dem Modul `$ ps.map` durchführen. ps.map wird ständig erweitert und hat inzwischen schon eine beindruckende Menge von Optionen, die alle Teile der Kartengestaltung beinhalten. Allerdings muss ps.map über eine Skript-Sprache bedient werden, nicht so einfach wie eine mit der Maus zu bedienende Benutzeroberfläche. Dass hat aber auch, wie so oft, den Vorteil, dass man wiederverwendbare Skripts schreiben kann, was bei der wiederholten Ausgabe der gleichen Art von Karten die Arbeit sehr beschleunigen kann.
Vor dem Aufruf des Moduls ist zuerst ein Format für das später benutzte Ausgabegerät (Drucker) auszuwählen, von denen etliche schon pre-definitert sind. Beispiel:

[1] Im Internet verfügbar: `http://www.xfig.org/`

[2] `http://www.geocities.com/SiliconValley/Network/1958/pstoedit/`, aber sonst auch in den meisten Linux-Distributionen enthalten

`$ ps.select a4` (entspricht Format DIN A4)

Die Definition eigener Druckerformate ist im Anhang A.6 beschrieben. Die Erzeugung der Karte erfolgt dann entweder interaktiv mit:

```
$ ps.map
```

wonach Sie anhand von Menüs durch den Kartenaufbau geführt werden, oder über eine Beschreibungsdatei (Textformat) mit allen benötigten Zuordnungen und Definitionen von Kartenelementen. Dort werden die darzustellenden Raster-, Vektor- und Punktkarten, die Farbzuordnungen, Linienbreiten usw. angegeben. Der Vorteil liegt wie gesagt darin, dass Sie Ihre Definitionen für die nächste Karte auf einfache Weise weiterverwenden können. Ein entsprechender Aufruf lautet:

```
$ ps.map input=beschreibungsdatei output=karte.ps
```

Es ist unmöglich hier alle Optionen von ps.map einzeln zu behandeln. Der Benutzer sollte deshalb die entsprechende html-Seite lesen. Um die Struktur in dieser optionalen Beschreibungsdatei verständlicher zu machen, wird ausserdem im Anhang ein Beispiel (Moordaten.psmap im Abschnitt A.7) vorgestellt. Trotzdem hier einige Hinweise.

Jede Art Daten, ob Raster, Vektor oder Punkt, kann mit ps.map dargestellt und relativ detailliert konfiguriert werden. So kann man zum Beispiel die Linienfarbe und -dicke von Vektorlinien, die Grösse von Punktobjekten, oder die Farbkombinationen von Rasterkarten bestimmen. Aber auch fast alle anderen Zusatzinformationen so wie Titel, Legende, Rahmen, usw., können mit ps.map dargestellt werden.

Nach der Erzeugung der Postscript-Datei kann die erzeugte Karte an den Drucker gesendet werden:

Allgemeiner Aufruf:

```
$ lpr -s -Pdruckername kartenname.ps
```

Beispiel:

```
$ lpr -s -Pclc-a4 karte1.ps
```

Postscriptdateien brauchen oft sehr viel Platz. Deshalb sollten Sie die Option `-s` angeben – sie verhindert, dass der Druckerdämon eine zusätzliche temporäre Kopie auf der Festplatte erstellt.

Im folgenden Abschnitt lernen Sie eine Alternative zu ps.map kennen, die sehr angenehm zu bedienen ist und sogar kartographisch-funktionale Vorteile gegenüber proprietärer Software wie „CorelDraw“ hat.

9.2 Kartengestaltung mit xfig

Das Programm `$ xfig` ist recht einfach zu bedienen. Es handelt sich quasi um ein Zeichenprogramm, das zusätzlich einige für kartographische Belange relevante Funktionen bereithält. Am interessantesten ist der einstellbare Maßstab (in der rechten Maßleistenecke), der beliebig gesetzt werden kann. Dadurch werden Linienlängen usw. beispielsweise gleich (je nach Maßstab) in

Metern angegeben, so dass sich sehr einfach Kartenrahmen und Legenden zeichnen lassen. Dazu klicken Sie mit der rechten Maustaste auf die Maßstabsangabe oben rechts, definieren im nun erscheinenden Fenster „Figure units: user defined“, als „Unit name: cm“, „Figure scale: user defined“ und schließlich als „Scale factor: 50000“ (oder eine andere Maßstabszahl). Nun werden Linienstrecken immer in geographischer Ausdehnung gemäß dem gewählten Maßstab angezeigt.

Generell wird Ihnen die aktuelle Bedeutung der Maustasten oben rechts erklärt. Das Programm unterstützt drei Maustasten. Zeichenelemente lassen sich gruppieren: Beispielsweise werden aus GRASS importierte Vektoren (dort mit v.out.xfig exportieren) immer in einer Objektgruppe zusammengefasst, um alle Vektoren gleichermaßen verschieben zu können.

Eine Rasterkarte aus GRASS lässt sich in `$ xfig` integrieren, indem sie im TIFF-Format aus GRASS exportiert und mit `$ xv` in das PNG-Format umgewandelt wird. In xfig gibt es ein „Picture“-Element in der Menüleiste, um die Karte mit der Maus zu platzieren. Für die Herstellung hochauflösender Karten für diesen Import sollte der CELL-Treiber in GRASS verwendet werden (vgl. Abschnitt 9.3).

Für die direkte Verarbeitung von Vektordaten kann das Script `$ v.out.xfig` (neu ab GRASS 4.2.1) benutzt werden. Die Vektorlinien, -flächen und -punkte werden hier in das xfig-eigene ASCII-Vektorformat exportiert. Punkte im Sites-Format sollten Sie vorher in das GRASS-Vektorformat umwandeln (s.to.vect) und mit der Vektorkarte überlagern (v.patch).

Eine andere Art, Karten in `$ xfig` zu importieren, besteht darin, die Karte mit ps.map zu erstellen und sie dann mit `$ pstoedit` in das xfig-Format umzuwandeln. Dabei ist es generell besser, jede einzelne Informationsebene einzeln mit `$ ps.map / $ pstoedit` zu exportieren, da diese dann in `$ xfig` einfacher getrennt zu bearbeiten sind. Man muss natürlich darauf achten, immer den gleichen Massstab beim Export zu benutzen.

Um mehrere Linien (vor allem Vektorlinien) auf einmal in xfig verschieben zu können, gibt es den „Compound“-Knopf (Gruppieren) in der Mitte der linken Menüleiste. Es ist mit der Maus (mittlere Taste) eine Box um die zu gruppierenden Linien zu ziehen und dann mit einem weiteren Klick auf die mittlere Taste zu bestätigen. Oben rechts wird immer die Bedeutung der einzelnen Mausknöpfe erläutert.

Eine Besonderheit gibt es in xfig: Flächen werden gestapelt und können sich daher verdecken. Jeder Fläche ist eine Tiefe („depth“) zugeordnet, die Sie über den „Edit“-Knopf einstellen können. Klicken Sie zuerst auf „Edit“ im linken Menü, dann auf das einzustellende Objekt. Es erscheint das „Edit-Fenster“, in dem Sie zahlreiche Parameter neben der Stapeltiefe einstellen können. Je größer die „depth“, desto tiefer liegt das Objekt. Standardtiefe ist „100“. In der Abbildung 25 hat die importierte Karte (als Bild importiert) die Tiefe „100“, die Ellipsen (obwohl sie transparent sind) wurden mit „99“ (kleinere Tiefe) belegt, damit sie über und nicht unter der Karte liegen. So lassen sich sehr einfach Zeichnungen und Kartenelemente anfertigen.

Nach kurzer Übungszeit können Sie mit xfig hochwertige Karten in beliebigen Papierformaten erzeugen und diese beispielsweise im Postscriptformat, aber zum Beispiel auch als PDF-Datei, abspeichern. Ein Beispiel zeigt die Abbildung 25.

9.3 Die Benutzung des CELL-Treibers

Alternativ zu ps.map kann es interessant sein, Karten aus GRASS hochauflösend für andere Programme (z.B. zur Postergestaltung) zu exportieren. Prinzipiell können alle im GRASS-Monitor darstellbaren Abbildungen (das heisst alle Datentypen) über den speziellen CELL-Treiber in eine Datei umgelenkt werden. Die Auflösung lässt sich beliebig hoch einstellen, limitierend sind eventuell nur die entstehenden Dateigrößen.
Wenn also Karten aus GRASS exportiert werden sollen, die sowohl Raster- als auch Vektordaten beinhalten, haben „Bildschirmfotos“ bekanntlich nur eine geringe Qualität. Daher wurde der CELL-Treiber geschaffen, der sich wie ein GRASS-Monitor verhält. Dieser Treiber wird gestartet, und alles, was nun über die Display-Befehle sonst angezeigt wird, in eine Datei umgeleitet, bis der Treiber wieder beendet wird. Der CELL-Treiber erzeugt Rasterdaten, dargestellte Vektordaten werden in das Rasterformat umgewandelt. Bei hoher Auflösung ist diese Umwandlung kaum erkennbar.
Ist die CELL-Ergebniskarte erzeugt (GRASS-Rasterdatei: D_cell), kann sie beispielsweise mit `$ r.out.tiff` exportiert werden. Die Schwierigkeit bei der Benutzung des Treibers besteht darin, dass man nicht sehen kann, was passiert. Er muss quasi „blind“ bedient werden (deshalb ist anzuraten, vorher den Ablauf mit dem normalen GRASS-Monitor zu üben oder ein kleines Script zu schreiben).
Das genaue Verfahren läuft folgendermaßen ab:

1. Die Größe der Ergebniskarte ist anhand von zwei Umgebungsvariablen im GRASS-Fenster festzulegen (Pixelanzahl in x- bzw. y-Richtung)

   ```
   $ export GRASS_WIDTH=xxx
   $ export GRASS_HEIGHT=yyy
   ```

   Beispiel ($1500 * 1000$):

   ```
   $ export GRASS_WIDTH=1500
   $ export GRASS_HEIGHT=1000
   ```

2. Dann wird der CELL-Treiber gestartet (dauert je nach Rechnergeschwindigkeit etwas, da der oben definierte Speicherplatz nun reserviert wird):

   ```
   $ d.mon start=CELL
   ```

3. Nun können alle Anzeigemodule für Karten, Legenden usw. von GRASS (d.*-Befehle) benutzt werden, wie beispielsweise `$ d.rast` oder `$ d.vect`. Die Daten werden vom CELL-Treiber in die GRASS-Rasterdatei „D_cell“ geschrieben.

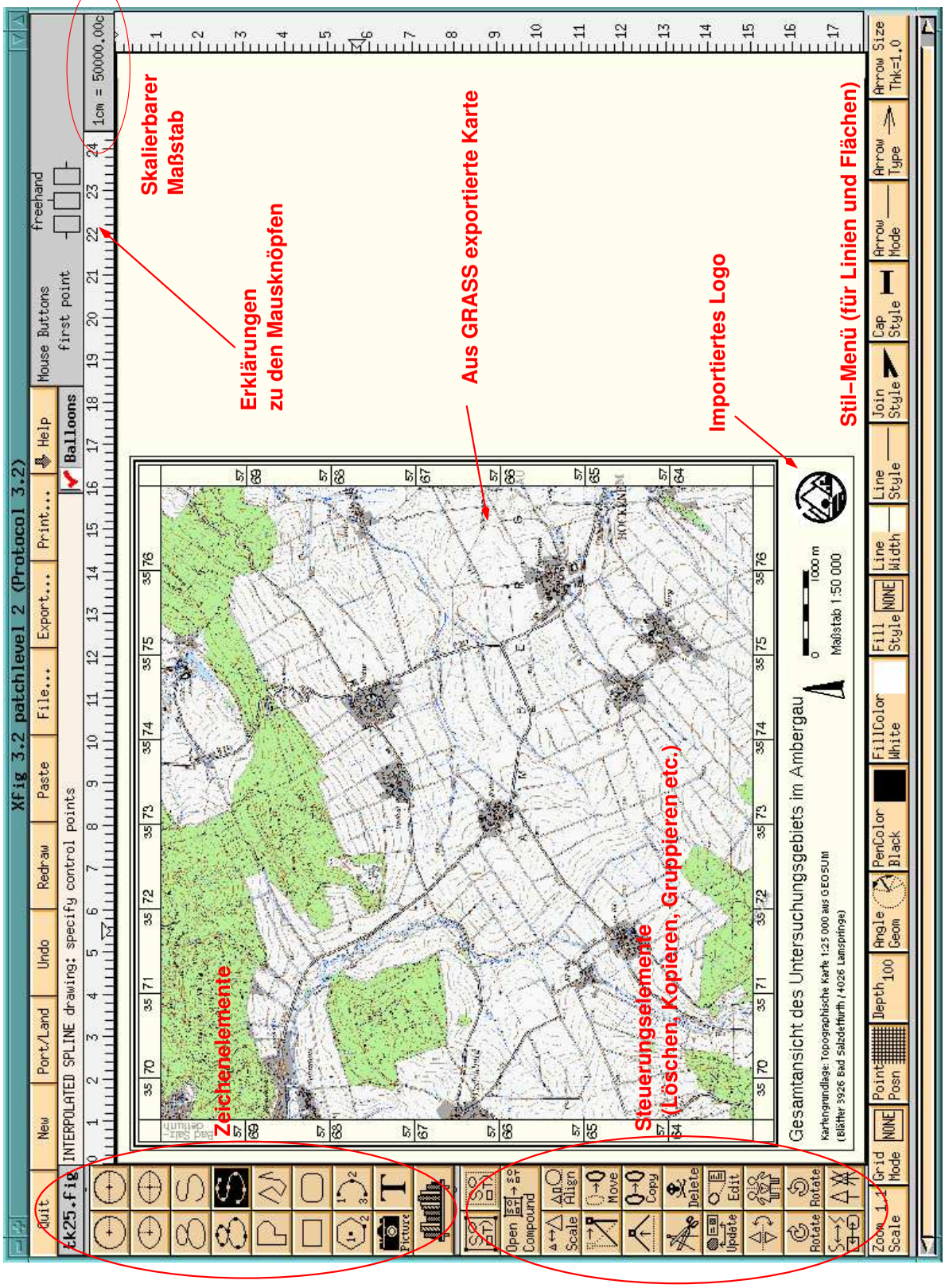


Abbildung 25: Kartenherstellung mit xfig

4. Wenn alle gewünschten Befehle aufgerufen wurden, wird der CELL-Treiber beendet. Dabei erzeugt GRASS die gewünschte Datei „D_cell“:

   ```
   $ d.mon stop=CELL
   ```

5. Die Datei „D_cell“ wird grundsätzlich in der xy-Projektion dargestellt, obwohl sie vielleicht in einer Gauß-Krüger-Projektion erzeugt wurde. Um die Datei zu betrachten oder zu exportieren, müssen daher erst die mapset-Einstellungen auf die Datei „D_cell“ umgesetzt werden. Praktischerweise sollte vorher die aktuelle Regionseinstellung gespeichert werden, um sie nach dem Export der erzeugten CELL-Datei auf einfache Weise wiederherstellen zu können:

   ```
   $ g.region save=aktuelle_region
   ```

   Nun wird die mapset auf die xy-Projektion der neuen „D_cell“ Datei umgesetzt:

   ```
   $ g.region raster=D_cell
   ```

   Dann kann entweder ein GRASS-Monitor mit `$ d.mon` zum Betrachten der Karte „D_cell“ gestartet oder die Datei „D_cell“ beispielsweise mit

   ```
   $ r.out.tiff
   ```

   exportiert werden. Ist der Monitor bereits geöffnet, muss `$ d.erase` aufgerufen werden, um auch den Monitor auf die xy-Projektion umzustellen.

6. Zum Schluss sollte die ursprüngliche (Gauß-Krüger-)Region wieder aktiviert werden, die vorher gespeichert worden war:

   ```
   $ g.region region=aktuelle_region
   ```

   Läuft der GRASS-Monitor bereits, wird er mit `$ d.erase` von der xy-Projektion von „D_cell“ wieder in die übliche Projektion zurückgesetzt.

Die exportierte Rasterdatei „D_cell“ (die auch gerasterte Vektoren enthalten kann) lässt sich dann beispielsweise für die Kartenherstellung in `$ xfig` verwenden (vgl. vorigen Abschnitt 9.2).
Die Benutzung des CELL-Treibers erscheint vielleicht etwas umständlich, dafür können aber auch Karten in beliebiger Größe mit hoher Auflösung erzeugt werden.
Alternativ läßt sich auch der PNG-Treiber benutzen (ab GRASS 5.x).

10 GRASS im praktischen Einsatz: Einige Anwendungsbeispiele

Für dieses Anwendungskapitel (und natürlich auch die übrigen Kapitel) sollten Sie sich über das Internet oder per CD-ROM den „SPEARFISH"-Testdatensatz für GRASS laden.[1] Dieser Datensatz ist ein umfangreicher GIS-Datensatz mit Raster-, Vektor- und Punktdaten im Format der GRASS-Datenbank. Zusätzlich gibt es den dazu passenden Datensatz „IMAGERY", der eine SPOT-Satellitenbildszene und einen Stereo-Luftbildsatz enthält. Die Daten beziehen sich auf dieselbe Region, sind aber nicht geocodiert (das Vorgehen lernen Sie in den beiden anschließenden Kapiteln in allgemeiner Form kennen).

Der Datensatz „SPEARFISH" deckt eine Fläche von zwei Kartenblättern im Maßstab 1:24.000 im westlichen South Dakota, U.S.A., ab. Die Namen der *quads* sind: Spearfish und Deadwood North, SD. Auch ein Großteil des „Black Hills National Forest" (Mount Rushmore) liegt im Gebiet. Die zugrunde liegende Projektion ist UTM (Universe Transverse Mercator) in der Zone 13 (4928000N, 4914000S, 590000W und 609000E) mit einer Rasterauflösung bis maximal 20m Zellenlänge. Einige Karten weisen auch nur 100m Zellenlänge auf. Informationen zu jeder Karte können Sie mit `$ v.report` und `$ r.report` abfragen. Datenquellen für diesen Datensatz waren das EROS Data Center, USACERL, USGS, U.S. Census Bureau, und die SPOT Image Corporation. Eine ausführlichere Kartenbeschreibung liegt im Internet.[2]

10.1 Installation des SPEARFISH-Datensatzes

Kopieren Sie den Datensatz in Ihr Arbeitsverzeichnis (Homeverzeichnis). Sie können mit `$ cd ~` hineinwechseln. Falls noch nicht vorhanden, erzeugen Sie ein Verzeichnis für die GRASS-Datenbank:

```
$ mkdir grassdata
```

und wechseln Sie hinein:

```
$ cd grassdata
```

Nun kann der Datensatz ausgepackt werden (die Punkte „../" zeigen an, dass sich der Datensatz im übergeordneten Verzeichnis befindet):

```
$ tar xvfz ../spearfish_grassdata.tar.gz
```

[1] Quelle für den „SPEARFISH"-Datensatz: `http://grass.itc.it/` (Rubrik „sample data")

[2] `http://grass.itc.it/gdp/tutorial/spearDB.ps.gz`

Die Liste der erscheinenden Dateien zeigt Ihnen an, dass die Daten nun im Unterverzeichnis „spearfish“ gespeichert werden. Anschließend sind, um eventuelle Probleme zu vermeiden, Lese- und Schreibrechte für Sie zu setzen:

```
$ chmod -R u+rw *
```

Dann können Sie wieder in Ihr Arbeitsverzeichnis zurückwechseln:

```
$ cd ~
```

und das Datenpaket löschen (wenn alles geklappt hat):

```
$ rm spearfish_grassdata.tar.gz
```

10.2 Arbeiten mit dem SPEARFISH-Datensatz

Nach erfolgreicher Installation des „SPEARFISH“-Datensatzes rufen Sie nun GRASS entsprechend auf (je nach verwendeter GRASS-Version):

`$ grass4.3` oder `$ grass5`

Es erfolgt die Angabe des gerade installierten Beispieldatensatzes (auf Groß- und Kleinschreibung ist bei UNIX zu achten!):

```
                                GRASS 5.0.0

     LOCATION: This is the name of an available geographic location.  -spearfish-
               is the sample data base for which all 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 with the command: g.region
     - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     LOCATION:   spearfish_____             (enter list for a list of locations)
     MAPSET:     PERMANENT_____             (or mapsets within a location)
     DATABASE:   /home/emil/grassdata_____________________________

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

Als „DATABASE“ tragen Sie Ihr entsprechendes Arbeitsverzeichnis ein. Wenn Sie es nicht wissen, brechen Sie mit <Strg><C> den Start von GRASS ab und geben Sie ein:

```
$ echo $HOME
```

Dann rufen Sie GRASS erneut auf.

Sind *location*, *mapset* und database korrekt eingetragen, geht es mit „ESC“-„RETURN“ nun weiter. Sie sehen die Lizenzmeldung von GRASS und dann das Kommandozeilen-Promptzeichen „GRASS: >“ Damit sind Sie in GRASS „angekommen“. Hier lassen sich die GRASS-Module, aber

auch alle UNIX-Programme aufrufen. Sie können nun, wenn Sie möchten, auch die graphische Benutzeroberfläche starten:

```
$ tcltkgrass &
```

Im Folgenden werden aber alle GIS-Abfragen mit den Modulnamen erläutert. Die meisten Funktionen sind auch in der „TclTkGRASS"-Oberfläche integriert. Über die Funktionsmenüs können Sie sie entsprechend finden.

Als erstes sollten Sie einen GRASS-Monitor für die Visualisierung von GIS-Daten öffnen:

```
$ d.mon x0
```

oder in TclTkGRASS: DISPLAY ⇝MONITOR ⇝START ⇝X0

Die Monitorgröße sollte über die Umgebungsvariablen eingestellt werden (vgl. Anhang A.1). Sie können allerdings auch die Größe des GRASS-Monitors mit der Maus verändern, wenn er gerade nicht aktiv benutzt wird (d.h. gerade Daten darin zur Anzeige aufgebaut werden).

Damit Sie nun erst einmal die vorhandenen Karten betrachten können, müssen Sie die Namen kennen. Zur Anzeige der vorhandenen Karten geben Sie, je nach Kartentyp, ein:

```
$ g.list vect
$ g.list rast
$ g.list sites
```

Alternativ können Sie auch nur `$ g.list` ohne Parameter aufrufen, dann werden Sie per Menü geführt. Geben Sie „help" als Parameter ein, erscheint generell bei GRASS-Modulen die Übersicht der möglichen Parameter (z.B. `$ g.list help`). Parameter in eckigen Klammern sind optional, im Menü wird bei diesen optionalen Parametern „required: no" angegeben und Sie können hier mit „return" quittieren.

Betrachten Sie als erstes beispielsweise die geologische Karte des Gebiets im Rasterformat:

```
$ d.rast geology
```

Die dargestellte Karte lässt sich mit

`$ d.what.rast map=geology` (oder einfach `$ d.what.rast geology`)

mit der Maus abfragen. Es ist ganz wichtig, zur Beendigung des Abfragemodus die rechte Maustaste im GRASS-Monitor zu betätigen (wie auch vom Modul angegeben). Ansonsten ist der Monitor blockiert, da das Modul ja auf einen Mausklick wartet.

Dieser Karte können Sie die Bodentypenkarte (z.B. in Rot) überlagern (vgl. Abb. 26):

```
$ d.vect soils c=red
```

Da die Bodentypenkarte auch im Rasterformat vorliegt, lässt sie sich entsprechend anzeigen:

```
$ d.rast soils
```

Auch im Rasterformat können Karten überlagert werden, allerdings sind nur Zellen der „unterliegenden" Karte sichbar, wenn die überlagerte Karte Lücken (also „no-data"-Zellen) aufweist. Als Beispiel können Sie Feldgrenzen im Rasterformat der eben angezeigten Bodentypenkarte überlagern:

```
$ d.rast -o fields
```

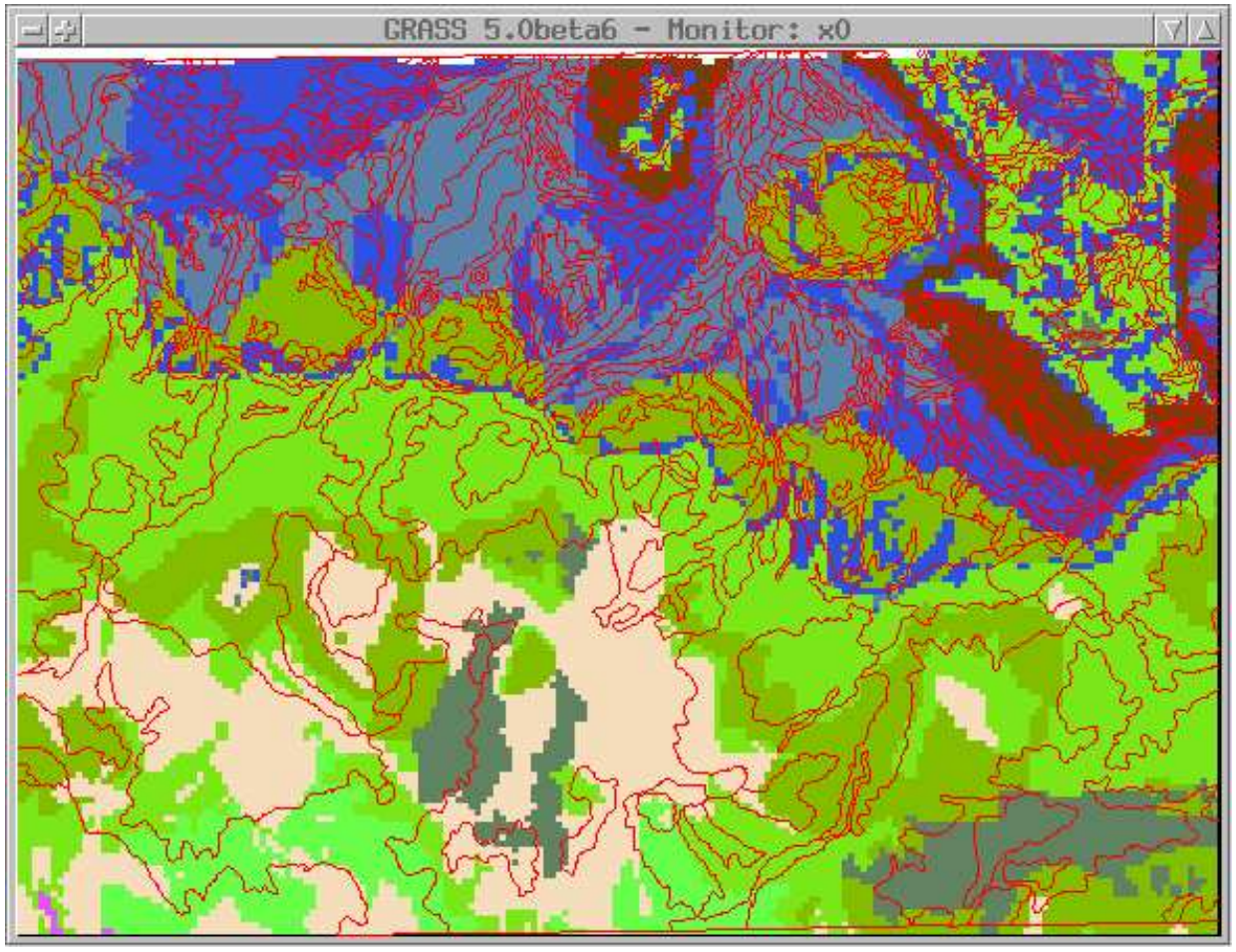


Abbildung 26: Geologische Rasterkarte mit überlagerten Bodentypgrenzen im Vektorformat (Spearfish-Datensatz)

Es lassen sich auch zwei oder mehr Karten gemeinsam mit der Maus abfragen (auch hier mit linker Maustaste abfragen, mit der rechten Maustaste die Abfrage beenden). Es darf allerdings generell bei GRASS kein Leerzeichen nach einem Komma bei Aufzählungen folgen:

```
$ d.what.rast soils,geology
```

Bei einer Abfrage im Monitor erscheint dann beispielsweise:

```
Buttons
 Left:  what's here
 Right: quit

602210(E) 4918490(N)
  soils in PERMANENT   (9)CBE
geology in PERMANENT   (4)sandstone
```

Als einfaches Beispiel für eine Berechnung im GIS soll der durchschnittliche pH-Wert im Gesamtgebiet untersucht werden. Die Ergebnistabellen sind aus Platzgründen hier nicht angegeben. Die Berechnung eines Durchschnittswerts für eine Rasterkarte erfolgt mit einem in GRASS enthaltenen Script (Scripte verhalten sich wie übliche GRASS-Module):

```
$ r.univar soils.ph
```

Sie können sich die Legende zur pH-Wert-Datei anschauen mit:

```
$ r.report soils.ph
```

Möchten Sie den durchschnittlichen pH-Wert pro Feld berechnen, wird zwischen Basiskarte und Bedeckungskarte unterschieden:

```
$ r.average base=fields cover=soils.ph out=ph.durchschnitt
```

Einen umfangreichen Report über die pH-Wert-Bereiche in Bezug zu den Feldbesitzern (unter Angabe der Flächengrößen, hier im km^2 und ha) erhalten Sie mit:

```
$ r.report m=soils.ph,fields units=k,h
```

Da die Bodentypenkarte auch als Vektorkarte vorliegt, können Sie sich die Flächenverbreitung der Bodentypen ebenfalls in Form eines Reports ausgeben lassen:

```
$ v.report soils type=area unit=k
```

Nun soll eine neue Karte berechnet werden, die als Bodentypenkarte nur leicht bis stark saure Böden enthält (sonst NULL). Es werden dazu die Klassen der pH-Wert-Karte kleiner 3 benutzt (also pH-Wert laut Legende unter pH 6.6). Mit r.mapcalc kann die Abfrage gestaltet werden:

```
$ r.mapcalc ''soils.sauer=if(soils.ph<3,soils,null())''
```

Sie besagt also: Wenn pH-Klasse kleiner als 3, dann kopiere für diese Rasterzelle die Bodentypennummer, sonst setze sie auf NULL. Diese Abfrage wird automatisch für alle Rasterzellen durchgeführt. Die Anführungszeichen (doppelte, auf der deutschen Tastatur auf Taste 2 zu finden) sind nur dann zu setzen, wenn r.mapcalc mit der Formel als Parameter aufgerufen wird. Wenn Sie erst r.mapcalc aufrufen und dann die Formel innerhalb des Moduls eingeben, sind die Anführungszeichen nicht anzugeben.

Nun fehlen allerdings die Textattribute in der Karte, da r.mapcalc sie nicht überträgt. Diese können mit dem UNIX-Programm „join“ automatisch über eine Reklassifizierungsdatei verknüpft werden. Zuerst wird eine Datei erzeugt, die die in der Karte „soils.sauer“ enthaltenen Bodentypennummern (die obiger Bedingung genügen) enthält:

```
$ r.stats soils.sauer > reclass1.txt
```

Ein Blick in die Datei zeigt, dass nur wenige Bodentypen in „soils.sauer“ enthalten sind (Ergebnis hier horizontal statt vertikal dargestellt):

```
$ more reclass1.txt

9  10  14  15  22  23  35  42  54  *
```

Der Stern steht für die NULL-Werte.

Nun wird eine vollständige Textattributdatei aus der Ausgangskarte erzeugt (quasi eine Legendendatei):

```
$ r.stats -l soils > reclass2.txt
```

Nach Anzeige der Bodenkartenlegende ist ersichtlich, dass jeder Bodentyp aus einer Nummer und einem Textkürzel (U.S.-Klassifikation) besteht (erster Buchstabe: Bodenname, zweiter Buchstabe:

Kartiereinheit, dritter Buchstabe (wenn vorhanden): Hangneigungsklasse). Eine genaue Beschreibung der Bodentypen und vieler bodenphysikalischer Parameter im Spearfish-Gebiet ist dem Handbuch „Lawrence County, South Dakota Soils Survey, August 1979, USDA, NRCS“ entnehmbar. Die Bodenkartenlegende stellt sich auszugsweise so dar:

```
$ more reclass2.txt

1 Aab
2 Ba
3 Bb
4 BcB
5 BcC
6 BeE
[...]
```

Nun sollen automatisiert die entsprechenden Textattribute der erstellten Legende der Karte „soils.sauer“ zugeordnet werden. Das erfolgt mit dem UNIX-Programm „join“. Es verhält sich wie ein Datenbankprogramm und ordnet zwei Dateien zeilenweise über ein gemeinsames Attribut zu. Damit werden für Attribute, die in der Legende „soils.sauer“ existieren, die Textattribute zeilenweise übertragen.

```
$ join reclass1.txt reclass2.txt > reclass3.txt
```

Das Ergebnis sieht so aus:

```
$ more reclass3.txt

9 CBE
10 CaD
14 GaD
15 GcD
22 MaC
23 MaD
35 Pe
42 Sd
54 Wb
* no data
```

Sie sehen, dass Sie ohne zusätzliche Datenbank auf einfache Weise Verknüpfungen erstellen können. Da „join“ über gemeinsame Attribute zuweist, ist die interne Reihenfolge in den Dateien unwichtig. Interessant wird die Nutzung der UNIX-Textwerkzeuge vor allem dann, wenn Sie GIS-Scripte programmieren.

Nun ist eine GRASS-typische Zuweisungsdatei zum Reklassifizieren zu erstellen. Die Struktur sieht so aus:

```
altes_Attribut = neues_Attribut optionales_Textattribut
```

Da hier die Textattribute zugewiesen werden sollen, wird eine entsprechende Zuordnung durchgeführt. Die Erzeugung kann wieder automatisiert erfolgen über das UNIX-Programm „paste“, das zeilenweise Dateien verknüpft (allerdings direkt und nicht über gemeinsame Attribute wie bei „join“). Sie verknüpfen nun also die Legendendatei der Karte „soils.sauer“ ohne Textattribute mit der Legendendatei mit Textattributen und fügen noch ein Gleichzeichen ein, das später von r.reclass erwartet wird:

```
$ paste -d'=' reclass1.txt reclass3.txt > reclass.rules
$ more reclass.rules

9=9 CBE
10=10 CaD
14=14 GaD
15=15 GcD
22=22 MaC
23=23 MaD
35=35 Pe
42=42 Sd
54=54 Wb
*=* no data
```

Es bleibt nur noch die Reklassifizierung der Karte „soils.sauer“, um die Textattribute zuzuweisen:

```
$ cat reclass.rules | r.reclass in=soils.sauer out=soils.sauer2
```

Damit haben Sie nun eine neue Bodenkarte mit Textattributen. Es wäre denkbar, dass Sie für diese vier Schritte ein Script schreiben, mit dem Sie die Zuweisung dann ganz automatisch durchführen. Beachten Sie: Bei reklassifizierten Karten darf die Basiskarte (hier: „soils.sauer“) nicht gelöscht werden, da es sich bei reklassifizierten Karten nur um eine interne Zuweisung handelt.

Als weiterer Aufgabenteil sollen die Flächen der Karte „soils.sauer2“ automatisch vektorisiert werden. Dazu geben Sie ein:

```
$ r.poly -l in=soils.sauer2 out=soils.sauer2
```

Das Modul erzeugt eine Vektorkarte „soils.sauer2“ (mit gerundeten Ecken durch „-l“). Für diese Karte ist anschließend die Topologie aufzubauen:

```
$ v.support soils.sauer2 option=build
```

Die Attribute werden beim automatischen Vektorisieren mitübertragen. Nun können Sie die erzeugte Vektorkarte anschauen:

```
$ d.erase
$ d.vect soils.sauer2
```

Mit

```
$ d.area soils.sauer2
```

können Sie Flächen auch ausgefüllt anzeigen lassen (und optional Flächen- und Linienfarbe wählen). Einen Maßstab können Sie mit der Maus auf der Karte auch platzieren:

```
$ d.barscale -m
```

Die angezeigten Flächenattribute lassen sich mit der Maus abfragen, es werden auch die jeweiligen Flächengrößen mit angegeben:

```
$ d.what.vect soils.sauer2
```

Alternativ können Sie auch ein anderes Modul benutzen, das Ihnen die selektierte Fläche jeweils farbig markiert:

```
$ v.area soils.sauer2
```

Eine Angabe der Flächengrößen dieser Vektorflächen (Vektortyp=Flächen, Einheit: km^2) erhalten Sie mit

```
$ v.report soils.sauer2 type=area units=k
```

Wollen Sie bestimmte Flächen grafisch darstellen, kann das selektiv mit

```
$ d.vect.area -f soils.sauer2 cat=23 col=red
```

erfolgen (hier mit Flächenfüllung in Rot).

10.3 Erosionsberechnung mit der USLE

In diesem Abschnitt wird vorgestellt, wie rasterbezogen erosive Abträge berechnet werden können. Das Verfahren ist zur Veranschaulichung stark vereinfacht, es soll Ihnen das prinzipielle Vorgehen zeigen.
Die allgemeine Bodenabtragsformel lautet (WISCHMEIER ET AL. 1978):
$A = R * K * L * S * C * P$

mit:

- A: mittlerer Bodenabtrag
- R: Niederschlagsfaktor
- K: Substratfaktor
- L: Hanglängenfaktor
- S: Hangneigungsfaktor (als LS: Topographiefaktor)
- C: Bedeckungs- und Bewirtschaftungsfaktor
- P: Bodenschutzmaßnahmenfaktor

Nun sind die einzelnen Parameter aus den vorliegenden Grundlagenkarten (vor allem „elevation.dem“) und weiteren Zahlenwerten zu berechnen. Bitte beachten Sie, dass mit dem „SPEARFISH“-Datensatz für diese Thematik keine guten Ergebnisse möglich sind, da das Höhenmodell nur mit einer vertikalen Meter-Auflösung ohne Nachkommastellen vorliegt.

In einem kleinen Exkurs werden Ihnen nun mehrere Möglichkeiten vorgestellt, eine Höhenrasterkarte höherer Auflösung zu erzeugen. Diese Karte enthält dadurch nicht mehr Infor-

mationen, aber Fließkommazahlen gegenüber der Integerkarte „elevation.dem“. Dadurch sind die folgenden Ergebnisse der Abtragsberechnung etwas besser. Mit eigenen Daten, die üblicherweise im Fließkommaformat von den Vermessungsämtern geliefert werden, sollten Sie keine Probleme haben. Die drei aufgezeigten Wege können Sie auch überspringen und bei „R-Faktor“ wieder einsteigen.

Weg 1: Methode Spline-Interpolation aus Höhenlinien

Hier werden zunächst Vektorhöhenlinien aus dem vorhandenen Höhenmodell „elevation.dem“ erzeugt, die Topologie aufgebaut und aus diesen Linien dann ein neues Rastermodell mit etwas höherer Auflösung interpoliert (RST-Methode):

```
$ r.contour -q in=elevation.dem step=10 out=contour10
$ d.vect contour10
$ g.region -p res=20
$ v.surf.rst in=contour10 el=elev20
$ d.rast elev20
```

Weg 2: Methode Spline-Interpolation aus Höhenpunkten aus Höhenlinien

Alternativ können Sie die Höhenlinien in Höhenpunkte umwandeln und dann wieder eine Rasteroberfläche interpolieren:

```
$ r.contour -q in=elevation.dem step=10 out=contour10
$ g.region -p res=20
$ v.to.sites -dia in=contour10 out=hoehenpunkte
$ s.surf.rst in=hoehenpunkte elev=elev20
$ d.rast elev20
```

Weg 3: Methode Spline-Interpolation aus Höhenpunkten aus Höhenmodell

Die dritte Methode basiert auf der Umwandlung von Rasterhöhenwerten in eine Höhenpunktkarte. Daraus wird dann mit Splines wiederum das Rasterhöhenmodell mit höherer Auflösung gewonnen:

```
$ r.to.sites -a in=elevation.dem out=hoehenpunkte
$ g.region -p res=20
$ s.surf.rst in=hoehenpunkte elev=elev20
$ d.rast elev20
```

Bitte setzen Sie im Folgenden gegebenenfalls den Namen Ihrer neu interpolierten Rasterhöhenkarte anstelle von „elevation.dem“ ein.

Nun werden also die einzelnen Faktoren für die Abtragsrechnung bestimmt. Für den R-Faktor wird ein fiktiver Wert von 65.0 angenommen, statt einer Berechnung über eine im Spearfish-Gebiet gültige Regressionsgleichung.

Der als nächster Faktor benötigte Substratfaktor (K-Faktor) ist bereits in der Datenbank regionalisiert als Rasterkarte vorhanden: „soils.Kfactor“ (Sie können `$ g.list rast` zur Überprüfung des Vorhandenseins benutzen).

Nun folgen einige Berechnungen für die folgenden Faktoren. Da die Höhenmodellkarte „elevation.dem“ in einer Auflösung von 30m Zellenlänge (oder ggf. 20m) vorliegt, sollten Sie diese Genauigkeit für die nun zu berechnenden Karten einstellen:

```
$ g.region res=30
```

Sie könnten theoretisch auch mit geringerer Genauigkeit rechnen (z.B. nur 100m Zellenlänge), in diesem Fall würden Zellen der Karte „elevation.dem“ entsprechend zusammengefasst und die Werte gemittelt. Alternativ zu obigem Befehl lässt sich eine Region auch nach Randkoordinaten und Auflösung einer Rasterkarte einstellen:

```
$ g.region -p rast=elevation.dem
```

Die zusätzliche Option „-p“ (print) gibt Ihnen die aktuell aus der Karte übernommenen Einstellungen an.

Zur Berechnung des Topographiefaktors (kombinierter LS-Faktor) wird das Höhenmodell analysiert. Zusätzlich sollen noch einige weitere Karten erzeugt werden, die in Betrachtung mit dem Bodenabtrag interessant sind. Aus dem Höhenmodell wird nun die lokale Hanglänge unter Berücksichtigung einer Barrierekarte berechnet (L-Faktor), das verwendete Modul erzeugt zusätzlich noch eine Karte der Fließakkumulation und der Fließliniendichte. Als Barrierekarte soll das Straßennetz („roads“) und das Schienenwegenetz („railroads“) dienen, da sie üblicherweise gegenüber der Umgebung leicht erhöht sind. Diese liegen als Vektor- und auch als Rasterkarten vor. Um eine zum Höhenmodell identische Auflösung zu haben, werden sie vom Vektor- in das Rasterformat konvertiert und dann zur Barrierekarte überlagert:

```
$ v.to.rast in=roads out=strassen
$ v.to.rast in=railroads out=eisenbahn
```

Das Modul r.patch überlagert zwei oder mehr Rasterkarten. Die erste Karte wird dabei über die zweite gelegt usw. Zu beachten ist, dass gleiche Kategorienummern der zweiten, dritten... Karte der ersten Karte zugeordnet werden. In diesem Beispiel wird also die Bahntrasse (Kategorienummer 1) als „interstate“ (Kategorienummer 1 in „strassen“) geändert:

```
$ r.patch in=strassen,eisenbahn out=barrieren
```

Für diese Fragestellung einer Barrierekarte ist das nicht von Bedeutung. Sollen die Attribute jedoch erhalten bleiben, ist vorher entweder „railroads“ als Vektorkarte zu reklassifizieren (Zuordnung einer anderen Kategorienummer für die Bahntrasse) oder „eisenbahn“ als Rasterkarte. Das kann mit `$ v.reclass` bzw. `$ r.reclass` erfolgen. In den üblichen Anwendungsfällen, in denen r.patch eingesetzt wird, ist dieser Verschmelzungseffekt dagegen gewünscht (z.B. Erstellung von Karten-

mosaiken). Sie sollten sich vorher also mit `$ v.report` bzw. `$ r.report` die Kartenlegende anzeigen lassen.
Nach Start eines GRASS-Monitors (`$ d.mon x0`) kann diese Barrierekarte betrachtet werden:

```
$ d.rast barrieren
```

Falls Sie in der Karte „zoomen“, sollten Sie daran denken, auf das große Projektgebiet zurückzustellen (`$ g.region rast=elevation.dem`). Ansonsten rechnet GRASS nur im Zoomausschnitt. Diese Eigenschaft können Sie aber für Rechenversuche zur Verkürzung der Rechenzeiten nutzen bzw. um kleine Ausschnitte ohne weiteren Aufwand zu exportieren.
Die für die Barrierenkarte benötigten „Zwischenkarten“ können wieder gelöscht werden (denken Sie daran: keine Leerzeichen bei Aufzählungen von Dateinamen):

```
$ g.remove rast=strassen,eisenbahn
```

Nun folgt die Berechnung der Hanglängen, der Fließakkumulation (Fließliniendichte) und Fließlinien (in einer Zeile einzugeben):

```
$ r.flow elevin=elevation.dem barin=barrieren lgout=lfaktor dsout=liniendichte
flout=fliesslinien
```

Die Karte „fliesslinien“ ist eine Vektorkarte, für die die Topologie aufzubauen ist:

```
$ v.support fliesslinien o=build
```

Standardmäßig wird „downhill“ (konvergierender Oberflächenabfluss) berechnet, der Parameter „-u“ kehrt die Berechnungsrichtung zu „uphill“ (dispergierender Oberflächenabfluss) um. Die berechneten Hanglängen sind 2D-Längen (also auf die Horizontale projiziert), um echte 3D-Längen zu erhalten, ist der Parameter „-3“ zu setzen.
Die Fließliniendichte („downhill“ berechnet) ergibt mit der Auflösung multipliziert die jeweils beisteuernde Fläche für einen Fließweg (hier in m^2):

```
$ r.mapcalc ''akk_flaeche=liniendichte * ewres() * nsres()''
```

Bei entsprechender Vergrößerung (Zoom mit d.zoom) können die Zellenwerte angezeigt werden:

```
$ d.rast liniendichte
$ d.rast.num liniendichte
```

Nach diesem Exkurs ist nun der nächste Parameter für der Abtragsgleichung zu berechnen. Es wird die Hangneigung (slope) in Grad benötigt, die im Datensatz zwar vorhanden ist, aber noch einmal für die 30m-Auflösung erstellt werden soll. Zusätzlich kann eine Expositionskarte (aspect) erzeugt werden, die nicht direkt benötigt wird:

```
$ r.slope.aspect elev=elevation.dem slope=slope.dem aspect=aspect.dem
```

Sie können nun diese Karten betrachten, insbesondere die Expositionskarte zeigt Ihnen anschaulich das Spearfish-Gebiet.

Der LS-Faktor berechnet sich aus der Multiplikation von L- und S-Faktor:

```
$ r.mapcalc ''lsfaktor1=lfaktor * slope.dem''
```

Es existiert auch eine alternative Formel (vgl. MITASOVA ET AL. 1999, pers. Mittl.) für den LS-Faktor. Sie kann ebenfalls mit r.mapcalc berechnet werden (in einer Zeile einzugeben):

```
$ r.mapcalc ''lsfaktor2=1.6 * exp(liniendichte * ewres()/22.1, 0.6) *
exp(sin(slope.dem)/0.09, 1.3)''
```

Die Differenzbildung zeigt die Unterschiede:

```
$ r.mapcalc ''diff=lsfaktor1 - lsfaktor2''
```

Sie können sich diese Differenzenkarte mit d.rast anschauen bzw. die Statistik mit r.univar berechnen lassen.

Der Bewirtschaftungsfaktor berechnet sich aus der Landnutzung: Mit `$ r.report` können Sie sich die Legende der Karte „vegcover“ ausgeben lassen. Nun sollen, je nach Landnutzung, C-Faktoren zugewiesen werden (nach NETELER & MITASOVA 2002, S. 306):

1 irrigated agriculture	0.2
2 rangeland	0.1
3 coniferous forest	0.001
4 deciduous forest	0.0005
5 mixed forest	0.0005
6 disturbed	0.5

Man könnte nun mit aufwendigen if-Bedingungen die C-Faktorkarte erzeugen. Es geht aber auch einfacher: Kopieren Sie mit

```
$ g.copy vegcover,cfaktor
```

die Karte „vegcover“ als „cfaktor“ und bearbeiten Sie dann die Legende der neuen Datei mit

```
$ r.support
```

Nach Angabe des Kartennamens „cfaktor“ geht es weiter: „Edit the header for [cfactor]?“ no, „Update the stats...:“ no, „Edit the category file for [cfactor]?“ yes. Es wird angegeben, wieviele Attribute (Kategorien) in der Karte auftreten „Highest category: 6“. Diesen Wert lassen Sie so, da ja die Anzahl der auftretenden Werte sich nicht ändert. Dann wird die Legende editiert, indem die Landnutzungen durch die C-Faktoren ersetzt werden:

```
TITLE:  Vegetation Cover_____________________________________________
CAT   NEW CATEGORY NAME
NUM
 0   no data_____________________________________________________
 1   0.2_________________________________________________________
 2   0.1_________________________________________________________
 3   0.001_______________________________________________________
```

```
4   0.0005
5   0.0005
6   0.5
```

Anschließend geht es mit <ESC><RETURN> weiter, alle folgenden Fragen können Sie mit „return“ übergehen. Nun ist Ihre C-Faktorenkarte erstellt. Beachten Sie, dass die Rasterzellen nun als Wert die Klassennummer tragen und als Textattribut den C-Faktor. Ab GRASS 5 könnten Sie den C-Faktor auch direkt als Fließkomma-Rasterzellenwert speichern (über if-Bedingungen in r.mapcalc).

Da über den P-Faktor nichts bekannt ist, wird später P=1 eingesetzt. Zum Schluss fehlt noch der K-Faktor, der aber im SPEARFISH-Datensatz bereits als Karte vorliegt. Sie können sich wieder Informationen über die Karte „K-Faktor“ anschauen mit:

```
$ r.report soils.Kfactor
```

Nun wird der Bodenabtrag berechnet (P-Faktor = 1). Sie müssen darauf achten, ob Klassennummern oder Attribute einer Karte zu verrechnen sind. In GRASS 4 ist das Attribut Hilfsmittel, um Fließkommazahlen bei einer Rasterkarte zu speichern, ab GRASS 5 können die Rasterwerte auch direkt im Fließkommaformat abgelegt werden. Die Karten „K-Faktor“ und „C-Faktor“ beinhalten den zu verrechnenden Wert hier als Textattribut (Überprüfung mit r.report), der LS-Faktor wurde dagegen über eine Formel direkt berechnet und hat die uns interessierenden Werte direkt in den Rasterzellen gespeichert. Um auf das Textattribut (hier natürlich Zahlen) zugreifen zu können, stellen Sie den „Klammeraffen“ (@) dem Kartennamen voran:

```
$ r.mapcalc ''abtrag=65.0 * @soils.Kfactor * lsfaktor * @cfaktor *
1.0''
```

Damit ist die Abtragskarte (in t$*$ ha^{-1} $*$ a^{-1}) berechnet. Mit

```
$ d.rast abtrag
```

können Sie sich die Karte anschauen und die Statistik mit

```
$ r.univar abtrag
```

ermitteln. Das Modul `$ d.histogram abtrag` zeigt Ihnen die Häufigkeitsverteilung.

Um eine Gefährdungskarte zu erstellen, die als Attribut wiederum die Landnutzung trägt, können Sie folgende Rechnung durchführen:

```
$ r.mapcalc ''gefaehrdung=if(abtrag > 200, vegcover, null())''
```

Die Abfrage besagt: Wenn der Abtrag über 200 t$*$ ha^{-1} $*$ a^{-1} liegt, dann soll der entsprechende Rasterzellenwert aus der Landnutzungskarte kopiert werden, wenn der Abtrag kleiner ist, soll NULL („no data“) eingesetzt werden. Sie erhalten also eine Karte der Landnutzung, die nur da Werte aufweist, wo der Abtrag hoch ist. Genauso könnten Sie auch Hangneigungsbereiche ermitteln, in denen der Abtrag bodengefährdend ist.

10.4 3D-Visualisierung mit NVIZ

Ab GRASS 5 steht ein völlig neues Visualisierungswerkzeug zur Verfügung, das von Helena Mitasova, Bill Brown und Lubos Mitas (University of Illinois, GMS-Labs) auf SGI-Plattformen entwickelt wurde. Jaro Hofierka (Bratislava) portierte es plattformunabhängig nach Tcl/Tk für PC/Linux, SUN, usw. Mit NVIZ können eine oder mehrere Raster-/Vektor- und Punktdatenkarten in 3D visualisiert, 3D-Abfragen durchgeführt und Filmanimationen erstellt werden. Das Werkzeug kann hier nur überblicksartig vorgestellt werden, im Internet gibt es eine ausführliche Anleitung.

Das Modul wird folgendermaßen aufgerufen:

```
$ nviz
```

Nun kann zunächst eine Rasterkarte angegeben werden, die die Höheninformationen trägt. Hier geben Sie, wenn Sie den SPEARFISH-Datensatz benutzen, „elevation.dem“ an. NVIZ fragt weitere Rasterkarten ab, die Sie mit „return“ übergehen können. Sie werden gebraucht, um Profile (z.B. geologische Schichten) zu visualisieren. Alle weiteren Fragen können Sie ebenfalls mit „return“ quittieren, da Karten über Menüs auch in NVIZ geladen werden können. Nun erscheint das Visualisierungsfenster mit einem Drahtgittermodell Ihrer Höhenkarte und das Steuerungsmenü.

Wenn Sie nun den „Surface“-Knopf betätigen, wird Ihnen das Höhenmodell in voller Farbe (Modus: „full rendering“) angezeigt. Der Blickwinkel kann folgendermaßen gesteuert werden: Mit der Maus ist der „Punkt“ anzufassen und zu bewegen (linke Taste). Sie finden diesen Punkt unterhalb des „Surface“-Knopfes in einem Feld. Mit etwas Eingewöhnung können Sie jeden beliebigen Blickwinkel erreichen. Der „perspective“-Regler dient zur Einstellung nah/fern. Mit dem „height“-Regler lässt sich die Augenhöhe einstellen, der „zexag“-Regler steuert die Überhöhung (Multiplikator für die Höhendaten).

Alternativ zur Benutzung der Regler können Sie auch Zahlenwerte eingeben („return“ zum Abschluss nach Zahleneingabe).

Bei Veränderungen wird immer das Gittermodell angezeigt, mit „Surface“ bekommen Sie wieder die Farbansicht. Sie sollten dem Programm Zeit zum Bildaufbau lassen, Rechnungen werden immer zuende geführt, auch wenn Sie bereits Einstellungen verändert haben (das sehen Sie an der CPU-Auslastung).

Nun soll eine Vektorkarte zusätzlich dargestellt werden. Im Menü „Panel“ können Sie mit „Vektor“ das Vektormenü einblenden. Der „NEW“-Knopf erlaubt, eine Vektorkarte aus einer *mapset* zu wählen. Nehmen Sie hier z.B. die Straßenkarte „roads“. Nach dem Laden der Karte aktivieren Sie den Schalter „Display on surface(s)“ und klicken auf „Draw current“ bzw. den Knopf „Vectors“ (neben „Surface“). Sie sehen nun also die Vektorkarte. Die Liniendicke kann mit dem „Line width“-Regler eingestellt werden (z.B. auf „1“ einstellen), der „Color“-Knopf lässt Sie die Vektorfarbe wählen.

Um beide Karten, das Höhenmodell mit überlagerter Straßenkarte zu zeigen, zeichnen Sie zuerst mit „Surface“ die Rasterfläche neu, dann mit „Vectors“ die Vektorkarte. Auf vergleichbare Weise

lassen sich Punktdaten (Sites) anzeigen, Sie können zwischen verschiedenen Symbolen und Symbolgrößen wählen.

In Abbildung 27 sehen Sie NVIZ mit einer Raster-, Vektor- und Punktdatendarstellung. Die angezeigte Rasterauflösung lässt sich justieren. Um den Rechenaufwand bei einem ersten Probieren zu vermindern, können Sie erst einmal die Auflösung heruntersetzen. Wählen Sie dazu im „Panel“-Menü „Surface“. Dort gibt es unter anderem einen Regler „Polygon resolution“ zur Steuerung der Auflösung. Bei „1“ gilt die geometrische Auflösung der *location* (die Sie vor dem Aufruf von NVIZ mit `$ g.region` verändern können). Höhere Werte sind entsprechende Faktoren, um die Auflösung in der Darstellung zu reduzieren. Hier können Sie auch ein Gitter über die Karte legen und mit „Grid Resolution“ die Gitterweite justieren. Um das Gitter einzuschalten, ist bei „Surface Style“ „Wire/Polygon“ zu aktivieren. „Polygon“ bezieht sich hier auf die OpenGL-Dreiecke, die zur Visualisierung der Rasterkarten mit Beleuchtung und Schummerung berechnet werden. Je besser die Auflösung ist, desto besser sieht natürlich das Ergebnis aus. Entsprechend erhöht sich auch der

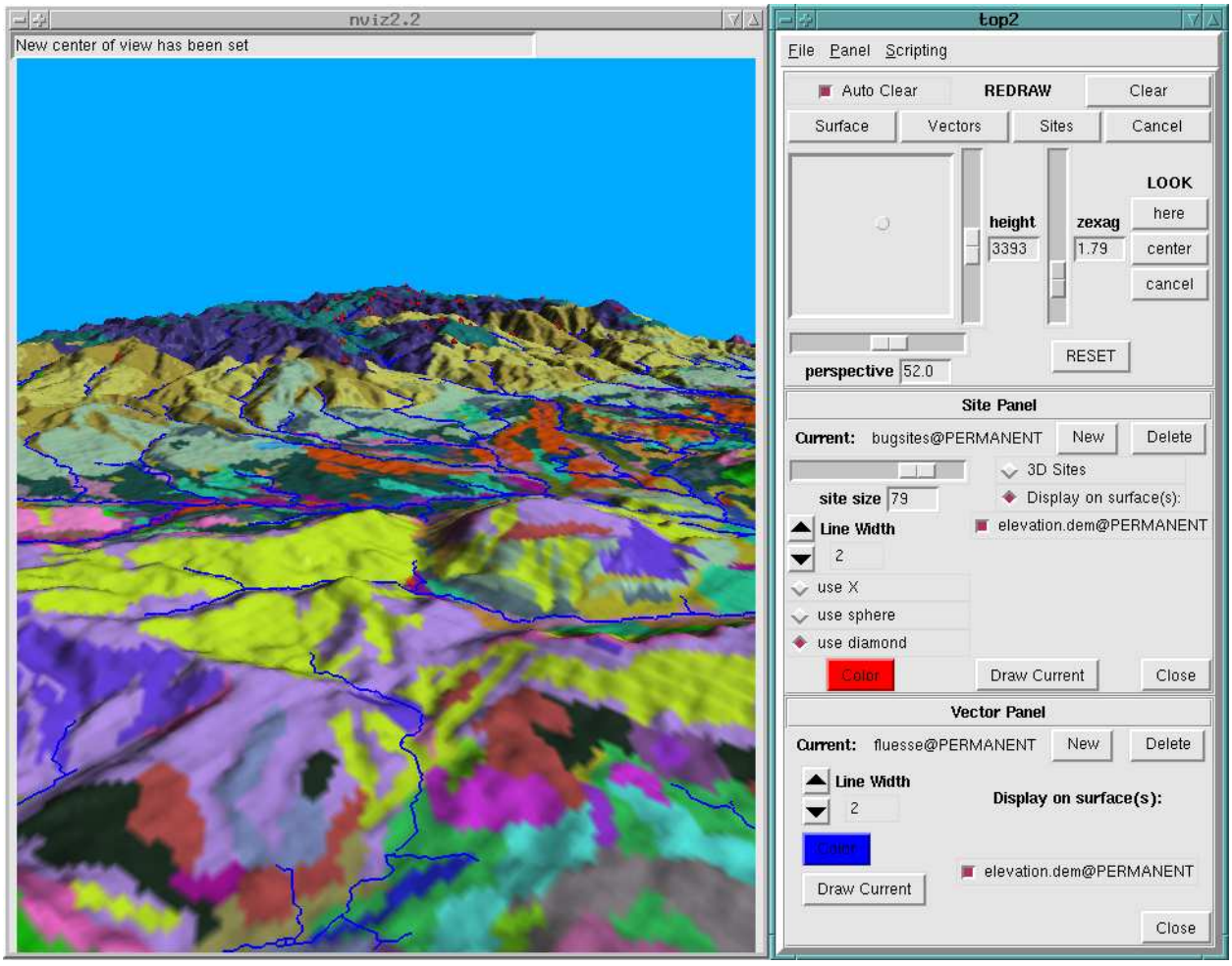


Abbildung 27: Bodenkarte mit überlagerten Flüssen im Vektorformat und Punktdaten (Spearfish-Datensatz)

Berechnungsaufwand bei der Visualisierung.

Nun soll die oben berechnete Rasterkarte „soils.sauer2“ mit Höheninformationen angezeigt werden. Sie benutzen also die Informationen Ihrer Karte als Farbgebung für das Höhenmodell. Dazu ist im Menü „Panel“ das Menü „Surface“ auszuwählen. Der Knopf „color“ lässt Sie die Karte „soils.sauer2“ auswählen. Mit „Draw current“ oder dem „Surface“-Knopf wird die Karte angezeigt.

Sie können in NVIZ 3D-Abfragen durchführen. Im Menü „Panel“ wählen Sie dazu das Menü „What's Here?“. Es erscheint eine neue Menüfläche mit einem Schalter „What's Here?“, der zu aktivieren ist. Dann können Sie in der Grafikausgabe Flächenattribute abfragen. Es werden in unserem Beispiel also Bodentypen der Karte „soils.sauer2“ angezeigt und die Strecke zwischen zwei Abfragepunkten. Über Knopf „Attributes“ kann festgelegt werden, was bei der Abfrage angezeigt werden soll.

Über das Menü „Panel“ - „Lights“ können Sie die Beleuchtung der Fläche(n) interaktiv verändern. Es erscheint ein Ball, der die Beleuchtungsverhältnisse gut darstellt, wenn Sie in diesem Menü etwas verändern („Show Model“ muss aktiviert sein). Die Regler „Brightness“, „Ambient“ sowie „Red“, „Green“, „Blue“ dienen der Steuerung der Beleuchtungsintensität und Farbgebung. Mit dem „Punkt“ im Feld sowie „Height“ können Sie die Position der Lichtquelle einstellen. Sie sehen die neuen Beleuchtungseffekte sofort am dargestellten Ball; wenn Sie den „Surface“-Knopf betätigen, für die Karte(n).

Als Weiteres soll nun Ihre Karte „soils.sauer2“ als Vektorkarte über „elevation.dem“ und der Rasterkarte „soils“ dargestellt werden. Sie erhalten damit also Vektorflächenumrandungen für die sauren Böden. Laden Sie die Karte „soils“ als „color“-Information in „Surface“. Wechseln Sie in das Vektormenü und laden Sie die oben hergestellte Vektorkarte „soils.sauer2“. Aktivieren Sie wie gehabt die Rasterkarte, auf der die Vektorkarte angezeigt werden soll. Mit dem „Surface“-Knopf können Sie die Bodenkarte darstellen, mit „Vectors“ die Flächenumrandungen darüberlegen. Sie sollten immer warten, bis die Rasterkarte aufgebaut ist, bevor Sie eine Vektorkarte zeichnen lassen.

Über „File“-„Image dump“ können Sie eine Ansicht als Grafikdatei speichern (SGI/RGB-Format, mit xv in PNG etc. umwandelbar). Über „File“-„Load state/Save state“ lassen sich aktuelle Einstellungen Ihrer Ansicht speichern.

Dynamische Visualisierungen in Form von Animationen sind auch mit NVIZ möglich. Es gibt zwei Menüs für diesen Zweck, zum einen „Animation“, zum anderen „Keyframe Animation“. Letztere Methode ist komplexer und u.a. lässt zeitgesteuerte Kameraschwenks bei variierter Flugrichtung zu. Im Folgenden soll nur die recht einfache „Animation“ vorgestellt werden.

Sie erreichen das Menü über „Panel“-„Animation“ (vgl. Abb. 28). Als „key frame“ wird ein Einzelbild bezeichnet, viele Einzelbilder ergeben dann kodiert im MPEG-Format den Film. Zunächst

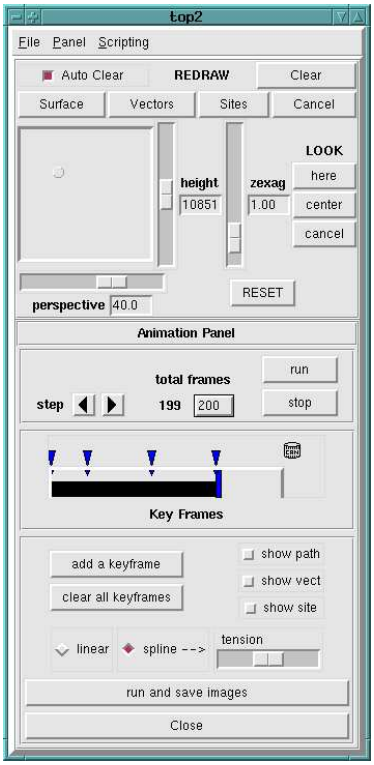


Abbildung 28: Animations-Menü

geben Sie an, wie viele Einzelbilder berechnet werden sollen: „total frames“. Die Voreinstellung ist 25, geben Sie beispielsweise 200 (danach „return“) an. Darunter befindet sich eine Zeitachse, die nun bis 200 reicht. Justieren Sie nun die Kameraposition (Betrachtungsauge) auf die Position, mit der die Animation beginnen soll. Es reicht das Drahtgittermodell aus. Nun klicken Sie auf „add a keyframe“ und diese Position ist gespeichert. Jetzt ist auf der Zeitachse die nächste Position zu bestimmen. Prinzipiell funktioniert es so, dass Sie mit der Kamera ein Stück weiterfahren und die Zwischeneinstellungen automatisch berechnet werden. Je nachdem, wo Sie die nächste Position auf der Zeitachse setzen und wieweit Sie mit der Kamera weiterfahren, ergibt sich daraus die Fluggeschwindigkeit. Da diese Animation nur sehr einfach ist, haben Sie im Gegensatz zum Menü „keyframe animation“ keine Minuten- und Sekundenangabe. Sie müssen also schätzen, wo ungefähr die nächste Marke zu setzen ist. Sie können, um erst einmal zu sehen, wie das Verfahren funktioniert, mit vier weiteren Marken und Kameraverschiebungen auskommen (also jeweils nach 25% Zeitanteil auf der Zeitachse). Das Setzen einer Marke auf der Zeitachse wird Ihnen durch ein blaues Dreieck angezeigt. Nun fahren Sie mit der Kamera weiter (z.B. „Punkt“ im Feld oder „perspective“-Regler) und drücken wieder „add a keyframe“. Dann wird wieder eine Zeitmarke gesetzt, die Kamera bewegt und diese Einstellung gespeichert, bis Sie am Ende der Zeitleiste angekommen sind. Schwarze Striche auf der Zeitachse indizieren die zu berechnenden Zwischenbilder.

Unten können Sie „spline“ aktivieren, um Schwenkänderungen in einer Kurve statt einem Winkelschwenk durchzuführen. „Tension“ regelt, wie weit eine Kurve ausgefahren wird. Das Rohprodukt Ihres Films können Sie anhand des Gittermodells mit dem „run“-Knopf ansehen. Gefällt Ihnen das

Ergebnis, können „echte“ Einzelbilder mit Oberflächenberechnung erzeugt werden. Die Schalter „show vect“ und „show site“ sind dafür gedacht, um gegebenenfalls Vektor- und Punktdatenkarten mit anzuzeigen. Wenn Sie nun „run and save images“ anwählen, ist ein Dateiname anzugeben (z.B. „film“) und „full rendering“ zu aktivieren (sonst wird nur das Gittermodell gespeichert). Jetzt werden die Einzelbilder erzeugt. Je nach Auflösung (auch „polygon resolution“ im „Surface“-Menü) und Bildgröße nimmt der Vorgang einige Zeit in Anspruch. Sie erhalten in diesem Beispiel 200 Dateien, die laufend nummeriert sind (film00000.rgb - film00199.rgb). Aus diesen Einzelbildern wird später ein Film erzeugt. Ganz wichtig ist es, NVIZ zu beenden, bevor Sie aus NVIZ gespeicherte Bilder ansehen oder zum Film kodieren, um Bildformatprobleme zu vermeiden.

Die Umwandlung des SGI-RGB-Formats beispielsweise in TIFF können Sie mit „xv“ durchführen oder mit den „netpbm-tools“:

```
$ sgitopnm < test.rgb | pnmtotiff > test.tif
```

Da aber ja ein Film erzeugt werden soll, benötigen Sie das Programm „mpeg_encode“[3]. Mit einer Parameterdatei sind Sie in der Lage, alle Einzelbilder (hier 200) automatisch zu konvertieren und zu kodieren. Die Parameterdatei sieht folgendermaßen aus (speichern Sie sie z.B. als „mpegparam.txt“):

```
OUTPUT meinfilm.mpg
INPUT_DIR .
INPUT
#### die Stellenzahl in [ ] muss mit den Stellen vor dem
#### Stern 5 ergeben (max. 99999 Bilder):
film00*.rgb [000-199]
END_INPUT
BASE_FILE_FORMAT PNM
GOP_SIZE        99
INPUT_CONVERT   sgitopnm *
BSEARCH_ALG     CROSS2
PSEARCH_ALG     LOGARITHMIC
PIXEL           HALF
PATTERN         IBBPBBI
IQSCALE         8
PQSCALE         10
BQSCALE         25
RANGE           10
SLICES_PER_FRAME 1
REFERENCE_FRAME ORIGINAL
```

[3] Im Internet: `ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/mpeg/encode/`

Die MPEG-Parameter können gemäß Anleitung zu „mpeg_encode“ auch variiert werden. Interessant sind nur die Zeilen „OUTPUT“ (wie soll der Film heißen?) und die Zeile zwischen „INPUT“ und „END_INPUT“ (wie heißen die Einzelbilddateien?). Die Kodierung erfolgt mit

```
$ mpeg_encode mpegparam.txt
```

Das Ergebnis können Sie mit jedem beliebigen MPEG-Abspielprogramm wie beispielsweise „xanim“ oder „mpeg_play“[4] anschauen. Die Bildgröße kann bei der Erzeugung der Einzelbilder über die NVIZ-Fenstergröße sowie über die MPEG-Parameter in „mpegparam.txt“ gesteuert werden.

10.5 Arbeiten mit weltweiten Daten: GTOPO30 und DCW

Im Internet finden sich inzwischen einige Geodatenbanken, die einen freien Zugang gestatten. Interessant sind vor allem das weltweite Höhenmodell „GTOPO30“ im 30 Bogensekundenraster (rund 1km Zellenweite) und die Vektorkarte der Welt „DCW“ – „Digital chart of the World 1:1.000.000“. Beide Datensätze liegen in der Längen-/Breitengradprojektion vor (Lat/Long, „ll“ bei GRASS). Als Ellipsoid liegt der DCW „wgs84“ zugrunde. Das weltweite Höhenmodell ist in „Kacheln“ aufgeteilt, Europa erstreckt sich über rund drei Kacheln.
Als Beispiel sollen Daten zu Österreich im Vektorformat geladen und das Land aus dem Europa-Höhenmodell extrahiert werden. Es wird also eine *location* mit Längen-/Breitengradprojektion aufgebaut, die Daten importiert und das Österreich-Höhenmodell exportiert.

Auf dem „GIS-Datadepot“-Server können Sie thematisch Karten länderweise im ARC-E00-Format bekommen:
`http://www.gisdatadepot.com/catalog/index.html`
Unter „Austria“ finden Sie „Nation wide Data“. Laden Sie unter „Admin/Political Boundaries - 1M - E00 Format“ die Karte „Political/Ocean - Network“: PONET.E00.GZ. Mit „gunzip“ können Sie die Datei entkomprimieren und mit „mv“ beispielsweise in „austria-politicalpoly.e00“ umbenennen.

Nun besorgen Sie sich die Höhendaten aus dem Internet:
`http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html`
Angegeben wird bei den Kacheln immer jeweils die Nordwestecke. Wählen Sie Europa aus und laden Sie die Datei (Kachel W020N90, einige MB groß). Das enthaltene Gebiet erstreckt sich von 20° W bis 20° Ost, von 40° N bis 90° N. Entpacken Sie das Paket in einem eigenen Unterverzeichnis mit „tar xvfz paketname.tar.gz“ (rund 100MB). Sie bekommen neben den Höhendaten auch Zusatzinformationen in weiteren Dateien und eine PNG-Datei als Voransicht. Nun sind diese Daten in GRASS zu importieren. Starten Sie GRASS und geben Sie als *location* sowie als *mapset* „europa“ ein. Die *database* sollte auf das „grassdata“-Verzeichnis in Ihrem Arbeitsverzeichnis zeigen. Mit „ESC“-„RETURN“ geht es weiter. Wählen Sie nun „latitude/longitude projection“ aus, als Ellipsoid

[4] Im Internet: `ftp://mm-ftp.cs.berkeley.edu/pub/multimedia/mpeg/play/`

„wgs84“ (Grundlage für GTOPO30). Die Regionsgrenzen ergeben sich aus dem zu verarbeitenden Gebiet. In unserem Europabeispiel gelten die oben angegebenen Werte (90N, 40S, 20W, 20E). Die „grid resolution“ beträgt gemäß Spezifikation von GTOPO30 in beiden Richtungen „00:00:30“, also 30 Bodensekunden. Mit „ESC“-„RETURN“ gelangen Sie wieder zum Eingangsbildschirm nach Überprüfung der angezeigten Werte. Da Sie für die *mapset* bereits einen Namen angegeben haben, geht es gleich mit „ESC“-„RETURN“ weiter, nach Akzeptieren der *mapset*-Erzeugung sind Sie in GRASS angekommen.

Nun werden die Höhendaten importiert. Es gibt zwei Möglichkeiten: Sie können r.in.ll oder r.in.bin benutzen. Hier wird die zweite Möglichkeit vorgestellt (in einer Zeile einzugeben):

```
$ r.in.bin -s input=W020N90.DEM output=gtopo30 bytes=2 north=90 south=40
east=20 west=-20 r=6000 c=4800
```

Der Import nimmt je nach Rechnerleistung einige Zeit in Anspruch. Da Ozeane keine Tiefendaten enthalten (diese würden Sie beim ETOPO5-Datensatz haben, der eine Auflösung von rund 10km hat), sind die Werte mit „-9999“ kodiert. In GRASS 5 können Sie sie auf NULL ändern:

```
$ r.null gtopo30 setnull=-9999
```

Öffnen Sie mit d.mon einen Monitor und betrachten Sie den importierten Europa-Datensatz:

```
$ d.rast gtopo30
```

Optional können Sie Hangneigungen berechnen, Hangexpositionen (mit r.slope.aspect), Wassereinzugsgebiete (mit r.watershed) usw. Wenn Sie ein Gebiet „zoomen“, werden die Berechnungen nur dort durchgeführt. Das dient der schnelleren Berechnung und spart Speicherplatz.

Die bereits gespeicherte Karte der politischen Grenze von Österreich kann jetzt importiert werden, da eine Datenbasis in Längen-/Breitengradprojektion vorliegt (Sie könnten diese natürlich auch ohne GTOPO30-Daten erzeugen).

Das Importmodul für ARC-E00-Dateien heißt:

```
$ m.in.e00
```

Zuerst ist der Dateiname der zu importierenden E00-Datei anzugeben: „austria-politicalpoly.e00“. Optional können Sie sagen, in welcher *mapset* die Karte gespeichert werden soll. Da nur die eine *mapset* „europa“ erzeugt worden war, kann „return“ gedrückt werden. Die Frage „What to do on input file:“ erlaubt mehrere Möglichkeiten. Gäben Sie „analyse“ ein, erhielten Sie nur eine Liste der Daten, die in der E00-Datei enthalten sind. Zum Importieren geben Sie stattdessen „vector“ ein, um die Vektordaten zu importieren. Prinzipiell können auch Rasterdaten in einer E00-Datei enthalten sein. Die Frage nach dem „Debugging level:“ können Sie mit „9“ beantworten (verbose). Damit sehen Sie den Importfortschritt und weitere Informationen. Da Sie alles mitlesen, brauchen Sie keine „log-Datei“, also bei „Name of file where log operations:“ geben Sie „return“ ein. Es erscheinen viele Meldungen und die Karte ist importiert. Überprüfen Sie den Import mit

```
$ g.list vect
```

Erzeugt wurde die Vektorkarte „ponet“ (political/ocean network). Das Modul m.in.e00 hat die Besonderheit, dass für jedes Attribut in den Vektordaten eine eigene Datei angelegt wird (da GRASS ohne externe Datenbank ja nur ein Attribut pro Karte verwalten kann). Es werden zum einen die eigentlichen Vektordaten von m.in.e00 erzeugt, zum anderen die Attributtabellen (allerdings mit <namen>.thematischeEndung). Ohne externe Datenbank (z.B. PostgreSQL) können Sie nun für jedes Thema eine eigene Vektorkarte erzeugen. Zwar sind die Vektoren dieser verschiedenen Karten immer identisch, enthalten aber jeweils unterschiedliche Attribute.
Prinzipiell wird dazu von der Vektorkarte „ponet“ also eine Kopie mit sinnvollem Namen angelegt, und die zugehörige Attributtabelle entsprechend umbenannt. Dann wird mit v.support die Topologie aufgebaut. Das macht man mehrfach für alle interessierenden Attribute, so dass entsprechend viele thematische Vektorkarten angelegt werden. Bei Anschluss einer externen Datenbank ist dieses Verfahren natürlich nicht notwendig, da dann alle Attribute an diese eine Vektorkarte „angehängt“ werden können.
Das Modul m.in.e00 hat nun also Folgendes angelegt: Eine Rohvektorkarte „ponet“ (mit g.list vect anzeigbar) und mehrere Attributtabellen, die in $LOCATION/dig_cats/ gespeichert sind. Sie stellen jetzt Kopien der Vektorinformationen her und benennen die zugehörige Attributtabelle um, damit der Zusammenhang hergestellt wird:

```
$ cd $LOCATION
$ cd dig_cats/
$ ls
```

Sie sehen (unter anderem) folgende Karten:

```
ponet.POLNTYPE   ponet.POPYCOUN   ponet.POPYTYPE
ponet.POLNSTAT   ponet.POPYADMIN  ponet.POPYREG
```

Ausführliche Erklärungen zur DCW, also zum Inhalt dieser Karten finden Sie im Internet.[5]
Nun soll die Landesgrenze als Vektorkarte aufgebaut werden:

1. Kopie erstellen von Rohvektorkarte „ponet“:

   ```
   $ g.copy vect=ponet,austria.grenze
   ```

2. Umbenennen der zugehörigen Attributtabelle:

   ```
   $ cd $LOCATION/dig_cats/
   ```
   (sofern Sie hier nicht schon sind)

   Es soll die Landesgrenzenkarte erzeugt werden („ponet.POPYCOUN“). Der Name muss identisch zur neu kopierten Vektorpolygonkarte sein, daher wird die Tabelle entsprechend umbenannt:

   ```
   $ mv ponet.POPYCOUN austria.grenze
   ```

3. Nun ist die Topologie aufzubauen:

   ```
   $ v.support m=austria.grenze option=build
   ```

[5] http://www.maproom.psu.edu/dcw/dcw_about.shtml

4. Sollen weitere thematische Karten erzeugt werden, zurück zu 2. Beispielsweise werden administrative Grenzen folgendermaßen erzeugt:

   ```
   $ mv ponet.POPYADMIN austria.admin
   $ g.copy vect=ponet,austria.admin
   $ v.support m=austria.admin opt=build
   ```

5. Wechseln Sie zurück in Ihr $HOME-Verzeichnis:

   ```
   $ cd ~
   ```

Jetzt können Sie mit der Vektorkarte (oder den Karten) arbeiten. Stellen Sie das Höhenmodell dar und darüber die Landesgrenzen von Österreich. Zur Vergrößerung (und Beschränkung des Gebiets bei Berechnungen/Datenexport) können Sie auf das Land „zoomen“:

```
$ g.region vect=austria.grenze
```

Es folgt

```
$ d.erase
```

um den GRASS-Monitor über die neuen Koordinaten zu informieren. Wie gehabt können Sie das Höhenmodell wieder anzeigen und darüber die Vektorgrenzlinienkarte (Landesgrenzen in rot):

```
$ d.vect austria.grenze col=red
```

Fragen Sie einmal testweise den Inhalt der Vektorkarte ab:

```
$ d.what.vect austria.grenze

       14:13:31.937639E  47:29:14.966592N
Line - Category $<$not tagged$>$
Area - Category 1 AU
Size - Sq Meters: 83964281488.720       Hectares: 8396428.149
           Acres: 20747521.912          Sq Miles: 32418.0030
```

Damit wird also automatisch die Landesfläche von Österreich berechnet.

Um nur mit den (Höhen-)Daten von Österreich weiterzuarbeiten, können Sie mit der Vektorgrenzlinienkarte eine Schablone (Maske) erstellen. Damit lässt sich Österreich aus den GTOPO30-Daten ausschneiden. Da Masken im Rasterformat erstellt werden, ist die Grenze in das Rasterformat zu konvertieren:

```
$ v.to.rast in=austria.grenze out=austria.grenze
```

Kontrollieren Sie das Ergebnis mit:

```
$ d.rast austria.grenze
```

Die Maske wird mit

```
$ r.mask
```

gesetzt. Unter „2 Identify a new mask“ geben Sie den Dateinamen an für „Enter name of data layer to be used for mask“: austria.grenze. Jetzt ist die Landesfläche als Maske zu aktivieren (auf 1 setzen):

```
   OLD CATEGORY NAME                                                CAT
                                                                    NUM
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .   0  0____
AU  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .   1  1____
  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .   2  0____
```

Mit „ESC“-„RETURN“ geht es weiter und mit „return“ aus r.mask heraus. Wenn Sie mit

```
$ d.rast gtopo30
```

das Höhenmodell anzeigen, sehen Sie nur noch die Höheninformationen für Österreich. Über

```
$ g.region -p
```

können Sie die Ausdehnung von Österreich abfragen (für die spätere Österreich-Datenbank in derselben Projektion). Diese Höhendaten lassen sich im ARC-ASCII-Grid-Format exportieren (die Randkoordinaten sind in der Datei enthalten):

```
$ r.out.arc m=gtopo30 > gtopo30austria.asc
```

Hier nicht vorgestellt: Es wäre denkbar, eine neue *location* (lat/long) in der Größe von Österreich zu erstellen, mit r.in.arc die Höhendaten dort zu importieren, um eine „Österreich-Datenbank“ aufzubauen. Mit „r.proj“ (vgl. Anhang A.5) ließe sich diese *location* auch in eine andere Projektion transformieren, sogar automatisiert ohne Angabe von Passpunkten (r.proj/v.proj arbeitet „nur“ für ganze *locations* und errechnet die Passpunkte aus den Gebietsecken).

Betrachten Sie einmal Österreich aus verschiedenen Blickwinkeln: Eine 3D-Betrachtung ist mit

```
$ nviz elev=gtopo30
```

möglich, wie oben vorgestellt. NVIZ berücksichtigt die gesetzte Maske, es sind also nur Höheninformationen für Österreich zu sehen. Sofern Sie beispielsweise eine Karte der Hangexpositionen berechnet haben, können Sie Sie als Farbinformation über das Höhenmodell legen. Genauso lassen sich natürlich auch die DCW-Vektordaten (mit) anzeigen.

Vergessen Sie nicht, die Maske zu entfernen (mit `$ g.list rast` können Sie immer überprüfen, ob die Datei MASK existiert). Über das Modul

```
$ r.mask
```

wird mit dem Menüpunkt 1 eine Maske entfernt, alternativ auch mit

```
$ g.remove MASK
```

Eine weitere Anregung: Errechnen Sie mit `$ r.watershed` die Tiefenlinien (streams) für Österreich. Das Modul `$ r.line` ermöglicht die automatische Vektorisierung der Linien.

Mit `$ r.sun` können Sie auch Energieeinstrahlung unter Berücksichtigung von Position, lokaler Hangneigung und -exposition sowie Trübung der Atmosphäre (Linke-Faktor) beispielsweise zur Planung von Solarkollektor-Positionierungen abschätzen.

10.6 Interpolation eines Höhenmodells aus Höhenpunkten

Dieser Abschnitt beschäftigt sich mit dem Vergleich von zwei Interpolationsalgorithmen, die zur Interpolation von Höhenmodellen aus punkthaften Höhendaten eingesetzt werden können. Sie lernen die Benutzung der entsprechenden Module kennen und sehen die qualitativ unterschiedlichen Ergebnisse. Durch Verwendung des SPEARFISH-Datensatzes ist eine Ergebniskontrolle möglich. Es werden für diese Aufgabe zufällig verteilte Höheninformationen aus dem vorhandenen Höhenmodell „elevation.dem“ erzeugt und aus diesen Höhenpunkten mit zwei Methoden wieder geschlossene Oberflächen interpoliert. Eine anschließende Differenzbildung mit vorhandenem Höhenmodell zeigt Stärken und Schwächen der jeweiligen Methoden. Hier sollen die IDW-Interpolation („nächster Nachbar“) und die RST-Methode („regularized splines with tension“) benutzt werden. GRASS bietet bekanntlich weitere Methoden, die aber hier nicht eingesetzt werden sollen.
Zunächst sind einmal zufällig verteilte Höhenpunkte aus der Karte „elevation.dem“ zu erzeugen. Als (nicht belegte) Faustregel für die Menge der zu erzeugenden Punkte diene die Formel: Anzahl= (rows*cols)/150. Mit `$ g.region -p` erhalten Sie die aktuelle Zeilen- und Spaltenanzahl.
Die Erzeugung der Karte mit den zufällig verteilten Höhenpunkten erfolgt mit (es sollen 224 Punkte in der Ergebniskarte sein):

```
$ r.random in=elevation.dem nsites=224 raster_output=hoehenpunkte
```

Nun liegt also eine Punkthöhenkarte im Rasterformat mit 224 zufällig verteilten Höhenpunkten vor. Sie können sie mit `$ d.rast` wie üblich betrachten. Informationen zur Kartenstatistik erhalten Sie mit `$ r.report` und `$ r.univar`.

Auf diese Karte sollen nun die beiden Interpolationsmethoden angewendet werden. Zuerst wird die IDW-Methode (im Rasterformat) benutzt (r.surf.idw ist nur für lat/long-*locations* einsetzbar):

```
$ r.surf.idw2 in=hoehenpunkte out=flaeche.idw
```

Die neue Karte heißt in diesem Beispiel „flaeche.idw“. Betrachten Sie das Ergebnis mit

```
$ d.rast flaeche.idw
```

Dann erfolgt die zweite Interpolation nach der Spline-Methode (RST). Dazu sind die Rasterpunkte in „Sites“ zu konvertieren:

```
$ r.to.sites -a in=hoehenpunkte out=hoehenpunkte
```

Aufgrund der unterschiedlichen GIS-Formate ergeben sich keine Probleme mit gleichen Dateinamen. Sie können diese Punktdatenkarte auch betrachten mit `$ d.sites hoehenpunkte`. Die Punkte werden als Kreuze dargestellt. Die RST-Interpolation erfolgt mit:

```
$ s.surf.rst in=hoehenpunkte elev=flaeche.rst
```

Das Modul erzeugt aus den Punkthöhen eine geschlossene Rasteroberfläche. Die neue Karte heißt „flaeche.rst“. Betrachten Sie das Ergebnis mit `$ d.rast flaeche.rst`
Nun können die Differenzen der interpolierten Karten zur „Wahrheit“ (elevation.dem) mit r.mapcalc berechnet werden:

```
$ r.mapcalc ''diff.idw=elevation.dem - flaeche.idw''
$ r.mapcalc ''diff.rst=elevation.dem - flaeche.rst''
```

Betrachten Sie die beiden Karten. Es zeigt sich vermutlich, dass ein wesentlich besseres Interpolationsergebnis mit dem RST-Algorithmus als mit dem IDW-Algorithmus erzielt wurde.
Worin unterscheiden sich nun die Ergebnisse statistisch? Die Flächenstatistik könnte folgendermaßen aussehen:

```
$ r.univar diff.idw

 Number of cells: 33231
 Minimum: -55.4745635986
 Maximum: 61.1578369141
 Range:  116.632
 Arithmetic Mean:  -0.788274
 Variance:  84.4759
 Standarddeviation: 9.19108
 Variation coefficient: -1165.98%

$ r.univar diff.rst

 Number of cells: 33231
 Minimum: -46.7217254639
 Maximum: 34.7516174316
 Range:  81.4733
 Arithmetic Mean:  -0.497715
 Variance:  40.9294
 Standarddeviation: 6.39761
 Variation coefficient: -1285.4%
```

Die Standardabweichung und Spannweite (Ergebnis zu Realität) sind bei der RST-Methode geringer. Insbesondere das Modul s.surf.rst kann über einige Parameter in der Berechnung weiter optimiert werden. Lesen Sie dazu bitte die Anleitung zum Modul und die dort angegebenen Fachaufsätze.
Üblicherweise können Sie derartige Tests natürlich nicht durchführen, da Sie dann ja nicht interpolieren müssten. Jedoch ist es im GIS immer sinnvoll, die Qualität einzelner Rechenschritte anhand von Testdaten zu testen und mit bekannten Ergebnissen zu vergleichen.
Nun können Sie sich abschließend alle Flächenkarten in einer kleinen Diashow anzeigen lassen:

```
$ slide.show.sh prefix=flaeche across=2 down=3
```

Geben Sie keine Parameter an, werden alle vorhandenen Rasterkarten dargestellt.

10.7 Reliefanalysen mit GRASS

10.7.1 Erzeugung synthetischer Höhenmodelle

Häufig kommt es vor, dass für bestimmte Fragestellungen keine Testdaten in gewünschter Auflösung vorliegen. GRASS bietet im Hinblick auf Reliefanalysen ein Modul zur Erzeugung synthetischer Höhenmodelle wählbarer Ausprägung (vgl. WOOD 1995). Die Höhenmodelle werden über die Angabe einer fraktalen Dimension erzeugt (vgl. MANDELBROT 1987). Diese Dimension D liegt zwischen den euklidischen Dimensionen 2 (Ebene) und 3 (Volumen). Je näher der Wert für D an 3 heranreicht, desto feinstrukturierter ist das erzeugte Reflief. Als Koordinatengrundlage und Auflösung werden immer die aktuellen Einstellungen genutzt (verändern mit g.region):

```
$ r.surf.fractal out=fractdem.201 d=2.01
```

Über `$ r.univar fractdem.201` wird Ihnen Minimum und Maximum angezeigt, mit r.mapcalc können Sie optional beide Werte durch Addition bzw. Multiplikation so anpassen, dass das Minimum z.B. bei Normalnull liegt und das Maximum einen bestimmten Wert nicht überschreitet.
Mit

```
$ r.slope.aspect el=fract.dem as=aspect.fract
```

wird eine Expositionskarte erzeugt, die Ihnen einen anschaulichen Eindruck vom (synthetischen) Gebiet gibt (vgl. Abb. 29). Ein feiner strukturiertes Höhenmodell erhalten Sie, wenn Sie die fraktale Dimension D erhöhen:

```
$ r.surf.fractal out=fractdem.25 d=2.5
```

Diese synthetischen Höhenmodelle lassen sich weiter analysieren. Beispielsweise können Sie die hydrologischen Einzugsgebiete berechnen:

```
$ r.watershed el=fractdem.2001 out=ezgkarte thres=10000
```

Eine geschummerte Karte der Einzugsgebiete (unter Verwendung der Hangexposition) kann auf einfache Weise mit `$ d.his` IHS-Transformation) hergestellt werden, das Ergebnis lässt sich optional speichern:

```
$ d.his i=aspect.fract h=ezgkarte [out=ezg.schummer]
```

10.7.2 Geomorphologische Untersuchungen

Neben den üblichen Hangneigungs- und Expositionsberechnungen mit `$ r.slope.aspect` können Sie in GRASS weitergehende Analysen durchführen. Ein Beleuchtungsmodell mit definiertem Sonnenstand steht mit dem Script `$ shade.rel.sh` zur Verfügung.
Speziellere Reliefuntersuchungen bei Auswertung eines Höhenmodells ermöglicht das Modul

```
$ r.param.scale
```

Das „moving window“ kann größer als die typische 3x3-Matrix eingestellt werden. Mit r.param.scale lassen sich folgende Werte berechnen (Generalisierung abhängig von „size“):

- elev: Matrixweit generalisierte Höhenwerte
- slope: Maximaler Gradient an einem Punkt

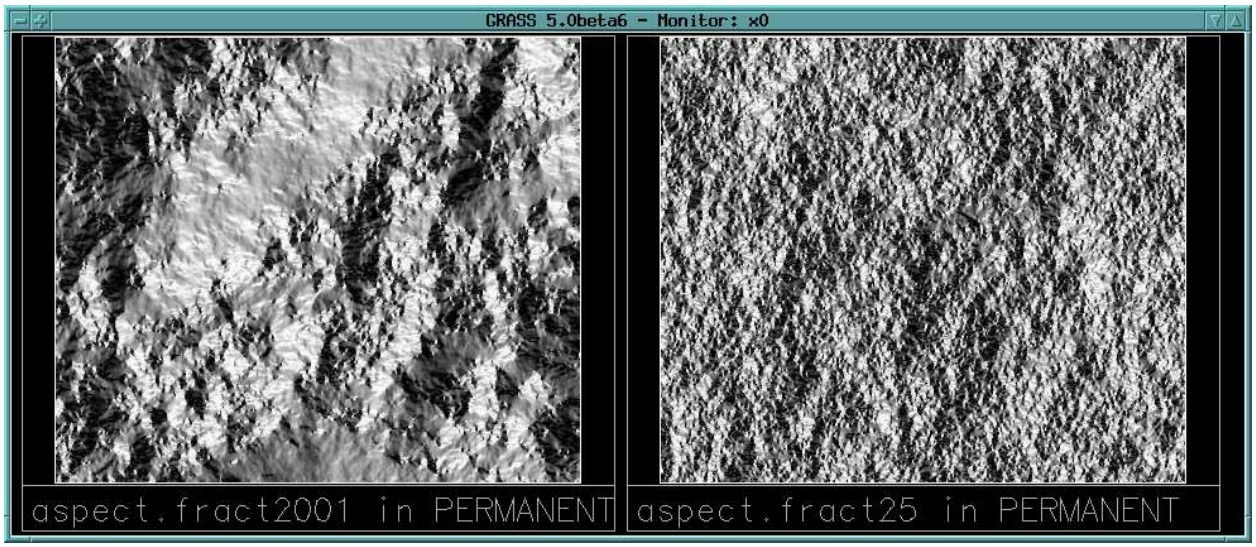


Abbildung 29: Erstellung synthetischer Höhenmodelle mit r.surf.fractal (links: fraktale Dimension D=2.001, rechts: D=2.5, Ansicht der Hangexposition)

- aspect: Richtung des maximalen Gradienten
- profc: Vertikalkrümmung (in Richtung des stärksten Gefälles)
- planc: Horizontalkrümmung
- crosc: Cross sectional convexity (tangent to contours, downslope).
- longc: Longitudinal convexity (perpendicular to contours downslope).
- minic: Minimale Krümmung im „moving window“
- maxic: Maximale Krümmung im „moving window“
- feature: peaks (Gipfel), ridges (Rücken), passes (Sattel), channels (Tiefenlinien), pits (abflusslose Senken) und planes (Ebenen)

Über „size“ wird die Größe des Suchfensters gesteuert, je größer die Matrix ist, desto mehr „weißer Rand“ ist an den Kartenrändern zu finden. Zu begründen ist dieses Resultat damit, dass nur bei vollständig gefülltem „moving window“ Ergebnisse berechnet werden können. Ein Hinweis: Mit `$ d.rast.num` können Sie sich die Rasterzellenwerte als Zahlen für kleiner Ausschnitte darstellen lassen.
Eine interaktive Möglichkeit, obige Parameter sowie „Moran“-, „Geary“-Statistik und quadratische Residuen mit der Maus im Bereich des eingestellten Suchfensters abzufragen, besteht mit
`$ d.param.scale`

Alternative Möglichkeiten, Profilkrümmungen zu berechnen, bestehen mit
`$ s.surf.rst` und `$ r.slope.aspect`.

„Echte“ Oberflächen lassen sich mit

```
$ r.surf.area
```

ermitteln. Es wird die projizierte ebene Fläche angegeben („Current Region plan area“) und die geschätzte dreidimensionale Oberfläche („Estimated Region Surface Area“).
Volumina können ebenfalls berechnet werden als Volumen zwischen den Höhenwerten und Normal Null. Das Modul heißt:

```
$ r.volume
```

Um beispielsweise das Volumen eines Kiesteichs zu ermitteln, ist zunächst die Teichumgebung zu maskieren. Ganz einfach kann dafür mit r.digit die Teichfläche vektorisiert und als Attribut „1“ vergeben werden. In r.mask wird dann entsprechend der Teich-Kategorie ebenfalls „1“ zugewiesen. Nun ist also nur noch der Teich sichtbar. Mit r.univar kann das Minimum ermittelt werden und mit r.mapcalc vom Höhenmodell subtrahiert werden, um das Volumen auf NN zu normieren (hier sei 1296m die tiefste Stelle im Teich):

```
$ r.mapcalc ''teich=elevation.dem - 1296''
```

Das Volumen im Teich wird nun über

```
$ r.volume teich
```

errechnet. Vergessen Sie nicht, die Maske wieder zu entfernen.

Rücken (ridges) im Gebiet können Sie alternativ zu r.param.scale auch mit r.flow ermitteln:

```
$ r.flow -u el=elevation.dem as=aspect dsout=density
```

Es ist sinnvoll, eine Filterung durchzuführen, um diffus verteilte Zellen, die nicht als Rücken akzeptabel sind, zu eliminieren (der Wert „20“ ist gegebenenfalls anzupassen):

```
$ r.mapcalc ''density.filt=if(density>20,1,0)''
```

Zum Schluss werden die Rücken noch „verdünnt“, um eine Maximalbreite von einem Pixel zu bekommen:

```
$ r.thin in=density.filt out=ridges
```

Um eine Wasserscheide als Grenzlinie um ein Einzugsgebiet zu erzeugen, sind mehrere Schritte notwendig. Zuerst werden die Einzugsgebiete bestimmter Größe ermittelt (threshold gibt die Mindest-Zellenzahl an):

```
$ r.watershed elev=elevation.dem basin=ezg thresh=10000 stream=fluesse
```

Nun soll beispielsweise das Einzugsgebiet Nr. 18 bearbeitet und dafür die Grenzlinie erzeugt werden. Zuerst ist in eine eigene Karte zu extrahieren:

```
$ r.mapcalc ''flaeche=if(ezg==18,1,null())''
```

Die Grenzlinie wird hergestellt, indem man diese Fläche um eine Zellenweite in alle Richtungen wachsen lässt und dann die kleinere Ausgangsfläche subtrahiert:

```
$ r.grow in=flaeche out=flaeche2
$ r.null null=0 m=flaeche
$ r.null null=0 m=flaeche2
```

```
$ r.mapcalc ''rand=flaeche2-flaeche''
```

Die Karte „rand“ enthält nun also die Umschließungsgrenze. Soll der Innenrand auch erzeugt werden, ist wieder r.grow einzusetzen:

```
$ r.grow in=rand out=rand.innen
```

Von dieser „dickeren“ Grenzlinie werden nun alle Außengebiete subtrahiert. Um NULL-Werte zu erhalten, wird eine umschließende if-Bedingung zusätzlich gebraucht:

```
$ r.mapcalc ''rand.innen1=if(rand.innen - if(ezg!=18,1,0),1,null())''
```

Bei der Reliefanalyse beispielweise im Hinblick auf Erosionsmodellierung kann es sinnvoll sein, ein Höhenmodell ohne lokale Senken zu bekommen. GRASS bietet einen Algorithmus (von JENSON & DOMINGUE 1988, vgl. Modulanleitung) für diesen Zweck an, um diese abflusslosen Senken zu füllen:

```
$ r.fill.dir
```

Zusätzlich wird eine Karte der Entwässerungsrichtung erzeugt, die in jeder Rasterzelle den entsprechenden Winkel kodiert hat. Ein Beispielaufruf sieht so aus:

```
$ r.fill.dir in=elevation.dem elev=elevation.fill dir=flow.dir typ=grass
```

Eine andere Auswertungsmöglichkeit für Reliefmodelle bietet die Sichtbarkeitsanalyse. Für einen anzugebenden Standort und die „Augenhöhe“ wird ringsrum unter Berücksichtigung des Reliefs berechnet, wie weit man von hier aus sehen kann. Die Ergebniskarte kodiert zellenweise den jeweiligen Blickwinkel (in Grad) auf diese Zelle. Eine Maximalreichweite kann angegeben werden, eine optionale Binärkarte die z.T. zeitaufwändige Rechnung auf Interessensflächen beschränken. Ein typischer Modulaufruf könnte sich dementsprechend ergeben als:

```
$ r.los in=elevation.dem out=sichtbarkeit coord=598046,4920205
```

Um verschiedene Standorte zu berücksichtigen, können Sie später die Einzelergebnisse mit r.mapcalc, r.combine, r.weight oder auch r.patch je nach Zielsetzung überlagern oder verschneiden. Einsatzgebiete sind die Planung von Windkraftanlagen, Mobilfunksendern usw. Allerdings wird hier von homogener Atmosphäre und „flacher“ Erde ausgegangen.

Die in vielerlei Anwendung wichtige Erzeugung von Profilen lässt sich in GRASS auf verschiedenen Wegen erreichen:

- `$ d.profile`: Profil über eine Strecke, die mit der Maus im GRASS-Monitor angegeben wird
- `$ r.profile`: Profil mit Werteausgabe entlang eines Weges (Angabe von Start-, Zwischen- und Endkoordinaten), optional Median-/Mittelwertbildung über die Rasterzellen
- `$ r.transect`: Profil mit Werteausgabe entlang einer Richtung (Angabe von Startkoordinaten, Winkel und Streckenlänge), optional Median-/Mittelwertbildung über die Rasterzellen

Diese „lockere Sammlung“ von Modulvorschlägen soll Ihnen als Anregung für eigene Arbeiten dienen. Im Folgenden stehen abschließend für dieses Kapitel stichpunkthaft die Ermittlung hydrologischer Kenngrößen und weiterer dort benötigter Parameter im Vordergrund.

10.8 Stichpunkte zur Erfassung hydrologischer Parameter in der Modellierung

- Ermittlung von Wasserscheiden/Einzugsgebiete:

  Berechnung mit `$ r.watershed`, die Mindestgrößen können mit „threshold“ eingestellt werden. „threshold“ ist als Rasterzellenzahl anzugeben (die Mindestfläche berechnet sich entsprechend aus Rasterzellenlänge $*$ -breite $*$ -anzahl, aktuelle Einstellungen zur Auflösung mit g.region abfragen). Die optional verlangte „depression map“ kann mit `$ r.basins.fill` ermittelt werden. Auch sind Halbeinzugsgebiete (half basins) ermittelbar (Nummerierung der rechten Seiten (stromaufwärts) um 1 größer als die der linken Seiten). Es können auch Abflussblockadekarten berücksichtigt werden. Das Modul berechnet etliche weitere Parameter in Kartenform (vgl. Modulanleitung).

  Gräben lassen sich berücksichtigen, indem sie in das Höhenmodell eingebaut werden. Dazu Gräben vektorisieren, Tiefe als Attribut (z.B. in Zentimetern „60“) vergeben, mit `$ g.region` Rasterauflösung einstellen. Über `$ v.to.rast` in Rasterzellen umwandeln, gegebenenfalls `$ r.buffer` zum Verbreitern der Rasterlinie einsetzen, zum Verdünnen an Ecken `$ r.thin` benutzen. Dann Höhenmodell um Gräben korrigiert: Gräben vom Höhenmodell subtrahieren, dabei Grabentiefe in Meter umrechnen, sofern Höhenmodell in Höhenmetern vorliegt. Beispiel:

  ```
  $ r.mapcalc ''dgm.neu = dgm - (graeben * 0.01)''
  ```

  Die Einzugsgebietsberechnung kann beispielsweise so erfolgen:

  ```
  $ r.watershed el=elevation.dem basin=basin thresh=1000 stream=fluesse
  ```

  Berechnete Flüsse (streams) tragen als Kodierung die Einzugsgebietsnummer, die wiederum nach Stromgrößen sortiert sind.

- Berechnung von Flächeninhalt bzw. Einzugsgebietsgröße sowie Einzugsgebietsumfang:

  Rasterbasiert kann die mit `$ r.watershed` berechnete Flächenkarte in `$ r.report` auf Flächenanteile untersucht werden. Alternativ lässt sie sich in das Vektorformat konvertieren mit `$ r.poly`, die Flächen- und Umfangsberechnung erfolgt dann mit `$ v.area` oder `$ v.report`.

- Ermittlung des Gebietsschwerpunkts:

  Vgl. Script in Abschnitt 13.1.

- Berechnung von Fließwegen:
  Hier gibt es verschiedene Möglichkeiten: mit `$ r.flow` (Fließwege als Vektoren betrachtet, MITASOVA ET AL. 1993, MITASOVA ET AL. 1996), mit `$ r.watershed` (D8-Algorithmus) oder `$ r.flowmd` (basierend auf „360° Vektorfließrichtungen, multiple directions“, anders als „multiple flow“ bei DESMET & GOVERS 1996).

- Ermittlung der Länge eines Fließwegs (Quelle-Pegel):

  Eine einfache, aber etwas ungenaue Methode ist folgende: Über `$ r.clump` erhalten zusammenhängende Flächen jeweils ein einheitliches Attribut, mit `$ r.thin` werden die Ströme auf eine Rasterbreite reduziert. Die Länge erhält man über die kombinierte Anwendung von `$ r.stats -c` mit `$ awk` und `$ r.reclass` (vgl. Abschnitt 6.17, dort als Flächenzuweisung vorgestellt). Da die Ströme nur eine Rasterzelle breit sind, ergibt sich die Länge aus der Zellenanzahl. Hier sollte eine „Kurvenkorrektur“ ($\sqrt{2}$) berücksichtigt werden.

- Höhendifferenz zwischen dem höchsten Punkt im Gebiet und einem Punkt am Fließweg (z.B. Pegel):

  Das Maximum im Gebiet wird wie üblich zuerst statistisch ermittelt mit:

  `$ r.univar elevation.dem` oder `$ r.stats -1 elevation.dem`

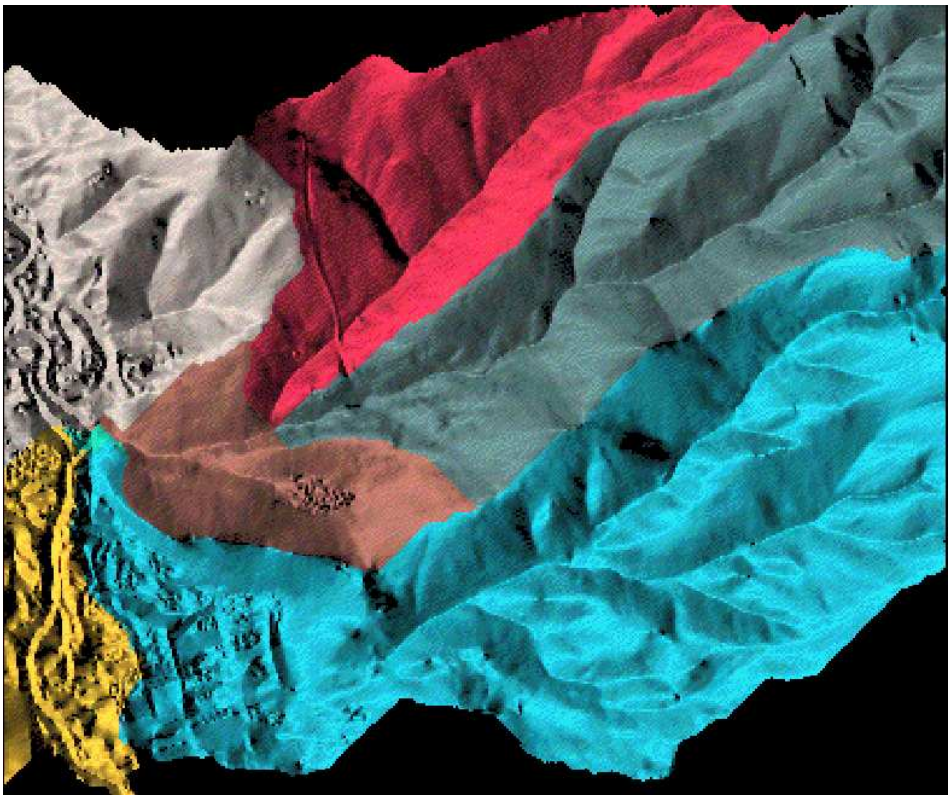


Abbildung 30: Ermittlung von Einzugsgebieten mit r.watershed (Schummerungsdarstellung mit Expositionskarte durch d.his)

Sie können dann das Maximum auch als Karte erzeugen (mit nur einem Wert in der ganzen Karte):

```
$ r.mapcalc ''max=if(elevation.dem==1840,elevation.dem,null())''
```

und schließlich die Koordinaten des Maximums (hier das werttragende Pixel) sich anzeigen lassen:

```
$ r.stats -1gn max
```

Einen Höhenpunkt (Pixel) kann man generell abfragen über `$ r.transect`:

```
$ r.transect map=elevation.dem line=591701,4917097,0,0
```

- Länge zwischen dem höchsten Punkt im Gebiet und einem Punkt am Fließweg (z.B. Pegel):

  Zuerst erzeugen Sie einen „Fließweg“ zwischen den Punkten:

  ```
  $ r.drain in=elevation.dem out=fliessweg coord=591701,4917097,593308,4925091
  ```

  Die Rasterlinie kann in eine Vektorlinie konvertiert werden:

  ```
  $ r.line in=fliessweg out=fliessweg
  ```

  Anschließend ist die Topologie aufzubauen:

  ```
  $ v.support m=fliessweg o=build
  ```

  Die Länge wird mit v.report ermittelt (z.B. in Kilometern):

  ```
  $ v.report m=fliessweg type=line unit=ki
  ```

  Alternativ können Sie beide Punkte digitalisieren (`$ v.digit` oder `$ r.digit`) und mit

  `$ r.distance` die Luftlinie berechnen lassen.

- Berechnung des aktuellen Fließgefälles und des mittleren Fließgefälles:

  Zuerst wird eine Maske erzeugt für die Pixel, an denen der Fließweg vorhanden ist (z.B. Stromnummer 64):

  ```
  $ r.mapcalc ''MASK=if(streams==64)''
  ```

  Nun werden nur noch Hangneigungen für Fließweg angezeigt bzw. analysiert:

  ```
  $ d.rast slope
  ```

  Berechnung der mittleren Hangneigung für den Fließweg:

  ```
  $ r.univar slope
  ```

  Speichern der aktuellen Hangneigungen nur für den Fließweg in einer neuen Karte:

  ```
  $ r.mapcalc ''slope.strom64=slope''
  ```

  Zum Schluss sollte die Maske wieder entfernt werden:

  ```
  $ g.remove MASK
  ```

- Bestimmung der mittlere Gebietshöhe:

  Setzen einer Maske für das interessierende Einzugsgebiet:

  ```
  $ r.mask basin
  ```

  Berechnung des Höhenmittelwertes:

  ```
  $ r.univar dem
  ```

- Ermittlung der mittleren und maximalen Geländeneigung:

  Berechnung der Hangneigungen mit `$ r.slope.aspect`, Bestimmung von Mittelwert und Maximum über `$ r.univar`

- Errechnung von Flächenanteilen mit bestimmter Exposition oder Neigung:

  Erstellung einer Maske, die den Bereich herausfiltert:

  ```
  $ r.mapcalc ''MASK=(if((aspect>40) && (aspect < 60))''
  ```

  Berechnung der Fläche für diesen Bereich mit `$ r.surf.area`

11 Satellitenbildverarbeitung

Die Satellitenbildverarbeitung ermöglicht die Lösung vielfältiger geographischer Fragestellungen wie Vegetationsmonitoring, Landnutzungsermittlung und Kartenaktualisierungen. Die Verfügbarkeit der Daten steigt und die wissenschaftlichen Methoden der Auswertung haben sich in den letzten Jahren stark entwickelt. Sehr hilfreiche Literatur in Bezug auf GRASS und Bildverarbeitung ist das „GRASS tutorial: Image processing“ von HARMON/SHAPIRO 1992, in dem viele Verfahren erläutert werden. Bei LÖFFLER 1994, JENSEN 1996 und SCHOWENGERDT 1998 finden sich umfangreiche Erklärungen.

Interessant sind v.a. Daten der Satelliten LANDSAT-TM5 und SPOT (Spektrum: visuell + infrarot) sowie ERS-1 und ERS-2 (Spektrum: Radar). Neu hinzugekommen ist jetzt der indische Satellit IRC-1C, der ein optisches Spektrum (in 4 Kanälen ähnlich zu SPOT) aufnimmt.

Während LANDSAT-TM5- und ERS-1/2-Daten eine Auflösung von 30m $*$ 30m haben (teilweise werden sie vom Hersteller auf 25m $*$ 25m bzw. 10m $*$ 10m hochgerechnet), beträgt sie bei SPOT 20m $*$ 20m (Farbkanäle) bzw. 10m $*$ 10m (panchromatisch). Die Bedeutung der LANDSAT-TM-Kanäle ist in Abschnitt A.8.3 beschrieben. Diese Bilder lassen sich sinnvoll für Fragestellungen in einem Maßstab von 1:200.000 bis 1:100.000 einsetzen. Einen zusätzlichen panchromatischen Kanal mit 15m Pixelauflösung für größere Maßstäbe hat der LANDSAT-TM7, der sich seit dem Frühjahr 1999 erfolgreich im Orbit befindet. Eine höhere Auflösung bietet der indische Satellit IRS-1C/D beim panchromatischen Kanal mit 5.80m pro Pixel, während die Auflösung seiner Farbkanäle bei 23.5m $*$ 23.5m liegt. Auf den Markt drängen seit kurzem kommerzielle Satellitensysteme im Meterbereich, allerdings zu Lasten der spektralen Auflösung.

Ein Hinweis: Die Verarbeitung und Analyse von Satellitendaten verhält sich in GRASS generell wie bei Rasterdaten. Die internen Formate sind identisch (Ausnahme: fouriertransformierte Daten), daher können prinzipiell alle Rasterkommandos und Bildverarbeitungskommandos gemischt verwendet werden. Nur der Übersichtlichkeit halber wurden die Bildverarbeitungsmodule mit einem anderen Kennbuchstaben („i.$*$“) gegenüber den Rasterkommandos („r.$*$“) versehen.

11.1 Geometrische Vorverarbeitung von multispektralen Satellitendaten

Üblicherweise werden Satellitendaten im UTM-Koordinatensystem auf Datenbändern geliefert. Das in Deutschland übliche Gauß-Krüger-System weicht projektionsmäßig in Projektionsellipsoid und Datum davon ab, so dass eine Entzerrung der Satellitenbilder auf diese Projektion erforderlich

wird. Dieses Kapitel beschreibt die Vorgehensweisen anhand von LANDSAT-TM-Daten, prinzipiell sind sie auf alle multispektralen Fernerkundungsplattformen übertragbar.
Hinter der benötigten Koordinatentransformation auf das Landeskoordinatensystem steckt folgender Ablauf: Die im UTM-System vorliegenden LANDSAT-Daten werden in eine xy-*location* eingelesen und dann auf eine neue *location*, die im Gauß-Krüger-System angelegt wird, entzerrt. Diese Transformation wird mit einer Affin-Transformation durchgeführt (vgl. Abb. 31 und Erläuterungen in Abschnitt 6.6), die als Parameter die vier Ecken der Satellitenbild-Szene bekommt. Anschließend liegt die Szene geokodiert im Gauß-Krüger-System vor, die xy-*location* kann gelöscht werden.
Eine Vorstellung der benötigten Rechenzeit liefert der folgende Vergleich: Eine LANDSAT-TM-Szene benötigt rund $7 * 76$MB Speicherkapazität (sieben Kanäle), gut 530MB insgesamt. Zur Transformation müssen also 1,06GB Festplattenplatz zur Verfügung stehen, sofern die gesamte Szene in einem Arbeitsgang transformiert werden soll. Steht weniger Platz bereit, kann in bis zu sieben Arbeitsgängen auch nur einer oder mehrere Kanäle pro Arbeitsgang bearbeitet werden. Pro Kanal dauert die Berechnung auf einer SUN-Workstation SPARC/25 zwei Stunden, insgesamt also gut 14 Stunden für die gesamte Szene, Linux-PCs sind inzwischen schneller.
Alternativ kann natürlich (bei entsprechender Koordinatenumrechnung) nur ein Ausschnitt in eine kleinere Gauß-Krüger-*location* transformiert werden.

Zunächst sind also zwei *locations* einzurichten:

- eine xy-*location* für die LANDSAT-Originaldaten
- eine Gauß-Krüger-*location*, die Ziel-*location*

Die Vorgehensweise wird im folgenden Abschnitt beschrieben.

11.1.1 Import von Satellitendaten

Um eine Satellitenbildszene importieren zu können, müssen die Daten in einem von GRASS unterstützten Format vorliegen. Es wird zwischen zwei Formattypen unterschieden:

- Formate, in denen nur ein Kanal pro Datei enthalten ist;
- Formate, in denen mehrere Kanäle pro Datei enthalten sein können.

Zur ersten Gruppe gehören folgende Formate: das SUN-Raster-Format, das TIFF- und das GIF-Format. Mehrere Kanäle pro Datei können im BIL- bzw. BSQ-Format, im ERDAS/LAN- und im NCSA/HDF-Format abgelegt werden.

Im Folgenden werden einige Importmöglichkeiten von Daten in GRASS vorgestellt.
Liegt das Satellitenbild kanalweise ohne Projektionsinformationen vor, wird in GRASS in eine xy-*location* importiert. Ansonsten wird ein Projektgebiet entsprechend der Randkoordinaten, Auflösung und Projektion eingerichtet.

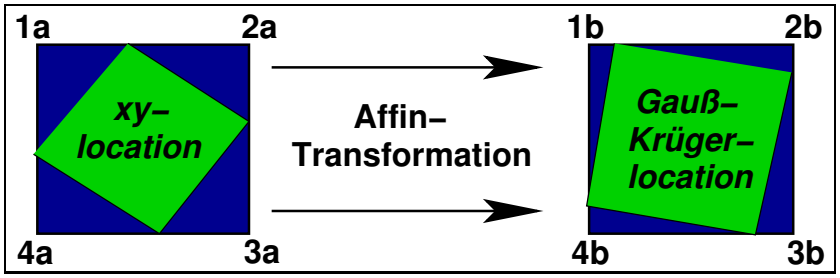


Abbildung 31: Affin-Transformation eines Satellitenbildes

Hier nun beispielhaft die Einrichtung einer xy-*location* für Satellitenrohdaten:

(a) Dazu gibt man beim Erscheinen des Begrüßungsbildschirms von GRASS einen neuen Namen für die *location* und die *mapset* an.

(b) Als Koordinatensystem wird „xy“ gewählt.

(c) Die Werte x- und y-Ausdehnung (Zeilen und Spalten) des Satellitenbildes kann man dem Header der Originaldaten entnehmen (siehe oben) oder bei kleinen Ausschnitten im Megabyte-Bereich z.B. mit dem Bildbetrachtungsprogramm `$ xv` in Erfahrung bringen (in xv Menüpunkt „Windows“ - „Image Info“).

(d) Das Koordinatenformular wird wie folgt ausgefüllt: Als Nord- und Ostwert sind immer die Zeilen- und Spaltenanzahl der Dateien anzugeben, die West- und Südwerte sind auf Null zu setzen (vgl. Abb. 32). Eine xy-*location* darf auch größer als das zu importierende Bild sein.

```
-------------------------------------------------------
|                 North: y-Wert                       |
|                        (aus Datei)                  |
|                                                     |
| West: 0                              East: x-Wert   |
|       (festgelegt)                          (aus Datei) |
|                                                     |
|                 South: 0                            |
|                        (festgelegt)                 |
-------------------------------------------------------
     Resolution: East-West:   1   (festgelegt)
                 North-South: 1   (festgelegt)
```

Abbildung 32: Berechnungen im Koordinatenformular zum Import in xy-*locations*

Damit ist die *location* eingerichtet. Die *mapset* wird nach dem erneuten Erreichen des Begrüßungsbildschirms von GRASS erzeugt.

11.1.1.1 Import von Daten im TIFF- oder SUN-Raster-Format

Mit `$ r.in.tiff` oder `$ r.in.sunrast` lässt sich eine Satellitenbild-Szene kanalweise importieren. Beachten Sie, dass beide Formate keine Koordinateninformationen beinhalten, Sie damit ohne weiteres in eine xy-*location* importieren können. Wollen Sie dagegen die Daten in eine *location* mit Projektionsangaben (also bereits geocodiert) importieren, sind nach dem Import mit `$ r.support` (Option: „Edit header“: yes) die Randkoordinaten der einzelnen Bildkanäle manuell einzugeben.

11.1.1.2 Import von Daten im ERDAS/LAN-Format

Das LAN-Format wird in GRASS ab der Version LAN7.4 oder neuer (in 4-bit, 8-bit oder 16-bit) unterstützt. Eine LAN-Datei kann in GRASS 4.x bis zu 7 Kanäle beinhalten, ab GRASS 5.0.x ist die Kanalanzahl unbeschränkt.
Das entsprechende GRASS-Modul zum Import dieser Daten wird folgendermaßen aufgerufen:
`$ i.in.erdas`
Das Modul importiert alle Kanäle (mit fortlaufender Nummerierung) oder kann selektiv kanalweise importieren. Auch lässt sich nur der „Header“, also die Dateiinformationen, anzeigen (Kanalanzahl, Kanalgrößen etc.), ohne die Daten zu importieren. Dabei ist trotzdem ein Namenspräfix für die Kanäle anzugeben.
Für den Export mehrerer Kanäle in eine LAN-Datei gibt es das korrespondierende Module
`$ i.out.erdas`

11.1.1.3 Import von Daten im HDF-Format

Recht verbreitet ist auch das HDF-Format für mehrkanalige Satellitendaten. Daten dieses Typs können mit
`$ r.in.gdal`
importiert werden. Die einzelnen Kanäle werden automatisch aus der HDF-Datei extrahiert und mit einer Nummern-Endung im Dateinamen in der GRASS-Datenbank fortlaufend nummeriert.

11.1.1.4 Import von Daten im BIL-/BSQ-Format

GRASS bietet auch die Möglichkeit, Daten direkt vom Magnetband, von CD-ROM und von Diskette einzulesen. Es gibt zwei Standardspeicherverfahren: das BIL- (band interleave) und das BSQ- (band sequential) Format. Das Einlesen von BIL- und BSQ-Daten ist nur in eine xy-*location* möglich. Die Lesegeräte unter UNIX (also auch unter Linux) heißen „devices“ und sind im Verzeichnis
`/dev/`
abgelegt. Also stehen beispielsweise

`/dev/cdrom` für das CD-ROM-Laufwerk

`/dev/rmt0` für das Tape-Laufwerk und

`/dev/fd0` für das erste Diskettenlaufwerk.

Diese Angaben sind systemspezifisch. Wichtig ist, das Medium vorher zu „mounten“ (vgl. Abschnitt 2.2.5), eine CD-ROM beispielsweise mit:

```
$ mount /dev/cdrom /cdrom
```

Um ein Medium im Hinblick auf seine Datenparameter zu untersuchen, können Sie das Modul

```
$ m.examine.tape
```

benutzen. Zunächst muss das „device“ angegeben werden, weitere Fragen können Sie mit „return“ quittieren. Mit etwas Glück (da GIS-Formate von den Herstellern häufig geändert werden) erhalten Sie die gewünschten Informationen. Allerdings sollten diese normalerweise vom Hersteller der Daten mitgeliefert werden.

Das Modul

```
$ i.tape.other
```

erlaubt nun das Einlesen der Daten. Als erstes muss auch hier das „device“ angegeben werden, die im folgenden angegebene Puffergröße können Sie i.a. übernehmen. Die nachfolgende Eingabemaske kann frei beschrieben werden: Tape-Nr., kurze Beschreibung und Titel für die zukünftigen GRASS-Rasterbilder. Anschließend geht es bandspezifisch weiter (vgl. Abb. 33):

Im gegebenen Beispiel enthält die BIL-Datei vier Kanäle. Die Zeilenlänge beträgt 512 Bytes, also 512 Pixel in x-Richtung der Bilder. Es werden hier alle Kanäle ausgepackt, „skippen“ (= Überspringen einzelner Kanäle) ist möglich. Eventuell müssen die Parametern etwas modifiziert werden, um ein Ergebnis zu erreichen.

Im Folgeschritt ist anzugeben, welche Kanäle extrahiert werden und wie die Bildszene unter GRASS heißen sollen. Für jeden Kanal wird automatisch eine Nummer angehängt. Zum Schluss kann man noch einen Ausschnitt oder alles selektieren: Entweder werden die gesamten Bilder ausgepackt oder nur Teilflächen. Das kann bei großen Szenen interessant sein (man muss allerdings die Koordinaten kennen). Üblicherweise wird von 1 bis x-max und 1 bis y-max ausgepackt (z.B.: start row: 1, end row: 512, start col: 1, end col: 512).

Damit wird die Importierung durchgeführt und ist abgeschlossen. Es sollte noch

```
$ r.support
```

zur Berechnung der Datenstatistik auf jeden Kanal angewendet werden („Edit header“: no, „update stats...“: yes, weitere Fragen: „return“).

Wollen Sie BIL-Daten direkt von der Festplatte einlesen, gibt es das Script `$ r.in.bil` für diesen Zweck. In diesem Fall benötigen Sie kein „device“.

```
-----------------------------------------------------------------------
|GENERIC TAPE EXTRACTION                                              |
|                                                                     |
|tape layout                                                          |
|                                                                     |
| 0___    number of tape files to be skipped                          |
| 0___    number of records in the remaining files to be skipped      |
| 0___    number of bytes in each record to be skipped                |
|                                                                     |
|band files                                                           |
|                                                                     |
| 4___    number of bands on the tape                                 |
|                                                                     |
|data format                                                          |
| _       band sequential  (BSQ)  |   mark one with an x              |
| x       band interleaved (BIL)  |                                   |
|                                                                     |
| 0____   if you select BSQ format and all the bands are in a single|
|         file enter the total number of records in the file. Other-|
|         wise enter 0                                                |
|                                                                     |
| 512__   length (in bytes) of the longest record on the tape         |
| 1___    blocking factor of data in the file                         |
|     AFTER COMPLETING ALL ANSWERS, HIT <ESC><RETURN> TO CONTINUE     |
|                       (OR <Ctrl-C> TO CANCEL)                       |
-----------------------------------------------------------------------
```

Abbildung 33: BIL-/BSQ-Formular zur Datenbandspezifikation in i.tape.other

11.1.2 Koordinatentransformation UTM nach Gauß-Krüger

Nachdem der Import der Satellitenbild-Szene abgeschlossen ist, kann mit den Vorbereitungen für die Transformation in das Gauß-Krüger-Koordinatensystem (oder natürlich in ein anderes System) begonnen werden. Die Szene ist zwar in eine xy-*location* oder UTM-*location* importiert worden, besitzt aber noch keine Landeskoordinaten.

Die UTM-Koordinaten einer Beispiel-Szene (Hannover) stellen sich so dar (die Informationen stammen aus dem Header der Szene, Abkürzungen: z.B. fcartoul= frame-carto., upper left):

y = 8496	xpxlsize	= 25.00
x = 8976	ypxlsize	= 25.00

UTM:	R	H
fcartoul	= 32427882	5833098
fcartour	= 32652226	5833098
fcartolr	= 32652226	5620763
fcartoll	= 32427882	5620763

Die Entzerrung auf die Gauß-Krüger-Projektion leistet die Affin-Transformation (mit i.rectify, Näheres s. Abschnitt 6.6). Dazu gibt man die vier Eckpunkte des Bildrahmens (frame-carto), nicht aber die des eigentlichen Satellitenbilds (scene-carto) an. Der schwarze, informationslose Rand wird also mittransformiert, da nur er eine Nord-Südausrichtung aufweist. Das eigentliche Satellitenbild liegt, durch die Flugbahn des LANDSAT bedingt, schräg. Da die Gauß-Krüger-Koordinaten im LANDSAT-Header nicht mit angegeben sind, muss man zunächst die UTM-Koordinaten umrechnen.[1]

Das oben genannte Beispiel ergibt für die umgerechneten Gauß-Krüger-Koordinaten:

GK-Rechtswert	GK-Hochwert
3427861.97	5834831.02
3652268.28	5834831.01
3652268.33	5622435.34
3427861.94	5622435.36

Mit den Maximalwerten wird nun eine neue Gauß-Krüger-*location* definiert (Auflösung: 25m, vgl. auch Abschnitt 4, wie eine Gauß-Krüger-*location* eingerichtet wird). Für dieses Beispiel ergibt sich (leicht gerundet):

north:	5834831
south:	5622435
west:	3427862
east:	3652268

Die Angabe der aktuellen Zeilen- und Spaltenzahl erhält man, wenn man das Koordinatenformular verlässt. Wenn Sie einmal Werte korrigiert müssen, z.B. bei Tippfehlern, beantworten Sie die Frage „Do you accept the region?“ mit „no“ und gelangen wieder zur Koordinateneingabe.

[1] Dazu eignet sich ab GRASS 5.0.x das Modul `$ m.proj` ganz hervorragend (vgl. Anhang A.5).

north:	5834831
south:	5622435
east:	3652268
west:	3427862
e-w res:	25.00066845 (Changed to conform to grid)
n-s res:	24.99952919 (Changed to conform to grid)
total rows:	8496
total cols:	8976
total cells:	76,260,096

Auf keinen Fall darf in dieser *location* gezoomt werden etc., da eine Datentransformation immer nur in der gesetzten Region (Ausschnitt) abläuft, der hier natürlich so groß wie das Gesamtgebiet sein soll. Nach dem Erzeugen der Gauß-Krüger-*location* und *mapset* verlassen Sie GRASS also sofort wieder. Dann rufen Sie es erneut mit der xy-*location* auf, die die UTM-Satellitenbilder beinhaltet. Sie können überlegen, ob Sie zur einfacheren Passpunktsuche zunächst ein oder zwei Farbkomposites erzeugen (je nach verfügbarem Platz auf der Festplatte), die Sie mit `$ i.composite` erzeugen können (vgl. Abschnitt 11.3.1.2).
Nun geht es folgendermaßen weiter:

(a) Sammlung der zu transformierenden Satellitenbilder in einer Bildgruppe:
`$ i.group`

- Namen für diese Bildgruppe vergeben: z.B. „satdaten“
- Auswahl aller Kanäle der LANDSAT-TM-Szene und eventuell der zusätzlichen Farbkomposits (mit einem „x“ markieren)
- Verlassen von `$ i.group` mit „return“

(b) Angabe der Ziel-*location* (target) und -*mapset* (die Gauß-Krüger-*location*) mit:
`$ i.target`

(c) Starten eines GRASS-Monitors: `$ d.mon start=x0`

(d) Zuweisung der neuen Gauß-Krüger-Koordinaten zu den vier Ecken des UTM-Satellitenbildes:
`$ i.points`

- die zu transformierende Gruppe von Satellitenbildern angeben
- Im GRASS-Monitor geht es nun weiter. Dort kann das für die Koordinatenzuweisung anzuzeigende Bild ausgewählt werden:
  z.B. `tm4.grey` oder ein Farbkomposite
- Mit der Maus wählen Sie, so gut es geht, den ersten Eckpunkt und geben das zugehörige Gauß-Krüger-Koordinatenpaar im Textfenster ein (den Rechts- und Hochwert durch ein Leerzeichen getrennt). So ist mit allen vier Ecken zu verfahren. Da es sehr schwierig ist,

die Ecken exakt zu treffen, wird im nächsten Schritt eine manuelle Korrektur vorgenommen. Dazu verlassen Sie `$ i.points`.

(e) Korrektur der Koordinaten-Eckwerte: Wechseln Sie in das Verzeichnis der GRASS-Database: `$ cd $LOCATION` Dort liegt ein Verzeichnis „group“, in dem sich ein Verzeichnis befindet, das den Namen der eben erzeugten Gruppe „satdaten“ trägt. Darin befindet sich die Datei „POINTS“, die Sie in einen Texteditor (z.B joe oder xedit) laden.
In den Spalten 1 und 2 sind die Eckkoordinaten der xy-*location* angegeben, die Spalten 3 und 4 dagegen beschreiben die jeweils zugewiesenen neuen Gauß-Krüger-Eckkoordinaten.

Die Spalten 1 und 2 sollten nun auf die eigentlichen Eckkoordinaten der xy-*location* korrigiert werden (für das Beispiel „Hannover“ hier):

north:	8496.000000
south:	0.000000
east:	8976.000000
west:	0.000000

Die Werte müssen mit denen, die man bei der Einrichtung der xy-*location* gewählt hat, übereinstimmen (s. Abschnitt 6.2).

Die geänderte Datei ist dann abzuspeichern.

(f) Erneute Kontrolle mit i.points:
Wie in Punkt (d) wird das Modul und darin die „ANALYSE“-Funktion aufgerufen. Der rms-error sollte nun Null betragen - wenn nicht, sind die Koordinaten noch einmal zu überprüfen (vgl. Punkt (e)).

(g) Jetzt kann die Transformation gestartet werden:
`$ i.rectify`
Wieder wird die Gruppe mit dem Satelliten angegeben, dann der Name, den der jeweilig zu transformierende Kanal in der Ziel-*location* erhalten soll. Da es sich um eine lineare Transformation handeln soll, wird als „order of transformation:“ „1“ angegeben.
Nun wird erfragt, ob (1.) in die „current region“ der Ziel-*location* oder (2.) in die „minimal region“ transformiert werden soll. Hier ist unbedingt Menüpunkt (1) zu wählen, nämlich „current region“, die ja der gesamten Gauß-Krüger-*location* entspricht. Die benötigte Rechenzeit kann vielleicht abgeschätzt werden, siehe dazu Abschnitt 11.1. Der Prozess arbeitet im Hintergrund, GRASS kann problemlos mit anderen Aufgaben beschäftigt oder verlassen werden (GRASS-Monitor schließen!), sofern dringendere Arbeiten rufen. Das Programm i.rectify gibt das Ende der Transformation durch eine email bekannt.

Nach erfolgreicher Transformation der Satellitenbildszene steht nun die Überprüfung eines eventuellen ungewollten Koordinatenversatzes an (bei ungenauer Passpunktvergabe). Dazu wird GRASS

neu mit der Gauß-Krüger-*location* aufgerufen. Ein Vergleich markanter Punkte beispielsweise mit einer zusätzlich importierten topographischen Karte oder ATKIS-Daten gibt Ihnen die Möglichkeit, die Lagegenauigkeit zu überprüfen. Zum besseren Erkennen sollten Sie Ausschnitte vergrößern.
Ist tatsächlich ein systematischer Versatz erkennbar, so sind die Werte der Gauß-Krüger-*location* zu verschieben. Dazu muss eine neue, hoffentlich endgültige Gauß-Krüger-*location* erzeugt werden und oben beschriebener Prozess der Passpunktsuche und Transformation wiederholt werden. Die nicht verwendbare Gauß-Krüger-*location* kann vorher gelöscht werden (siehe Anhang A.3.1).
Ein Hinweis: Sollte der Platz nicht ausreichen, die Satellitendaten kurzfristig zweifach (in der xy-*location* und Gauß-Krüger-*location*) vorzuhalten, müssen die Kanäle einzeln transformiert und jeweils nach erfolgreicher Transformation in der xy-*location* gelöscht werden. Das ist im Gegensatz zum Übertragen „in einem Rutsch“ leider ein etwas aufwendigeres Unterfangen.

11.1.3 Koordinatentransformation kleiner Bild-Ausschnitte auf eine Georeferenz

Im vorherigen Abschnitt wurde die Transformation einer gesamten Szene besprochen. Soll dagegen nur eine kleine Ausschnittsfläche transformiert werden oder sind die Randkoordinaten der Satellitenbildszene nicht bekannt, muss anders vorgegangen werden. Es wird eine digitale Grundlage benötigt, auf die entzerrt werden kann (Raster- oder Vektordaten, z.B. topographische Karte). Statt einer Affintransformation wie im vorherigen Abschnitt wird nun eine nicht-lineare Polynomtransformation höherer Ordnung mit mehr als vier Passpunkten durchgeführt.
GRASS transformiert immer nur den Ausschnitt, der in der „Ziellocation“ sichtbar ist (bei aktuell gesetzter Auflösung), lassen sich sehr einfach Ausschnittstransformationen durchführen. Im Gegensatz zum vorherigen Abschnitt werden Sie hier also die *mapset* auf den Sie interessierenden Bereich einstellen.
Diese Transformationsart benötigt je nach Grad des verwendeten Polynoms unterschiedlich viele Stützstellen, im Satellitenbild als Passpunkte bezeichnet.

Die nicht-lineare Polynom-Transformation kann 2. bis x-ter Ordnung sein, die Zahl n der benötigten Passpunkte ergibt sich aus der Ordnungszahl p (Polynomgrad):

$$n = \frac{(p+1)\cdot(p+2)}{2} \tag{11.1}$$

Für eine Transformation 3. Ordnung (p=3) werden also n=10 Passpunkte benötigt. Es ist aber dringend geraten, eine überbestimmte Transformation durchzuführen. Deshalb sind 20 Passpunkte bei einem Polynom 3. Ordnung empfehlenswert. Allgemein üblich ist die Transformation 3. Ordnung.

Sie benötigen nun also wieder eine xy-*location* mit den Satelliten-Rohdaten und Ihre Gauß-Krüger-*location*, in die Sie Referenzierungsdaten importieren müssen (sofern noch nicht vorhanden): beispielsweise eine topographische Karte oder ATKIS-Daten. Rufen Sie zunächst GRASS mit der Gauß-Krüger-*location* auf und setzen Sie die Koordinaten entsprechend Ihres gewünschten Ausschnitts

(mit g.region oder d.zoom). Die Auflösung ist ebenfalls einzustellen, da sie für die Transformation relevant ist. Da jede Karte in GRASS ihre eigene Auflösung haben kann, ist hier bei LANDSAT-Daten 25m einzustellen.
Nun verlassen Sie GRASS (Gauß-Krüger) und rufen es erneut mit der xy-*location* auf. Die Vorgehensweise verläuft bis auf das Setzen der Passpunkte wie im vorherigen Abschnitt. Je nachdem, ob auf eine Rasterdatengrundlage (z.B. topographische Karte) oder eine Vektorkarte (z.B. ATKIS-Daten) entzerrt werden soll, wird das entsprechende Modul zur Passpunktsuche gestartet:

In `$ i.points` können Sie zwei *location* anzeigen, links die xy-*location* mit den Rohdaten, rechts die Georeferenzkarte. Sie laden also zunächst in der linken Seite eine Rohkarte (z.B. LANDSAT Kanal 4 oder ein Farbkomposite). Dann wird in der rechten Fensterhälfte eine zweite Rasterkarte ausgewählt: Im Menü „PLOT RASTER“ anklicken, dann in die rechte Fensterhälfte, um dort die Georeferenz auszuwählen und anzuzeigen. Wollen Sie stattdessen eine Vektorkarte als Georeferenz nehmen, benutzen Sie `$ i.vpoints` statt `$ i.points`.
Wurden im vorherigen Abschnitt die Gauß-Krüger-Koordinaten per Tastatur eingegeben, so werden nun die Passpunkte jeweils in der linken Fensterhälfte (Satellitenbild) als auch in der rechten Hälfte (Entzerrungsgrundlage Raster/Vektor) mit der Maus zugewiesen. Dazu klicken Sie einen Passpunkt im Satellitenbild an und den korrespondierenden Punkt in der Georeferenz. Die „ZOOM“-Funktion ist dafür sehr hilfreich. Auf diese Art lassen sich die geographischen Beziehungen zwischen dem unreferenzierten Satellitenbild und der Georeferenz setzen. Mit „ANALYSE“ können Sie (wie oben beschrieben) die Lagequalität überprüfen. Sie sollten einen RMS < 0.5 Rasterzellenauflösung (z.B. 12.5m) erreichen.

Benötigt werden rund 15-20 Passpunkte, die idealerweise im zu transformierenden Ausschnitt gleichverteilt sein sollten. Es wird hier keine weitere Korrektur mit einem Texteditor durchgeführt, da die Punkte schon korrekt liegen (sollten). Empfehlenswert ist es, zum Setzen der Passpunkte in der linken Fensterhälfte grauwertverbesserte Rasterbilder (z.B. `tm4.grey`) oder ein Farbkomposite zu benutzen. Die folgende Transformation wird automatisch mit allen Bildern durchgeführt, die in der Bildgruppe angegeben wurden, Ihre Referenzpunkte gelten für alle Kanäle.

Anschließend wird die Polynomtransformation mit

```
$ i.rectify
```

gestartet. Dabei ist zuerst die Ordnung des Polynoms („order“) anzugeben, also die Zahl 3. Anschließend wird erfragt, ob (1.) in die „current region“ der Ziel-*location* oder (2.) in die „minimal region“ transformiert werden soll. Hier ist unbedingt Menüpunkt (1) zu wählen, nämlich „current region“, die ja den vorher gesetzten Ausschnitt der Gauß-Krüger-*location* darstellt.

Nach dem Ende der Transformation schickt das Modul eine email, der Rechner kann derweil parallel mit anderen (wenig rechenintensiven) Aufgaben beschäftigt werden.

Nach der Benutzung von i.vpoints sollte der GRASS-Monitor mit
`$ d.frame -e` aufgeräumt werden.

11.2 Kontrastverbesserung bei Satellitenbildern

Wer über einen Computer ohne 24bit-Farbkarte verfügt, wird unter GRASS Satellitenbilder im Gegensatz zu der Darstellung durch Bildbetrachtungsprogramme wie „xv“ als sehr kontrastarm empfinden. Das liegt daran, dass die meisten Bildbetrachtungsprogramme eine Streckung der Farbskala durchführen, damit die Bilder auch bei nur 256 Farben kontrastreich wirken.

Es sind zwei Arten der Kontraststreckung zu unterscheiden:

1. Modifikation der Farbtabelle
2. Modifikation der Rasterzellenwerte

Die Grauwertänderung (Kontraststreckung) der Farbtabelle ist der üblichere Fall, da ja an sich die Werte im Satellitenbild unangetastet bleiben sollen. Sie dient beispielsweise als (optische) Arbeitsgrundlage und Hilfsmittel z.B. für eine Entzerrung. Bearbeitet werden bei der damit natürlich alle Kanäle. Geeignet sind vor allem die LANDSAT-Kanäle `tm4` und `tm5`, da sie die kontrastreichste Struktur aufweisen.

1. Es erfolgt das Einlesen des Rasterfiles und die Kontrastverbesserung in
   `$ r.colors`
   Die Ausgabe erfolgt unter gleichem Namen, d.h. die Farbtabelle der angegebenen Datei wird modifiziert bzw. neu erstellt.
2. `$ d.histogram` und `$ d.rast`
   ... zeigen, dass nun mehr als vier Grauwerte im Spektrum von 0 bis 255 zur Verfügung stehen.

Soll das Satellitenbild später wieder aussehen wie vorher, sollte vor der Benutzung des Moduls von der Farbtabelle manuell eine Kopie erstellt werden:

```
$ cd $LOCATION/colr
$ cp <tabelle> <tabelle.org>
```

Die Farbtabelle hat denselben Namen wie die zugehörige Rasterdatei. Später kann dann die originale Farbtabelle wieder zurückkopiert werden (die neue wird einfach überschrieben):

```
$ cd $LOCATION/colr
$ cp <tabelle.org> <tabelle>
```

Damit sieht das Rasterbild aus wie vorher.

Der zweite Fall ist die direkte Modifikation der Rasterzellenwerte, die in Spezialanwendungen wichtig sein kann. Die Module `$ r.stats` und `$ r.mapcalc` sind dafür nötig. Doch darauf soll in diesem Zusammenhang nicht weiter eingegangen werden.

11.3 Klassifizierung von Satellitenbildern

In einer multispektralen Klassifizierung werden prinzipiell Pixel gleicher oder ähnlicher Farbe zu einer Farbe zusammengefasst, um eine thematische Gliederung zu schaffen. Die Pixelwerte im Satellitenbild sind durch die von Boden, Vegetation usw. abhängige Strahlungsreflexion festgelegt. Dabei entsprechen ähnliche Pixelfarben in einem Kanal im Allgemeinen gleichen Bodenreflexionsparametern, sofern Hangneigung und Exposition konstant bleiben. Bei Kenntnis des Projektgebiets können anschließend diese Klassen konkreten Bodennutzungen bzw. -bedeckungen zugeordnet werden. Unter bestimmten Bedingungen lassen sich auch Rückschlüsse auf den Gesteinsuntergrund oder den Bodenwasserhaushalt ziehen. Problematisch sind allerdings die unterschiedlichen klimatischen Einflüsse bei verschiedenen Bildszenen, die das Reflexionsverhalten der Erdoberfläche beeinflussen.

Es werden sämtliche Werte aller Satellitenbildkanäle gleichzeitig auf Homogenitäten und Gruppierbarkeiten hin untersucht. Als Ergebnis stehen mehrere Klassen, die in sich annähernd homogen sind (Annahme einer Gaußschen Normalverteilung für jede Klasse, JACOBS 1998:181). Jede Klasse entspricht einer Landnutzung bzw. weiteren Bodenbedeckung (Wasser etc.). Anhand der Vegetationsklassen und des Aufnahmezeitpunkts der Satellitenbildszene kann die Pflanzenphänologie und damit die aktuelle Bodenbedeckung bestimmt werden. Problematisch sind für diese Verfahren Mischpixel, in denen verschiedene Objekte enthalten sind (bei Feldrändern, urbanen Flächen usw.). Die begrenzte Auflösung der Sensoren erfasst in diesen Flächen mehrere Objekte, die wiederum die Gesamtstatistik der Szene verändern. Das führt teilweise zu verschobenen Klassenabgrenzungen und damit nicht mehr validierbaren Landnutzungsklassen. Einzige Lösung ist die oft angewendete Maskierung problematischer Flächen (v.a. urbaner Gebiete) vor der Klassifizierung.

Die Auswertung von Satellitenbildern ist in drei Vorgehensweisen möglich: die unüberwachte, die überwachte Klassifizierung und eine Kombination aus beiden, die teilüberwachte Klassifizierung. GRASS bietet alle drei Methoden, deren Ablauf nun beschrieben werden soll. Die ersten zwei Varianten, basierend auf `$ i.maxlik` (Maximum Likelihood Klassifizierung), sind **radiometrische Klassifizierungen**, als dritte Möglichkeit wird anschließend eine **radiometrisch/geometrische Klassifizierung** beschrieben, basierend auf `$ i.smap` (Sequential Maximum a Posteriori Klassifizierung).

Die radiometrische Klassifikation („spectral pattern analysis“) nutzt den spektralen Informationsgehalt in den Bildkanälen. Näheres wird in den folgenden Kapiteln erläutert. In der kombinierten radiometrischen/geometrischen Klassifikation („spatial pattern analysis“) wird zusätzlich eine Nachbarschaftsuntersuchung der Pixel vorgenommen. Der Vorteil liegt darin, dass beispielsweise bei Flächen gleicher Nutzung die Farbähnlichkeit benachbarter Pixel mitberücksichtigt wird und lokale Inhomogenitäten nicht störend einfließen. Die Ergebnisse sind i.a. wesentlich besser als mit der üblichen radiometrische Klassifizierung. Doch wird zunächst auf die rein radiometrische Methode eingegangen.

11.3.1 Radiometrische Klassifizierungen

11.3.1.1 Unüberwachte Klassifizierung

Die unüberwachte Klassifizierung (unsupervised classification) ist die Zuordnung der Rasterpixel zu verschiedenen Spektralklassen auf automatisiertem Wege. Dabei wird dem Rechner unter Vorgabe bestimmter Parameter überlassen, diese Zuordnung durchzuführen. Im ersten Schritt wird mit einer Clusterbildung (Wertegruppierung basierend auf ähnlichen statistischen Eigenschaften) in einer Satellitenbildszene untersucht, wieviele ähnliche Rasterzellenwerte (sie entsprechen den Farben im Bild) vorliegen und zu einem Wert zusammengefasst werden können. Vergleichbar ist das mit der Erstellung einer Legende bei einer Karte, bei der zunächst untersucht wird, wie viele Kartensignaturen es überhaupt gibt. Die Zuordnung zu den räumlichen Einheiten erfolgt später im zweiten Schritt. Der iterative Clusteralgorithmus berechnet zunächst also Clustermittelwerte und Kovarianzmatrizen (Modul i.cluster). Dafür wird eine Auswertung der Häufigkeitsverteilung der einzelnen Rasterwerte in den einzelnen Kanälen (zwei- bis mehrdimensional) durchgeführt. Als Ergebnis bilden sich abgegrenzte „Punktwolken“, also die Abgrenzung von Häufigkeitskonzentrationen (vgl. Abb. 34). Jede „Wolke“ charakterisiert eine bestimmte Objektreflexion und wird als eine Klasse (cluster) zusammengefasst. Sie steht in direktem Zusammenhang mit den kanal- und pixelweise ermittelten Objektreflexionen. Aus diesem Grunde werden auch mindestens zwei Bilddatensätze benötigt, um eine Clusterung durchzuführen. Bei den einzelnen Iterationen verändern sich die Clustermittelwerte, da jedesmal die Pixel den gebildeten Clustern neu zugeordnet werden (der Algorithmus zielt auf das Erreichen der maximalen Distanz zwischen den Clustern) und sich dadurch die Cluster wiederum verändern.

Mit diesen gewonnenen statistischen Werten erfolgt dann die räumliche Zuordnung (Modul i.maxlik) der einzelnen Pixel nach der größten Wahrscheinlichkeit ihrer Klassenzugehörigkeit (Maximum Likelihood Diskriminant-Analyse). Es wird ein Chi-Quadrat-Test mit unterschiedlichen

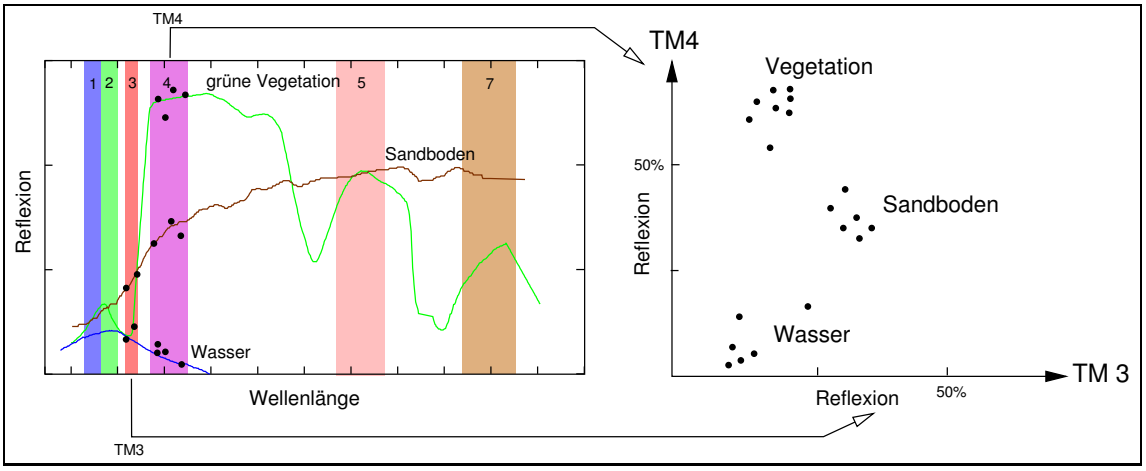


Abbildung 34: Häufigkeitsverteilung der Pixelwerte zweier Satellitenbildkanäle im zweidimensionalen Merkmalsraum (Beisp.: LANDSAT-TM)

Schwellenwerten bis zur Erreichung der vorgegebenen Konvergenz (Stabilität der Pixelzuordnungen bei den einzelnen Iterationsschritten) durchgeführt. Das Ergebnis ist ein neues Rasterbild, das gegenüber den zugrunde gelegten Bildkanälen (z.B. LANDSAT-TM Kanäle 1 bis 7) so vereinfacht ist, dass nur noch die durch i.cluster ermittelten Klassen enthalten sind. Außerdem errechnet das Modul für jedes Pixel den Konfidenzgrad, also die Wahrscheinlichkeit, mit der dieses Pixel einer Klasse zugeordnet werden konnte. Diese Wahrscheinlichkeiten werden in einem weiteren Rasterbild („reject threshold map layer“) dargestellt, der sogenannten „Zurückweisungsdatei“. Hohe Wahrscheinlichkeiten führen in dieser Datei zu hohen Pixelwerten (entspricht hellen Grauwerten), schlecht zugeordnete Pixel besitzen eine niedrige Zuordnungswahrscheinlichkeit und erscheinen demnach in diesem Bild dunkel. Ideal ist also eine gleichmäßig helle Zurückweisungsdatei (Rasterfläche). In weiteren Schritten können beispielsweise die schlecht zugeordneten Pixelflächen ausmaskiert (Modul `$ r.mask`) und gesondert bearbeitet werden.

Vor der Klassifizierung können Sie sich die „Punktwolken“ anschauen mit
`$ dcorrelate.sh kanal1 kanal2 [kanal3] [kanal4]`
Dazu ist vorher ein GRASS-Monitor zu öffnen. Als Parameter sind zwischen zwei und vier Satellitenbildkanäle anzugeben, in verschiedenen Farben werden dann die Merkmalsräume angezeigt. Sie sehen also die Datengrundlage, die für den Clustering-Prozess benutzt wird.

Die unüberwachte Klassifizierung wird mit der

(a) Sammlung der zu bearbeitenden Satellitenbilder in einer Bildgruppe begonnen:
`$ i.group`

1. Neue Gruppe erstellen (Namen eingeben) und alle Kanäle angeben (Markierung mit „x“).
2. Über Menüpunkt 5 eine Untergruppe (subgroup) festlegen, auch hier wieder alle Kanäle benennen.

(b) Anschließend erfolgt die Clusterbildung mit `$ i.cluster`:

Dabei ist die Anzahl der Klassen, die bei der Clusteranalyse initial vor dem ersten Durchlauf des Algorithmus' ausgewiesen werden sollen, anzugeben („*number of initial classes*“, Näheres siehe unten). Weitere Parameter können zunächst von der Vorgabe übernommen werden. Ihre Bedeutung ist folgende (Klasse = cluster):

- *Minimum class size* (minimale Klassengröße): Minimale Pixelanzahl, die in einem Bild vorhanden sein muss, um eine Klasse auszuweisen.
- *Class separation* (minimale Klassentrennung): Abstand zwischen einzelnen Klassen, unterhalb dessen die jeweiligen Klassen beim Iterationsprozess zu einer Klasse verschmolzen werden. Je größer der vorgegebene Abstand, desto mehr Klassen werden verschmolzen (übliche

Werte liegen hier zwischen 0.5 und 1.5). Bei einer Erhöhung sollte auch „Maximum number of iterations“ hochgesetzt werden, um eine hohe „Percent convergence“ zu erreichen. Je mehr verschmolzen wird, desto niedriger sinkt dieser Wert, ergo sind wiederum mehr Iterationen nötig, um die Konvergenz zu erhöhen.

- *Percent convergence* (prozentuale Konvergenz): gibt an, wieviel Prozent der Pixel bei den einzelnen iterativen Durchläufen von i.cluster stabil bleiben müssen und nicht mehr ihren zugehörigen Cluster wechseln, um den Cluster-Prozess beenden zu können.
- *Maximum number of iterations* (maximale Anzahl der Iterationen): Es werden mehrere Iterationen mit unterschiedlichen Schwellenwerten durchgeführt, bis der Chi-Quadrat-Test die vorgegebene Konvergenz erreicht hat.
- *Sampling intervals* (Pixelzusammenfassung in Zeilen und Spalten): Zur Vereinfachung der Rechnung werden Pixel zu Blöcken zusammengefasst. Wird die Rechenkapazität aufgrund zu klein gewählter Blöcke zu knapp, sendet i.cluster eine email mit einer Warnung.

Wie wird nun konkret vorgegangen?
Nach dem Aufruf von `$ i.cluster` ist Folgendes einzugeben:

1. Gruppe, Untergruppe (subgroup) auswählen.
2. Dateinamen für „Result-signature“ angeben (enthält die Clusterinformationen für i.maxlik): z.B. `tm89.result.sig`
3. Im ersten Durchlauf keinen Dateinamen für „Seed-signature“ eingeben (diese Option ist für Clusterinformationen aus vorherigen Durchläufen vorgesehen oder für definierte Vorgaben bei einer teilüberwachten Klassifizierung mit i.class): „return“
4. Dateinamen für „Report-file“ angeben (liegt später im aktuellen Verzeichnis und enthält statistische Angaben zum Auswertungsprozess):
   z.B. `tm89.report.txt`
5. „Run in background?“: „no“.
6. Das nachfolgende Formular gestattet die Angabe der oben angesprochenen Parameter:
   „Number of initial classes“: 20 (testweise – ob die Zahl variiert werden muss, zeigt erst die Überprüfung des Ergebnisses z.B. bei Nichterreichen der Konvergenz). Die übrigen Parameter können zunächst übernommen werden. Weiter geht es mit „ESC“.

(c) Nach erfolgreicher Clusteranalyse, evt. nach Variation der Parameter, beginnt die Durchführung der unüberwachten Klassifizierung nach dem „Maximum-Likelihood-Verfahren“:
`$ i.maxlik`
Nach dem Verfahren der maximalen Wahrscheinlichkeit werden die Pixel im Satellitenbild entsprechend der im Clusterprozess gefundenen Signaturen (Klassen) zugewiesen:

1. Zuerst wird die Bildgruppe ausgewählt.

2. *Result signature map:* Diese Datei ist das Ergebnis aus dem Clusterprozess, das die gefundenen Signaturen enthält (als Dateiname ist der Name aus (b.2) anzugeben (s.o.)).

3. *Classified map layer:* Name der zu errechnenden Ergebnisdatei (das klassifizierte Satellitenbild), z.B. `landsat89.class`.

4. *Reject threshold map:* Diese Rasterdatei enthält die Zurückweisungsdaten, also den lokalisierten Fehler, der beim Zuweisen der Pixel zu den Klassendefinitionen aus i.cluster entstand. Die hier dunkel erscheinenden Flächen konnten nicht oder nur schlecht zugewiesen werden.
   Man erhält so ein Qualitätsmaß für die Klassifizierung. Je gleichmäßiger und heller die Grauwerte (die Gleichmäßigkeit dieser Rasterdatei, Überprüfung später mit `$ d.rast`), desto geringer der Zuordnungsfehler.
   Dateiname: z.B. `tm89.class.reject`

GRASS führt in der anschließenden Rechnung die unüberwachte Klassifizierung durch.

(d) Die graphische Ausgabe der berechneten Klassifizierungsergebnisse erfolgt mit:

1. Die Überprüfung der Zurückweisungsdatei:
   z.B. `$ d.rast tm89.class.reject`
   Je gleichmäßiger der Grauwert, desto geringer sind die Zuordnungsfehler. Wenn das Ergebnis qualitätsmäßig zu schlecht ist, müssen die Zahl der Klassen oder die anderen Parameter verändert und die Cluster- sowie Maximum-Likelihood-Analyse neu durchgeführt werden.

2. Das klassifizierte Satellitenbild wird z.B. mit `$ d.rast tm89.class`
   visualisiert. Den klassifizierten Flächen können manuell Bodenbedeckungen oder Nutzungen zugeordnet werden.

(e) Um die Farben anzupassen (Wasser = blau etc.), kann mit `$ d.colors` die Farbtabelle des Bildes modifiziert werden. Eine Übersicht der gängigen Farbwerte findet sich im Abschnitt A.8.2, alle definierten Farben (gemäß X11-Konsortium) liefert der UNIX-Befehl: `$ showrgb`

11.3.1.2 Überwachte Klassifizierung

Die überwachte Klassifizierung (supervised classification) basiert auf der Verwendung bekannter Testflächen (Trainingskarte) zur Klassifikation. Hier ergeben sich die einzelnen Merkmalsklassen nicht rein aus den statistischen Bilddaten, sondern aus von der Benutzerin oder dem Benutzer vorgegebenen Daten. Dabei werden Testflächen im Satellitenbild eingegrenzt und über den Klassifizierungsalgorithmus (statistische Analyse der gesamten Szene) weitere Flächen mit denselben Pixeleigenschaften herausgesucht.

Bei dieser radiometrischen Klassifikation werden Histogramme für die einzelnen Kanäle erstellt und die Möglichkeit gegeben, die Standardabweichung für die Zuordnung der Pixel zu beeinflussen. Die spektralen Signaturen (Klassen) werden aus den regionalen Mittelwerten der Testflächen und den zugehörigen Kovarianzmatrizen berechnet.

Die Testflächen können entweder direkt im Klassifizierungsmodul i.class digitalisiert (erster Teil der Beschreibung) oder extern vorbereitet werden (zweiter Teil in diesem Kapitel).

1. Das Verfahren mit der **direkten Vektorisierung** der Testflächen läuft folgendermaßen ab:

(a) Zusammenfassung der Bilder mit `$ i.group` zu einer Bildgruppe:

1. Neue Gruppe erstellen (Namen eingeben) und alle Kanäle angeben (Markierung mit „x“).
2. Über Menüpunkt 5 die Untergruppe (subgroup) festlegen, auch hier wieder alle Kanäle angeben.

(b) Erzeugen eines Echtfarbbilds, das zur Markierung der Testflächen dienen soll, mit
`$ i.composite`

Als Werte für die Farben werden definiert (b=Blau, r=Rot, g=Grün):

b = `tm1`

g = `tm2`

r = `tm3`

Ein Komposit ist eine Zusammensetzung dreier Bilder zu einem einzigen über die Zuordnung der einzelnen Bilder zu den drei Grundfarben. Im Anhang sind weitere Komposits für andere Zwecke beschrieben (siehe Abschnitt A.8.3).

Als „Number of color levels“ sollte 10 angegeben werden. Weniger „levels“ beschleunigen zwar die Rechendauer, die Farben wirken dann aber nicht mehr realistisch. Da drei Grundfarben vorliegen (R, G, B), berechnet sich die Zahl der Farben nach:

$$Farbenanzahl = Level^3 \tag{11.2}$$

Für 6 Level ergeben sich also 216 Farben, für 7 Level 343 Farben usw.

Ein Hinweis: Taucht bei der Erstellung des Komposites eine „WARNING: histogram...“ auf, werden die Farben nicht passend berechnet. In diesem Fall können Sie mit Strg-C das Modul i.composite beenden und sollten die Bildstatistik mit `$ r.support` für jeden Bildkanal berechnen („Edit header“: no, „update stats...“: yes, weitere Fragen: return). Anschließend wird i.composite ohne Warnung korrekt arbeiten.

(c) Idealerweise blendet man vor der Klassifizierung Ortschaften und Straßen beispielsweise durch eine Überlagerung mit ATKIS-Daten oder von selbstdigitalisierten Linien und Polygonen (mit `$ v.digit`) in den TM-Kanälen und dem Echtfarb-Komposit aus. Da bebaute Flächen ein sehr breites Spektrum aufweisen, kann man auf diese Art viele Zuweisungsprobleme umgehen.

Diese Überlagerung wird folgendermaßen durchgeführt:

Die Vektordatei muss in das Rasterformat umgewandelt werden (vgl. Abschnitt 7.6). Dabei haben alle relevanten Vektoren (Straßen, Ortschaften) den Wert 1, der Rest ist auf 0 zu setzen. Sind die Vektoren unterschiedlich „gelabelt“, so sollten diese Änderungen in einer Kopie durchgeführt werden, um die Vektordatei später weiterverwenden zu können. In `$ r.mapcalc` werden diese Labelwerte in der neu entstandenen Rasterdatei für die Überlagerung mit den Satellitenbildern dann invertiert (die Nachkommastelle gibt an, auf welchen Wert das Pixel gesetzt werden soll, in einer Zeile einzugeben):

```
mapcalc> atkis.neu = if(atkis == 10 ,0) + if(atkis == 20 ,0)
+ if(atkis == 30 ,0) + if(atkis == 40 ,1)
```

Hier galten vorher folgende Labelwerte: Straßen: 10, Wasser: 20 (alles Linienvektoren), Siedlungen: 30, Ackerflächen: 40 (alles Polygone). Im Ergebnis enthält die Rasterdatei `atkis.neu` in diesem Beispiel nun Linienstrukturen und Siedlungsflächen mit dem Wert 0, alle landwirtschaftlichen Flächen haben den Wert 1.

Alle Satellitenbilder und das Echtfarb-Komposit werden nun mit dieser gerasterten Vektordatei multipliziert (Beispiel):

```
mapcalc> kompositneu = kompositalt * atkis.neu
```

Damit sind alle Straßen, Wasserläufe und Ortschaften ausmaskiert und beeinflussen die Klassifizierung nicht mehr.

(d) Start des Klassifizierungsmoduls:

```
$ i.class
```

- *Result signature map:* Diese Datei enthält nach der Testflächenausweisung die Klassendefinitionen. Bei jedem Durchlauf von i.class (z.B. bei einer Unterbrechung der Arbeit) muss ein neuer Name vergeben werden, da sonst diese Daten aus vorherigen Durchläufen überschrieben werden. Die Dateien stehen prinzipiell im lesbaren ASCII-Textformat im Verzeichnis:

  `$LOCATION/group/<gruppenname>/subgroup/<subgruppenname>/sig/`

- *Seed signature:* ermöglicht das Einlesen bereits vorher berechneter Klassensignaturen.

In dem Modul lässt man sich als „cell map to be displayed“ das vorher erzeugte Echtfarb-Komposit ausgeben. Die „ZOOM“-Funktion erlaubt, Ausschnitte zu vergrößern, in denen

dann mit „DEFINE REGION“, „DRAW REGION“ die Testflächen digitalisiert werden können, deren Nutzung bzw. Bodenbedeckung bekannt ist. Das Modul sucht im Gesamtbild nach den Spektralsignaturen der jeweiligen Testfläche. Sehr wichtig ist, außerhalb des im GRASS-Monitor dargestellten Rasterbilds keine Testflächen zu digitalisieren, da sonst Absturzgefahr für i.class besteht. Um zu guten Ergebnissen zu gelangen, sollte bei Testflächen, die mit Äckern deckungsgleich sind, die Ränder nicht mitdigitalisiert werden, da dort überwiegend Mischpixel durch die Randvegetation auftreten. Sehr wichtig ist, nicht zu kleine Testflächen zu digitalisieren, da diese Gebiete sonst aus statistischen Gründen ignoriert werden (i.class gibt eine Warnung aus).

Beim Verlassen des Klassifizierungsmoduls werden die aktuell ausgewiesenen Klassen gespeichert (für jede Klasse kann auch einzeln eine eigene Signatur erzeugt werden, indem i.class nach der Digitalisierung jeder Testfläche verlassen wird).

(e) Die Klassenzuweisung erfolgt mit:

```
$ i.maxlik
```

Im Gegensatz zur unüberwachten Klassifizierung (s. Abschnitt 11.3.1.1) werden nun als „Result signature map“ die vorher erzeugten Klassen verwendet. Die übrigen Angaben verhalten sich wie im vorherigen Abschnitt.

2. Nun folgt der Weg der **externen Vektorisierung von Testflächen**: Sie bietet sich an, wenn die Testflächen aus einer topographischen Karte oder anderen Quellen (beispielsweise Flächeninformationen aus ARC/INFO-Import) gewonnen werden sollen.
Mit `$ v.digit` (siehe Kapitel Vektorverarbeitung) werden Flächen digitalisiert („areas“), die dann mit `$ v.to.rast` in eine Rasterdatei umgewandelt werden müssen. Dabei ist extrem wichtig, dass die Flächen z.B. in v.digit ein Label, also einen Wert zugewiesen bekommen. Ansonsten tauchen sie nach der Konvertierung nicht mehr auf.
Alternativ kann auch direkt mit einem Rasterbild mit `$ r.digit` als Grundlage digitalisiert werden. Beim Verlassen dieses Moduls gibt man den Namen der neu erstellten Flächeninformationsdatei an.
Sind die Flächeninformationen im ARC/INFO-„ungenerate“-Format, können sie importiert (s. Abschnitt A.4) und mit umgewandelt werden.

Diese Trainingsflächenkarte ist der Input für `$ i.gensig`. Dieses Modul erstellt die Signaturdatei anhand der Flächenstatistik. Die eigentliche Analyse nimmt anschließend wieder `$ i.maxlik` vor. Das Modul i.class wird hier also nicht eingesetzt.

11.3.1.3 Teilüberwachte Klassifizierung

Die teilüberwachte Klassifizierung läuft im Prinzip wie die oben beschriebene unüberwachte Klassifizierung ab. Der Unterschied besteht in der Berücksichtigung von Testflächen, die vor dem Aufruf von `$ i.cluster` mit `$ i.class` bzw. `$ i.gensig` (in Kombination mit der externen

Digitalisierung) erstellt werden müssen (vgl. dazu die „Überwachte Klassifizierung“ im vorigen Abschnitt). Dann wird das Clustermodul `$ i.cluster` gestartet und als „Seed signature“ das Ergebnis aus i.class angegeben (dessen „Result signature“-Datei). Anschließend wird wie bei der „unüberwachten Klassifizierung“ weiterverfahren.

11.3.2 Überwachte geometrische Klassifizierung

GRASS bietet noch ein weiteres überwachtes Klassifikationsverfahren an. Der verwendete Algorithmus der geometrischen Klassifikation heißt „*SMAP - sequential maximum a posteriori - estimation*“. Dabei wird die Analyse nicht, wie oben beschrieben, pixelweise, sondern matrixweise durchgeführt, es werden also nachbarschaftliche Ähnlichkeiten berücksichtigt (SCHOWENGERDT 1997, S. 107, REDSLOB 1998, S. 123-127, vgl. auch RIPLEY 1996, S. 167-168). Diese Kombination führt zu einer deutlichen Verbesserung der Klassifikationsergebnisse.
Die Schritte bei der SMAP-Klassifikation sehen folgendermaßen aus:

(a) Zusammenfassung der Bilder mit `$ i.group` zu einer Bildgruppe:

1. Neue Gruppe erstellen (Namen eingeben) und alle Kanäle angeben (Markierung mit „x“).
2. Über Menüpunkt 5 die Untergruppe (subgroup) festlegen, auch hier wieder alle Kanäle benennen.

(b) Externe Digitalisierung von Testflächen mit `$ v.digit` oder `$ r.digit` bzw. Import. Konvertierung der Vektorflächen zu einer Rastertrainingskarte mit `$ v.to.rast`. Wichtig ist, dass später nur die Klassen ausgewiesen werden, die auch eine durch eine Testfläche abgedeckte Spektralinformation haben. D.h. die Zahl der Testflächen bestimmt die Zahl der Klassen. Wichtig ist, dass die Testflächen mehrere Pixel groß sein müssen, da sie sonst ignoriert werden.

(c) Berechnung der Cluster mit:

```
$ i.gensigset
```

Das Modul fragt zunächst nach der Trainingskarte, dann nach group und subgroup aus i.group. Das vom Modul erstellte „subgroup signature file“ entspricht dem oben beschriebenen „result signature file“.

(d) Die überwachte Klassifikation erfolgt mit:

```
$ i.smap
```

Group und subgroup müssen wieder angegeben werden und dann das „subgroup signature file“. Anschließend erfolgt die Berechnung.

Mit einer ausreichenden Anzahl von Testflächen sind die Ergebnisse mit diesem Algorithmus besser als mit dem „Maximum-Likelihood“-Verfahren. Ein Vergleich lässt sich bei MCCAULEY UND ENGEL 1994 nachlesen.

11.3.3 Kurzübersicht der Klassifikationsverfahren

Wie Sie gesehen haben, gibt es verschiedene Möglichkeiten, multispektrale Daten zu klassifizieren. Hier noch einmal zusammengefasst die drei Klassifizierungsmöglichkeiten:

	un-überwacht	**radiometrisch, überwacht**		**radio- und geometrisch, überwacht**
Vorbereitung	i.cluster	i.class (Bildschirm)	i.gensig (ATKIS etc.)	i.gensigset
Berechnung	i.maxlik	i.maxlik	i.maxlik	i.smap

Tabelle 11.1: Klassifikationsmethoden in GRASS

11.4 Hauptachsentransformation

Bei Satellitendaten handelt es sich um mehrdimensionale Datensätze, die sowohl das eigentliche Signal als auch Störungen enthalten. Eine in der Satellitendaten-Auswertung vielfach verwendete Methode ist die Analyse von Datensätzen mit der Hauptkomponentenanalyse (auch als *Principal Component Analysis* (PCA) oder *Principal Component Transformation* (PCT) bezeichnet). Sie soll hier im Überblick vorgestellt werden.

Bei der Hauptkomponentenanalyse (PCA) handelt es sich um eine algebraische Methode zur Varianzanalyse eines Datensatzes, bei der die Varianz jeder Variablen (hier die Datenmatrix eines Bildkanals) durch die ermittelten Hauptkomponenten „erklärt“ wird (BAHRENBERG 1992:207). Ein Ziel der PCA ist die Berechnung von stochastisch unabhängigen Faktoren, mit denen sich die Datenmatrizen als Linearkombination dieser Faktoren in neue Datenmatrizen, die sogenannten Hauptkomponenten, unter Erhalt der Gesamtvarianz transformieren lassen (SCHOWENGERDT 1997:191). Diese Hauptkomponenten sind also wiederum Bildkanäle, die aber im Gegensatz zu den ursprünglichen Daten keine linearen Abhängigkeiten mehr voneinander aufweisen. Sie bilden ein neues spektrales Koordinatensystem für die Daten.
Die erste Hauptkomponente erklärt dabei den größten Teil der Gesamtvarianz. Die zweite Hauptkomponente beschreibt den größten Teil der Varianz, die von der ersten nicht erklärt wird (da sie orthogonal zur ersten ist) usw. Insgesamt werden also bei einem mehrkanaligen Bilddatensatz in der PCA maximal genauso viele neue Kanäle berechnet, wie Eingangskanäle vorhanden sind. Allerdings befindet sich die größte erklärte Varianz in den ersten Kanälen, die geringste (z.B. auch das unkorrelierte Bildrauschen) im letzten. Häufig genügen weniger Hauptkomponenten als Eingangsvariablen, um den größten Teil der Varianz zu reproduzieren. Eine übliche Anwendung dieser Eigenschaft ist die Möglichkeit, so eine Satellitenbildszene ohne relevanten Informationsverlust auf weniger Kanäle zu reduzieren und Speicherplatz zu sparen. In Abb. 35 ist die Reduktion für zwei Satellitenbildkanäle dargestellt.

So kann beispielsweise eine LANDSAT-Satellitenbildszene auf drei Hauptkanäle reduziert werden, um eine RGB-Darstellung (`$ d.rgb`) oder eine Kompositbildung (`$ i.composite`) zu ermöglichen. Die erste Hauptkomponente enthält eine gewichtete Summe der Eingangskanäle und zeigt gemittelte Albedounterschiede. Die zweite Hauptkomponente stellt die wesentlichen spektralen Unterschiede, die dritte die geringeren spektralen Unterschiede dar (nach BÄHR/VÖGTLE 1991).

Die Hauptkomponentenanalyse basiert auf einer Korrelationsrechnung. Fasst man die Eingangsdaten vektoriell auf, werden in der Rechnung neue Vektoren gesucht, die möglichst hoch mit den Originalvektoren korrelieren. Die Abbildung 36 zeigt in der linken Hälfte standardisierte Datenvektoren (gleiche Beträge) und die zugehörigen Vektoren der Hauptkomponenten. Diese Datenvektoren entsprechen in Koordinatendarstellung (rechts in Abb. 36) Datenpunkten in ihrem Originaldaten-Koordinatensystem. Zusätzlich zeigt die rechte Abbildung die Lage des neuen Hauptkomponenten-Koordinatensystems.

Diese PCA-Vektoren sind die Eigenvektoren der Datenmatrix, die den PCA-Merkmalsraum aufspannen. Das PCA-Koordinatensystem ist gegenüber dem Original-Koordinatensystem der Eingangsdaten im Allgemeinen gedreht. Die Korrelationskoeffizienten zwischen einer Hauptkomponente und den Variablen (Eingangsdaten) werden als Komponentenladungen bezeichnet. Die Summe der quadrierten Komponentenladungen ergeben den Eigenwert einer Hauptkomponente (BAHRENBERG 1992:218). Die Länge der Eigenvektoren ist gleich der Wurzel aus den Eigenwerten (ÜBERLA 1968:98). Die einzelnen Komponentenladungen geben an, welcher Anteil der Varianz der **je-**

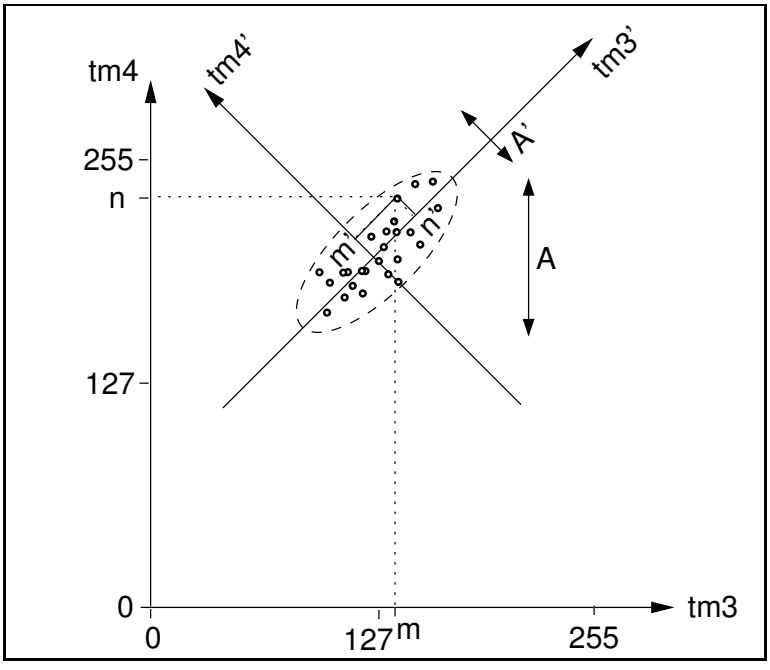


Abbildung 35: Hauptachsentransformation zur Datenreduktion im Merkmalsraum zweier Satellitenbildkanäle, Amplitude A wird zu A' reduziert

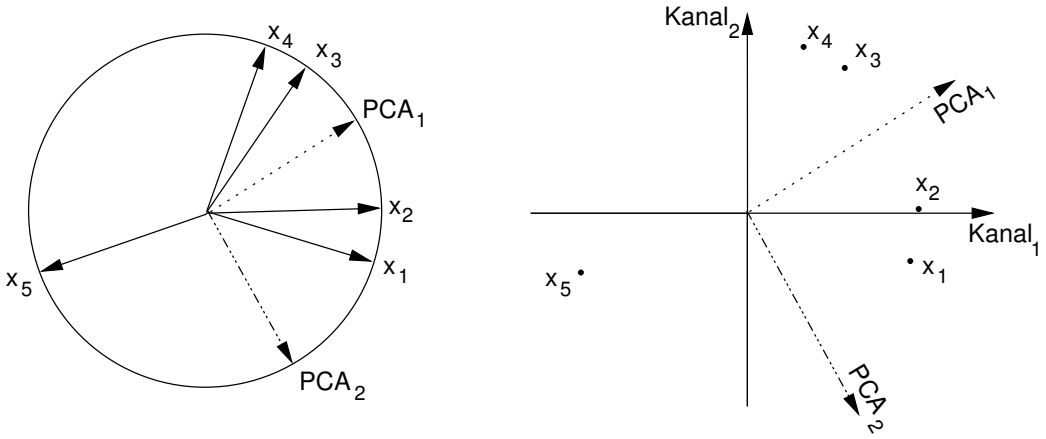


Abbildung 36: Abbildung von Datenpunkten als standardisierte Datenvektoren x_1 bis x_5 mit den zugehörigen ersten beiden orthogonalen Hauptkomponenten-Vektoren PCA_1 und PCA_2 in Kreisdarstellung (links) und als Punktdaten in Koordinatendarstellung (rechts) (linker Teil nach BAHRENBERG 1992:215, veränd.)

weiligen Variablen auf die Hauptkomponente entfällt. Um die Gesamtvarianz zu ermitteln, die eine Hauptkomponente erklärt, teilt man den Eigenwert durch die Anzahl der Komponentenladungen. Die Matrix der Komponentenladungen (Korrelationskoeffizienten, Eigenvektoren) wird üblicherweise vom Transformationsprogramm zur Analyse ausgegeben (so auch in GRASS vom Modul *i.pca* am Ende der Berechnungen). Die genannten Ladungen sind darin spaltenweise gespeichert.

Der zentrale Punkt, weshalb eine Hauptkomponentenanalyse für die Auswertung von Satellitendaten geeignet ist, resultiert aus der Eigenschaft dieser Transformation, die Daten ihrer Varianz nach neu zu sortieren. Die Zahl der Ergebniskanäle ist identisch mit der der Eingangskanäle – bei einer LANDSAT-Szene werden somit sechs PCA-Kanäle errechnet. Der erste neue Bildkanal weist den größten Anteil an erklärter Varianz auf, den niedrigsten Anteil im letzten Kanal. Es befinden sich also die wesentlichen Bildinformationen in den ersten vier bis fünf Kanälen, in den letzten dagegen hauptsächlich das unkorrelierte Bildrauschen (SCHOWENGERDT 1997:191).

Wichtig ist, dass die Hauptkomponentenanalyse datenabhängig ist, also je nach Landschaftstyp, Bodenbedeckung, Jahreszeit, gewähltem Ausschnitt usw. in Satellitenbildern unterschiedliche Ergebnisse erzielt werden. Dabei spielt nicht nur eine Überschneidung mancher Satelliten-Spektralbereiche eine Rolle (bei LANDSAT-TM überlappen sich die Spektralfilter der Kanäle 1-3 geringfügig), sondern auch Materialkorrelation (gleiches Reflexionsverhalten in verschiedenen Kanälen, vgl. SCHOWENGERDT 1997:187). Wenn ein Gebiet im Satellitenbild beispielsweise recht homogen von Vegetation bedeckt ist, kommt es durch das ähnliche Spektralverhalten zur räumlichen Korrelation zwischen den Bildpunkten.

Die Hauptachsentransformation wird in GRASS mit

```
$ i.pca
```

durchgeführt. Zunächst werden die zu transformierenden Bilder angegeben (mindestens zwei; wenn kein weiteres mehr angegeben werden soll, „return“ betätigen). Dann folgt das Namenspräfix für die zu berechnenden PCA-transformierten Bilddateien und die Option, die Farbwerte neu zu skalieren (standardmäßig wird auf 256 Farben skaliert). Die entstehenden Bilder erhalten laufende Nummern als Dateinamensendung. Ihre Anzahl entspricht der Anzahl der Eingabebilder. Wenn Sie alle Ergebnisbilder nebeneinander darstellen, erkennen Sie den abnehmenden Informationsgehalt mit steigender PCA-Kanalnummer.

Das gegenteilige Verfahren ist die „Kanonische Komponententransformation“ mit dem Modul `$ i.cca`. Dabei wird die maximale Trennung zwischen den einzelnen Kanälen angestrebt.

11.5 Verbesserung der Auflösung von Satellitenbildern

Um LANDSAT-TM-Bilder in ihrer Auflösung rechnerisch zu verbessern, kann eine Farbtransformation durchgeführt werden, die die höhere Auflösung panchromatischer SPOT-Bilder ausnutzt („Image fusion“).

Die Farbinformationen stammen aus den Kanälen tm7, tm4 und tm3, die geometrische Auflösung vom zugehörigen panchromatischen SPOT-Kanal (vgl. Abb. 37). Nach der Berechnung liegen drei Farbkanäle vor (Rot, Grün, Blau), die zu einer Rasterdatei zusammengefasst werden.
In GRASS wird wie folgt gearbeitet (Methode nach BÄHR/VÖGTLE 1991):

(a) Die Satellitenbilder sind, wie in den vorigen Kapiteln beschrieben, entsprechend den Anforderungen zu importieren bzw. zu geocodieren. Es liegen nun die LANDSAT-Kanäle tm1 bis tm7 sowie der panchromatische Kanal von SPOT vor.

(b) Vor der Farbtransformation müssen Ratiotransformationen (Verhältnisbildung aus zwei Kanälen) durchgeführt werden (vgl. auch Abschnitt A.8.3):

Um Ratio tm7/tm4 zu bilden, wird das Modul r.mapcalc verwendet:

```
$ r.mapcalc

mapcalc> ratio7.4 = 1.0 * tm7 / tm4
```

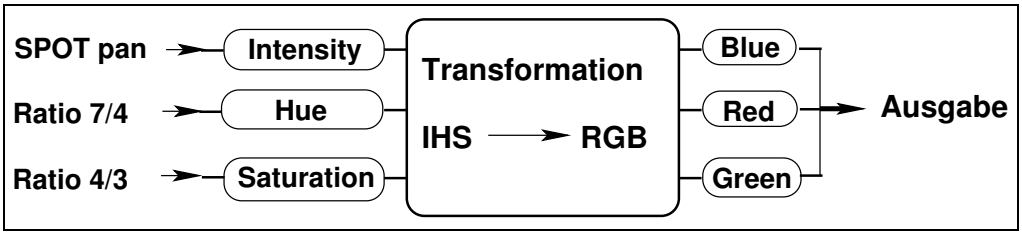


Abbildung 37: Auflösungsverbesserung von Satellitenbildern

(Wenn Sie GRASS 4.x verwenden, sollten sie die Formel mit 100 multiplizieren, um die Kommawerte der Rechnung zu erhalten.)

Als zweites wird Ratio tm4/tm3 berechnet:

```
mapcalc> ratio4.3 = 1.0 * tm4 / tm3
```

Nun wird r.mapcalc mit „exit“ verlassen.

(c) Anschließend kann die Farbtransformation durchgeführt werden:

```
$ i.his.rgb
```

Zuerst werden die Namen der Eingabekanäle für die Transformation angegeben:

1. Angabe - hue-Wert: ratio7.4
2. Angabe - intensity-Wert: spot
3. Angabe - saturation-Wert: ratio4.3

Dann folgen die Namen der drei Ausgabekanäle:

4. Angabe - Rot-Kanal: red.ihs
5. Angabe - Grün-Kanal: green.ihs
6. Angabe - Blau-Kanal: blue.ihs

Nach einiger Rechenzeit liegen die drei RGB-Kanäle vor (rot, grün und blau).

(d) Um die getrennten Farbkanäle zu einem Bild zusammenzuführen, kann man das Modul

```
$ d.rgb
```

nutzen.

Gefragt wird nach dem Rot-Kanal (hier red.ihs), Grün-Kanal (green.ihs) und nach dem Blau-Kanal (blue.ihs). Danach ist die Ausgabedatei anzugeben, die alle Kanäle enthält.

(e) Mit `$ d.rast` kann diese Ausgabedatei betrachtet werden, die Auflösung sollte sich gegenüber den LANDSAT-TM-Kanälen verbessert haben. Eventuell ist eine Kontrastverbesserung nötig (s. Abschnitt 11.2).

11.6 Fouriertransformation und inverse Fouriertransformation

Ein sehr interessantes Werkzeug für die radiometrische Analyse (spektrale Analyse) und Bildverbesserung ist die Fouriertransformation. Wenn ein Satellitenbild Störungen aufweist, sind sie auch im fouriertransformierten Bild in Form von Streifen erkennbar. Es ist nun möglich, diese Streifen aus dem transformierten Bild auszumaskieren und dann den Datensatz zurückzutransformieren. Damit werden die Störungen im Satellitenbild stark minimiert oder gar ganz ausgelöscht. Anschauliche Beispiele finden sich bei LILLESAND 1994 und JENSEN 1996.

Bei der Fouriertransformation wird ein Bild aus seiner geometrischen Darstellung (mit Koordinaten) in eine komplexe Darstellung der Frequenzanteile transformiert. Die komplexen Anteile sind in zwei Bilddateien abgelegt, die die realen und imaginären Frequenzanteile des transformierten Bildes enthalten (s. Abb. 38). Beide Anteile sind also in je einer Datei abgespeichert.
Die Interpretation eines fouriertransformierten Bildes kann sehr kompliziert sein. Zur Theorie gibt JENSEN 1996 gute Erläuterungen. Üblicherweise wird nur das reale Spektrum zur Interpretation herangezogen. Allerdings werden beide Spektren für eine spätere Rücktransformation benötigt.
Die Frequenzen sind entlang beider Axen dargestellt. Der Nullpunkt (DC-Punkt = direct current) hat die Frequenz 0,0. Er liegt bei den Koordinaten (S/2+1, S/2+1), wobei S der Bildgröße entspricht. Die Bilder müssen quadratisch und die Größe eine Potenz von 2 ($256 * 256$, $512 * 512$ etc.) sein. GRASS teilt die Bilder automatisch auf. Bei einer Bildgröße von $512 * 512$ würde der DC-Punkt also bei x=257, y=257 liegen. Je weiter man sich vom DC-Punkt entfernt, desto höher sind die Frequenzen, die sich symmetrisch um den DC-Punkt anordnen. Auch Farben werden dargestellt. Die Helligkeit eines Punktes gibt die Häufigkeit dieser Frequenz im Originalbild an.

11.6.1 Transformation und Bildfilterung

Generell handelt es sich bei den Filtern um „Binärmatrixfilter“, d.h. als Filter wird eine Matrix mit den Zahlen 0 und 1 in der Größe des Spektralbildes aufgebaut. Anschließend erfolgt eine Multiplikation beider Spektren mit diesem Filter. Dort, wo „Einsen“ stehen, bleibt das Bild erhalten, bei „Nullen“ wird die Information gelöscht. So lassen sich beispielsweise oben angesprochene Störstreifen ausmaskieren. Anschließend werden der Imaginär- und Realteil zu einem „normalen“ Bild zurücktransformiert.

In GRASS wird dieses Binärmatrixfilter über Digitalisierung erstellt. Die angesprochene Multiplikation erfolgt automatisch über das Setzen des Filters als Maske, die dann bei der Rücktransformation

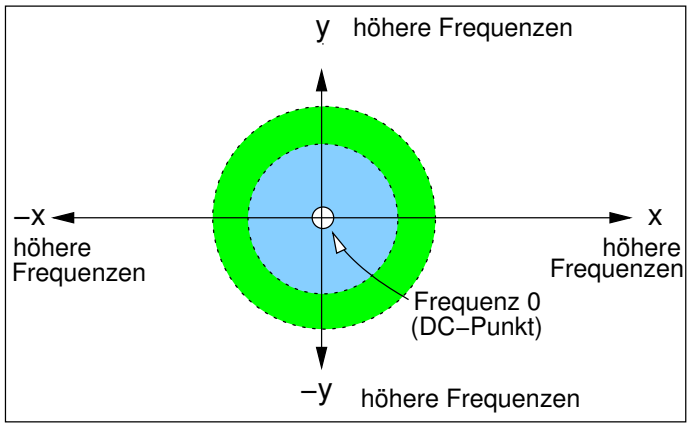


Abbildung 38: Frequenzverteilung im Fourierspektrum

berücksichtigt wird. Während bei **Störungen** Streifen ausmaskiert werden, gibt es zur **Bildverbesserung** folgende Standardfilter (vgl. auch Abb. 39, Schwarz entspricht dem Wert „1“, Weiss dem Wert „0“):

- Hochpassfilter: Herausfilterung der tiefen Frequenzen
- Tiefpassfilter: Herausfilterung der hohen Frequenzen
- Bandpassfilter: Herausfilterung der tiefen und hohen Frequenzen unter Durchlassung der mittleren Frequenzen
- Cutpassfilter: Herausfilterung der mittleren Frequenzen unter Durchlassung der tiefen und hohen Frequenzen

Konkret bedeutet das:

- Hochpassfilter: Schärfung der Kanten
- Tiefpassfilter: Betonung der Flächen
- Bandpassfilter, Cutpassfilter: Stärkung/Schwächung schmaler Strukturen

Für Störbeseitigungen sind die Digitalisierflächen länglicher Gestalt, zur Bildverbesserung müssen sie kreisförmig sein (um die gewünschten Frequenzen vollständig zu erreichen).

Zunächst muss also das fouriertransformierte Bild berechnet werden:

```
$ i.fft
```

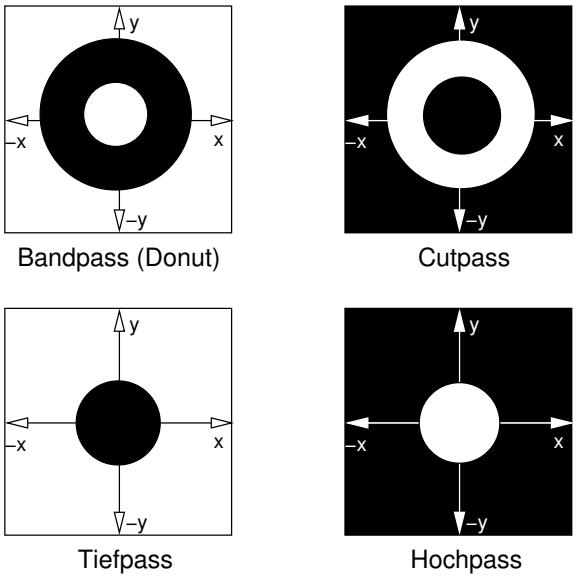


Abbildung 39: Standardfilter zur Bildverbesserung bei der Fouriertransformation

Die erste Eingabe ist der Name des Originalbilds, die zweite der Name des Realspektrums und die dritte der des Imaginärspektrums. Es entstehen also zwei neue Dateien. Der Realteil sollte nach der Berechnung mit

```
$ d.rast
```

dargestellt werden. Sind Störungen im Ausgangsbild gewesen, sollten sie jetzt als Streifen erkennbar sein. Schwierig ist, dass sich diese Streifen kaum abheben. Es kann also schnell das Problem auftreten, dass zuviel ausmaskiert wird. Ein Beispiel, in dem die auszumaskierenden Stellen markiert sind, ist in Abb. 40 zu sehen. Markiert sind die Störungen, die ausmaskiert werden können.

Nach der Darstellung des Realteils kann der „Bau“ des Filters beginnen. Dazu wird das Modul

```
$ r.digit
```

eingesetzt. Da das Realbild auf dem Monitor dargestellt ist, erhält das Filter auch gleich die richtige Größe.

Flächen lassen sich im Rasterformat digitalisieren, nachdem „A“ - „area“ angewählt wurde. Mit der Maus wird nun weitergearbeitet. Ist die Fläche vollständig (rechte Maustaste), muss eine „category number“ angegeben werden: hier also 0 oder 1. Bei 0 wird die Fläche eliminiert, bei 1 bleibt der Flächeninhalt für die spätere Rücktransformation bestehen (da ja Filter und Real-/Imaginärteil intern multipliziert werden). Das nachfolgend abgefragte Label kann ein Textkommentar sein. Es ist sinnvoll, hier „eins“ bzw. „null“ hinzuschreiben, um später Klarheit zu haben (z.B. bei Abfragen mit `$ d.what.rast` etc. zur Kontrolle). Beim Verlassen des Moduls muss der Dateiname für dieses Rasterbild angegeben werden. Mit r.digit lassen sich auch Kreise, Linien und Punkte (unendlich kurze Linie) digitalisieren.

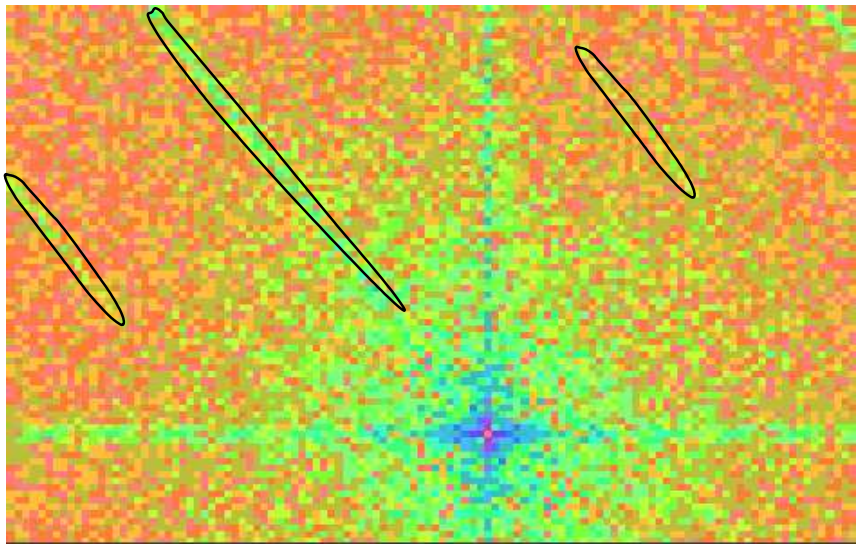


Abbildung 40: Realspektrum eines fouriertransformierten Satellitenbildes mit Störsignalen (umrahmt)

Um komplexere Filter zu erzeugen (beispielsweise ein Hochpassfilter), ist es häufig einfacher, invers zu arbeiten. D.h. die erzeugte(n) Fläche(n) werden zunächst mit dem Wert 1 versehen. Dann wird anschließend das Modul

```
$ r.mapcalc
```

aufgerufen. Mit der Operation

```
mapcalc> dummy = filter - 1
mapcalc> inverses.filter = dummy * (-1)
```

entsteht ein inverses Filter. Die „dummy“-Datei kann mit `$ g.remove` gelöscht werden.

Nun muss das Filter als Maske für die Rücktransformation gesetzt werden. Dazu verwendet man das Modul:

```
$ r.mask
```

Mit Option 2 kann die Filterdatei angegeben werden. Im nachfolgenden „Category“-Feld besteht ein alternativer Weg, gesetzte Flächenwerte (0 oder 1) umzudrehen. Nach dem Setzen des Filters kann die Rücktransformation erfolgen.

11.6.2 Rücktransformation

Für die Rücktransformation wird das Modul

```
$ i.ifft
```

benötigt. Zuerst muss der Name für den Realteil, als zweites der für den Imaginärteil angegeben werden. Als drittes wird der Name der neuen, zu erstellenden Rasterdatei erfragt.
Wichtig ist, nun die **Maske wieder zu entfernen**, die ja immer noch Gültigkeit hat. Mit

```
$ r.mask
```

und der Option 1 wird die Maske entfernt (die Datei bleibt aber bestehen und ist wiederverwendbar). Dann kann das Bild dargestellt werden.

Da das Thema sehr komplex ist, sei für weitere, v.a. theoretische Ausführungen auf die Literatur verwiesen (z.B. SCHOWENGERDT 1997).

11.7 Matrixfilter

Eine andere Filtermethode ist die „wandernde Rastermatrix“. Dabei handelt es sich um ein zeilenweise horizontal bewegtes Fenster definierbarer Größe („moving window“), in dem Kalkulationen mit allen im Fenster erfassten Rasterzellen durchgeführt werden. Es liegt auch dem Modul r.mapcalc zugrunde. Nutzbar ist dieses Verfahren für:

- Hoch- und Tiefpassfilterung (Schärfung, Weichzeichnung)
- Kantenerkennung über Richtungs- und Gradientfilter (edge detection)

- Mittelung (averaging)
- Vorbereitung zur Bildbinärisierung

Es gibt zwei Wege, diese Filter zu definieren: Sie können sowohl das Modul `$ r.mapcalc` als auch `$ r.mfilter` verwenden. Der Gebrauch von r.mapcalc ist aufwendig, da die Matrix mit relativen Koordinaten definiert werden muss: Beispielsweise ist [-1,1] die linke obere Ecke, [0,1] das rechte mittlere Feld in einer 3x3-Matrix. Mehr Informationen finden Sie dazu im „r.mapcalc-analysis“-Tutorial (s. Literatur).
Der zweite, einfachere Weg mit r.mfilter geht über die Matrixdefinition in einer Textdatei. Sie wird zusätzlich zum Eingangsrasterbild und Namen für das zu erzeugende Rasterbild als Filter angegeben.
Ein Beispiel für ein 7x7-Mittelwertfilter, das scharfe Übergänge in einer Rasterdatei filtert (so in einer Textdatei zu speichern):

```
TITLE     7x7 Tiefpass
MATRIX    7
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
1 1 1 1 1 1 1
DIVISOR   49
TYPE      S
```

Der Mittelwert bleibt erhalten, wenn die Summe der Filterwerte gleich der Zeilenzahl $*$ Spaltenzahl entspricht.
Es gibt sequentielle und parallele Filter. Bei sequentiellen Filtern (TYPE S) gehen die veränderten Rasterzellenwerte in der Nachbarschaft der zu berechnenden Mittelzelle aus der neuen Karte ein, bei parallelen Filtern (TYPE P) werden die Rasterzellenwerte der Originalkarte verwendet. Richtungsfiltern sind als parallele Filter aufzubauen. Näheres dazu steht in den Manualseiten zu r.mfilter.
In obigem Beispiel wird jede Zelle im „moving window“ mit 1 multipliziert, die Ergebnisse summiert und durch 49 geteilt (Summe von $7 * 7 * 1$).
Ein Hochpassfilter, das zur Bildschärfung dient, kann folgendermaßen programmiert werden:

```
TITLE     5x5 Hochpass
MATRIX    5
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1
```

```
-1 -1 24 -1 -1
-1 -1 -1 -1 -1
-1 -1 -1 -1 -1
DIVISOR   25
TYPE      S
```

In diesem Beispiel wird die Mittelzelle im Fenster mit 24 gewichtet, die anderen Zellen entsprechend anders. Die gesamte Matrix wird zum Schluss durch 25 geteilt und die Werte in einer neuen Datei abgelegt.
Für eine Datenfilterung wird also zuerst die Textdatei mit der Filterdefinition erzeugt, dann
`$ r.mfilter`
gestartet. Nach Angabe von zu bearbeitender Rasterdatei und neuem Namen für die Ausgabe muss die Filtertextdatei genannt werden. Sie sollte im aktuellen Verzeichnis liegen. Die restlichen Optionen können übernommen werden. Die einzige Limitierung gegenüber r.mapcalc ist, dass nur ganze Zahlen im Filter eingesetzt werden dürfen. Wenn Sie Fließkommazahlen oder trigonometrische Funktionen verwenden wollen, sollten Sie r.mapcalc einsetzen. Auch für eine anschließende Schwellwertbinärisierung ist r.mapcalc bestens geeignet (if-Bedingung).

Die Farbtabelle können Sie mit `$ r.colors` wieder auf „Grau“ setzen, sofern nötig. Eine Filterdefinitionsdatei kann auch mehrere Filter enthalten, die dann nacheinander auf das Bild angewendet werden.

12 Orthofoto-Herstellung aus Luftbildern

In den letzten Jahren hat sich die satellitengestützte Fernerkundung in großen Schritten fortentwickelt, doch sie kann weiterhin konventionelle Luftbilder nicht ersetzen. Beide Aufnahmesysteme existieren parallel nebeneinander und dienen unterschiedlichen Aufgabenstellungen.
Durch eine digitale Verarbeitung im GIS können Luftbilder auch in teilautomatisierte Auswertungsprozesse eingebunden werden. Mit GRASS lassen sich zum Teil sogar kostspielige fotogrammetrische Geräte ersetzen, wie beispielsweise bei der Produktion von Orthofotos.

12.1 Grundlagen

Luftbilder werden für eine Vielzahl von Zwecken eingesetzt, neben der Herstellung thematischer Karten können Flächen- und Streckenmessungen im großen Maßstab (im Vergleich zu satellitengestützten Methoden) durchgeführt werden. Weitere Einsatzmöglichkeiten bestehen in der Aktualisierung von Karten, Biotopkartierungen, Ermittlung versiegelter Flächen für Regenwasserversickerung oder in anderen Projekten. Im Folgenden wird gezeigt, wie Sie, ausgehend von einem analog (also als Diapositiv) vorliegenden Luftbild, ein digitales Orthofoto mit GRASS herstellen können. In diesem Kapitel geht es ausschließlich um Nadir- und Senkrechtluftbilder, auf Schrägaufnahmen (z.B. aus der Luftbildarchäologie) lässt sich die hier beschriebene Methode modifiziert jedoch auch anwenden.
Nimmt man einmal an, das Bildflugzeug bewege sich in idealer Weise über dem zu fotografierenden Gebiet, so steht das Lot einer vom Flugzeug nach unten schauenden Kamera senkrecht auf der Erdoberfläche. Man spricht von einer „Nadiraufnahme“. Die Lotlinie wird als Aufnahmeachse (*optical axis, plumb line*) bezeichnet. Bis zu einer Lotabweichung von 3° in jede Richtung handelt es sich um eine „Senkrechtaufnahme“ (LÖFFLER 1994:86), darüber hinaus wird von einer Schrägaufnahme gesprochen.
Der Nadir (*nadir, plumb point*) ist der Punkt auf der Erdoberfläche, der senkrecht unter der aufnehmenden Kamera liegt. Es ist der Berührpunkt der Aufnahmeachse mit der Erdoberfläche. Wird ideal ohne Abweichung von dieser Lotlinie fotografiert, liegt der Nadir im Bildmittelpunkt. Dem Luftbild liegt die Zentralprojektion zugrunde. Auch bei der oben angesprochenen Senkrechtaufnahme (mit geringer Neigung des Flugzeugs) kann diese Projektion angenommen werden.

Es ergeben sich bei einem Luftbild gegenüber einer Karte (Orthogonalprojektion) wesentliche Unterschiede: Durch die Zentralprojektion im Luftbild ist das Luftbild nur im Bildmittelpunkt (Nadir)

unverzerrt, es kommt vom Bildmittelpunkt zum Bildrand zu einer sich steigernden verzerrten Darstellung (HILDEBRANDT 1996:151). Diese Abweichungen verstärken sich bei unebenem Gelände (vgl. Abb. 41) und Neigung des Flugzeugs.
Der mittlere Bildmaßstab m_b (bzw. die Bildmaßstabszahl M) ist höhenabhängig und kann folgendermaßen berechnet werden:

$$M = \frac{h_g}{c_k} \tag{12.1}$$

$$m_b = \frac{1}{M} \tag{12.2}$$

Dabei ist h_g [in Metern] die Höhe über Grund und somit reliefabhängig: Von der im Luftbild angegebenen Flughöhe über Meeresspiegel muss die Geländehöhe subtrahiert werden. Die Kammerkonstante c_k der Kamera (identisch mit der Brennweite f) ist im Luftbild in Millimeter angegeben und für diese Formel in Meter umzurechnen. M gibt die Bildmaßstabszahl an. Der interessierende Bildmaßstab m_b ergibt sich aus Formel 12.2. Es ist zu beachten, dass im Luftbild höhere Geländepunkte in einem größeren Maßstab abgebildet werden als tieferliegende (HILDEBRANDT 1996:152). Außerdem findet durch die Zentralprojektion ein horizontaler Versatz statt. So lässt sich mit der Formel je nach Relieftyp bestenfalls ein mittlerer Bildmaßstab ermitteln.

Ist das fotografierte Gelände „eben“, kann auch ein unentzerrtes Luftbild zu Strecken- und Flächenmessungen herangezogen werden. In diesem Fall werden die nur geringen Verzerrungen vernachlässigt. Eine Beurteilung, ob es sich um ein derartiges Luftbild handelt, lässt sich aus dem Verhältnis der Höhenunterschiede im Luftbild zum Luftbildmaßstab errechnen (SCHNEIDER 1975, in BIERHALS 1988:91). Die Höhenunterschiede im Luftbild dürfen nicht größer sein als die Bildmaßstabszahl geteilt durch 500. Wurde beispielsweise eine Luftbildbefliegung im Maßstab 1:10.000 durchgeführt, sind maximal 20 Meter Höhendifferenz im Luftbild erlaubt

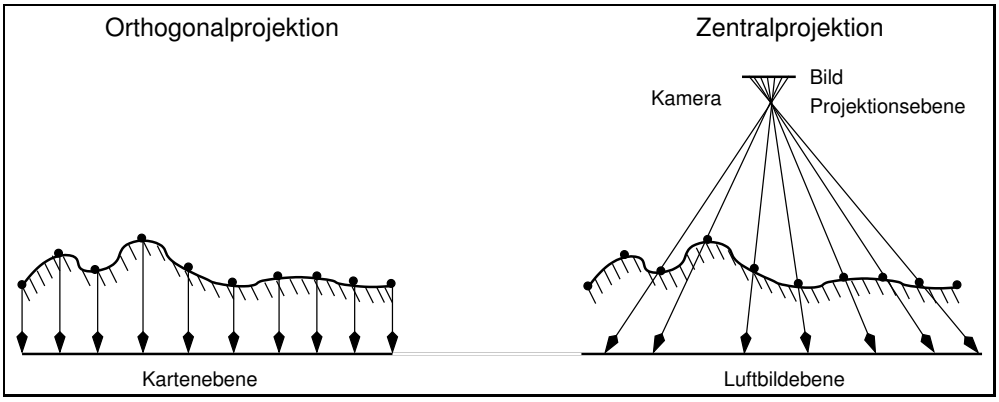


Abbildung 41: Geländeabbildung auf Kartenebene und Luftbildebene (nach ALBERTZ 1991:66, leicht verändert)

(10000/500=20), um direkte Strecken- oder Flächenmessungen durchführen zu können. Andernfalls ist ein Orthofoto herzustellen, wie im Folgenden vorgestellt.

Weist das Flugzeug eine Neigung aus der Horizontalen auf, ist der Bildmaßstab in der Bildhälfte, deren Abstand zum Flugzeug sich verkürzt hat, bei ebenem Gelände geringer als in der weiter entfernten Bildhälfte. Unebenes Gelände trägt zu einer weiteren Variation bei.

Bei vorhandener leichter Neigung des Flugzeugs (Senkrechtaufnahme) ist der Bildmittelpunkt nicht mehr mit dem Nadir identisch. Der Bildmittelpunkt (Bildhauptpunkt, *point of symmetry, principal point*), ergibt sich, indem sich gegenüberliegende Rahmenmarken (*fiducial points, reseau marks*) miteinander durch Linien verbunden werden. Der Nadir liegt davon versetzt, dieser Winkel wird als Nadirdistanz bezeichnet (vgl. Abb. 42). Um für diesen Fall den Nadir zu ermitteln, kann man die Fluchtlinien ideal senkrecht auf der Erdoberfläche stehender Objekte verbinden.

Die Erdkrümmung ist ein weiterer Faktor, der das Luftbild beeinflusst. Sie kommt aber erst bei raumfahrtgestützten Aufnahmen (KVR 1000 etc.) zum Tragen. Zu erwähnen ist auch, dass Schrumpfung und Ausdehnung des Filmmaterials ebenfalls zu Verschiebungen führen können.

Was hat also ein Orthofotomodul zu leisten? Es muss

- die i.a. vorhandene Neigung der Flugzeugs aus der Horizontalebene ausgleichen,
- die Zentralprojektion in eine orthogonale Kartenprojektion umrechnen und dabei durch Höhenunterschiede bedingte Maßstabsabweichungen umrechnen, sowie

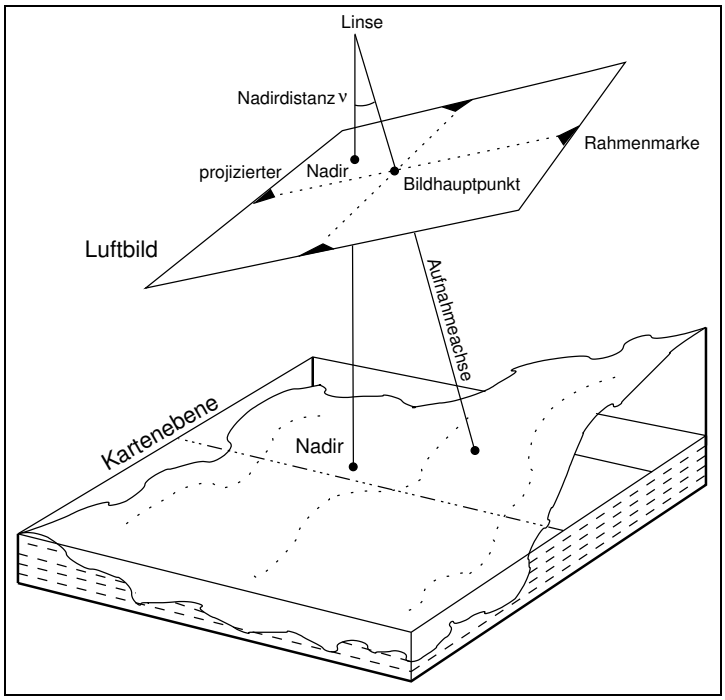


Abbildung 42: Bezeichnungen im Luftbild (nach LÖFFLER 1994:89, verändert)

- das Luftbild um rund 90° bzw. 270° in Nordrichtung drehen, da Bildflüge üblicherweise in West-Ost- und Ost-West-Richtung durchgeführt werden (damit liegen Bilder bei einer stereoskopischen Betrachtung nebeneinander). Bei Nord-Süd-Flügen ergeben sich entsprechend andere Winkel.

12.2 Vom Luftbild zum Orthofoto

Im GIS brauchen Sie also nicht nur ein Luftbild, sondern auch Informationen über das Relief – eine Datei mit geokodierten Höhendaten. Um eine Geokodierung des Luftbilds vornehmen zu können, darf natürlich als dritte „Zutat“ ein topographischer Bezug, i.a. also eine Karte, nicht fehlen. Dabei sollte der Maßstab der Karte sinnvollerweise zum mittleren Luftbildmaßstab passen oder größer sein. Aus der Karte werden später die Koordinaten einiger Referenzpunkte gelesen und im Luftbild im GRASS-Monitor markiert. Dem Luftbild selbst kann Uhrzeit, Flughöhe über Meeresspiegel, Neigung (Dosenlibelle) und Kammerkonstante aus den Nebenabbildungen (Datenstreifen) entnommen werden. Der mittlere Bildmaßstab lässt sich aus der Bildmaßstabszahl m_b nach Formel 12.2 ermitteln.

Prinzipiell ist zwischen Pseudoorthofotos und „echten“ Orthofotos zu unterscheiden. Pseudoorthofotos sind einfacher zu erstellen, bei ihnen werden Gebäudeverzerrungen nicht korrigiert. Die Schrägansichten der Gebäude oder anderer Objekte mit signifikanter Höhe bleiben unkorrigiert. Bei „echten“ Orthofotos sind dagegen alle Gebäude nur noch in der Aufsicht erkennbar, es gibt damit einen entsprechenden sichttoten Bereich auf den vom Luftbildmittelpunkt abgewandten Gebäudeseiten. Um „echte“ Orthofotos herstellen zu können, wird ein Höhenmodell mit integrierten Gebäudehöhen benötigt. Das kann beispielsweise durch Laserscanbefliegungen erzeugt werden. Im Allgemeinen stehen aber nur Geländehöhenmodelle zur Verfügung, die keine Objekthöhen enthalten. Daher ist das Pseudoorthofoto (noch) der Regelfall.

Zunächst werden im GIS die Parameter der benutzten Kamera (Reihenmesskammer) spezifiziert. Diese Daten werden üblicherweise mit dem Luftbild mitgeliefert oder können, wenn der Kameraname bekannt ist, z.T. der Literatur entnommen werden (z.B. HILDEBRANDT 1996:80f.). Nach der Eingabe der topographischen Referenzpunkte (Passpunkte, *GCPs - ground coordinate points*) findet im Rechner die Transformation von einem xy-Koordinatensystem (mit internem Bezug der Bildkoordinaten in Millimeter) in Landeskoordinaten (z.B. Gauß-Krüger-Koordinaten) statt – als Ergebnis steht das gedrehte, entzerrte Orthofoto.

12.3 Die Umsetzung in GRASS

Vorbereitend muss das Luftbild auf einem Durchlichtscanner gescannt werden, falls es nur analog vorliegt. Um aus einem normalen Scanner einen Durchlichtscanner zu machen, ist ein spezieller

Aufsatz (Zubehör) nötig.

Aus dem mittleren Bildmaßstab (vgl. Formel 12.1) kann man errechnen, welche Auflösung in dpi am Scanner eingestellt werden muss, um eine bestimmte Bodenauflösung zu erreichen. Die geometrische Auflösung am Boden R_g in Bezug auf einen Maßstab ergibt sich über die Maßstabszahl M und die Scanauflösung R_s:

$$R_g[cm] = \frac{M}{R_s[Zeilen/cm]} \tag{12.3}$$

Ein Beispiel:

Der Maßstab des zu scannenden Luftbildes sei 1:10.000. Die Maßstabszahl ist demnach M = 10000cm. Als geometrische Auflösung am Boden R_g für das Luftbild werde beispielsweise eine Rasterzellenlänge/-breite von 40cm/Rasterzelle angestrebt. Nach Umstellung der Formel 12.3 ergibt sich demnach für die einzustellende Scannerauflösung R_s:

$$\begin{aligned} R_s[Zeilen/cm] &= \frac{M}{R_g[cm]} \\ &= \frac{10000}{40\,cm} = 250\,\frac{1}{cm} \end{aligned} \tag{12.4}$$

$$R_s[dpi] = \frac{M}{R_g[cm]} * 2.54\,\frac{cm}{in} = 250\,\frac{1}{cm} * 2.54\,\frac{cm}{in} = 635\,dpi \tag{12.5}$$

In diesem Fall muss der Scanner demgemäß auf 635dpi eingestellt werden. Da es sich nur um einen mittleren Maßstab im Luftbild handelt, kommt es unweigerlich zu Abweichungen. Diese werden über die Transformationen und Interpolationen im Orthofotomodul von GRASS ausgeglichen. Sie sollten auch bedenken, dass eine Scangenauigkeit von beispielsweise 40cm keine Objekte dieser Größe erkennen lässt! Zur Objekterkennung sind immer mehrere Pixel nötig.

Beim Scannen ist zu überlegen, ob mit 256 Farben (8bit) oder einer höheren Farbauflösung gescannt werden soll. Der Platzbedarf steigt entsprechend: Für ein Standardluftbild von 23cm * 23cm ergibt sich, wenn mit 1200dpi bei 24bit Farbtiefe gescannt, ein Datenvolumen von über 300MB pro Bild. Nach dem Scannen sollte mit einem Bildverarbeitungsprogramm (z.B. xv) überprüft werden, ob die hellen Markierungspunkte innerhalb der Rahmenmarken erkennbar sind. Diese sind später für den Bildorientierungsvorgang essentiell wichtig. Daran ist also die Mindest-Scanauflösung zu knüpfen. Sollten diese Rahmenmarken gänzlich im Luftbild fehlen, können Sie ersatzweise mit den Ecken des Luftbilds arbeiten. Außerdem sollten Sie sich die Werte für Flughöhe, Kammerkonstante c und Uhrzeit der Aufnahme aufschreiben (mit „Zoom“ gut erkennbar).

12.3.1 Erstellung der Gauß-Krüger-*location*

Das Rohluftbild wird während seiner Entzerrung in eine *location* mit Koordinatensystem transformiert. Diese *location* muss vorher eingerichtet werden, da hier auch die bezuggebende Karte und das Höhenmodell abzulegen sind. Zunächst müssen Sie daher diese *location* für das Höhenmodell und die (gescannte) geocodierte topographische Karte einrichten. Als Grundauflösung sollte die größte vorkommende Genauigkeit gewählt werden (zum Beispiel der Wert der Ziel-Bodenauflösung für das oder die Luftbilder). Es empfiehlt sich, als Auflösung *(grid resolution)* in dieser Ziel-*location* Werte größer Eins zu verwenden. Wird eine Bodenauflösung unterhalb eines Meters angestrebt, ist für die GRASS-*location* diese Auflösung in Zentimetern anzugeben. Näheres zur Einrichtung ist im Kapitel 4 beschrieben. Es kann generell natürlich auch eine andere Projektion als Gauß-Krüger verwendet werden.

Nach dem Einrichten der Ziel-*location* werden das Höhenmodell und die topographische Karte importiert. Vor dem Import jeder Karte muss die kartenspezifische Auflösung x (mit der sie gescannt bzw. digitalisiert wurde, z.B. in Metern) in der *location* gesetzt werden. Ist die *location* in Zentimetern definiert, muss entsprechend umgerechnet werden. Das Setzen der Auflösung vor dem Import erfolgt so:

```
$ g.region res=x
```

Dann kann mit `r.in.ascii`, `r.in.gif`, `r.in.tif`, `r.in.arctiff` (ab GRASS 4.2.1) etc. die Karte importiert werden. Nach dem Import besitzt die Karte zunächst ihre Pixelausdehnung als Randkoordinaten (Ausnahme: `r.in.arctiff`), deshalb ist anschließend `r.support` zur Zuweisung der korrekten Gauß-Krüger-Randkoordinaten anzuwenden („Edit header“: yes). Nun sollte sich die Karte auf dem GRASS-Monitor visualisieren lassen.

Höhendaten befinden sich im Allgemeinen in ASCII-Dateien mit Hoch-, Rechtswert und Höhenangabe. Sie können mit `s.in.ascii.dem` (GRASS 4.x) bzw. `r.in.arc` oder `s.in.ascii` (GRASS 5.0.x) importiert werden. Stimmt die Auflösung der Höhendaten schon mit der Zielauflösung überein, lassen sich die Höhendaten mit `s.to.rast` direkt in das Rasterformat als Höhenmodell umwandeln. Sollte erforderlich sein, sie in eine andere Auflösung zu interpolieren, so ist diese zunächst mit `g.region` festzulegen. Anschließend kann das Höhenmodell (im Rasterformat) mit `s.surf.idw` (GRASS 4.x) bzw. `s.surf.rst` (GRASS 5.0.x) aus den Höhenpunkten interpoliert werden.

Wichtig ist, vor dem Verlassen der Gauß-Krüger-*location* die Zielauflösung einzustellen, die das Orthofoto später erhalten soll. In diesem Beispiel wären das also 40cm. Außerdem sollte in der Ziel-*location* nicht gezoomt werden, da die Einstellungen für den Transformationsprozess genutzt werden. Das Rücksetzen auf Standardeinstellung, aber gegebenenfalls abweichender Auflösung, erfolgt mit (hier x=40cm):

`g.region -d res=x`, Kontrolle mit `g.region -p`

12.3.2 Erstellung der xy-Luftbild-*location*

Für die xy-*location*, die das oder die Rohluftbilder aufnehmen soll, wird GRASS neu aufgerufen. Da problemlos mehrere Luftbilder quasi „übereinander“ in dieselbe xy-*location* importiert werden können, sind Länge und Breite großzügig zu bemessen. Allgemein werden der West- und der Südwert auf Null gesetzt, Nord- und Ostkoordinaten erhalten positive Werte. Die Auflösung beträgt wie üblich in xy-*locations* 1 Pixel in beide Richtungen.
Importieren können Sie gescannten Luftbilder mit den Modulen `r.in.gif` (8bit), `r.in.tiff` (8/24bit) bzw. `r.in.ppm` (24bit). Diese *location* braucht nicht verlassen zu werden, da von hier aus auch die Transformation stattfindet.

12.3.3 Die Orthofotoherstellung

Es liegen nun sowohl das Höhenmodell und die topographische Karte als auch die gescannten Luftbilder vor (in zwei *locations*). Für die Benutzung des Orthofotomoduls wird zunächst ein GRASS-Monitor (`d.mon`) gestartet, dann geht es weiter mit der Erzeugung einer Bildgruppe. Darin wird nur ein Bild „angemeldet“, nämlich das zu entzerrende Luftbild. Rufen Sie dazu `$ i.group` auf, um eine neue Bildgruppe zu erzeugen und das Luftbild auszuwählen. Eine „Subgroup“ wird nicht benötigt.
Dann wird das Orthophotomodul aufgerufen:

```
$ i.ortho.photo
```

Geben Sie zunächst die gerade erzeugte Bildgruppe an. Dann sehen Sie das Hauptmenü. Im Prinzip werden die Menüpunkten der Reihe nach abgearbeitet. Die Arbeit mit dem Modul `i.ortho.photo` kann übrigens jederzeit unterbrochen und später fortgesetzt werden. Alle Einstellungen speichert das Modul automatisch.

1. Select/Modify imagery group

Diesen Punkt haben Sie bereits erledigt. Wenn nötig, könnten Sie hier eine andere Bildgruppe wählen.

2. Select/Modify imagery group target

Auch bei der Angabe des Transformationsziels gibt es nichts Neues zu erlernen: Geben Sie hier als *location/mapset* die zuvor erzeugte Gauß-Krüger-*location* an (bzw. Ihrer Ziel-*location* mit einer anderen Projektion). Dieser Menüpunkt entspricht `i.target` (vgl. Abschnitt 11.1.2).

3. Select/Modify target elevation model

Das in der Ziel-*location* gespeicherte Höhenmodell wird hier mit seinem Namen angegeben.

4. Select/Modify imagery group camera

Nun wird es etwas „fotogrammetrischer“ – die Kameradaten sind zu definieren. Sofern keine Kameradatei bereits existiert, die Sie nutzen könnten, geben Sie einen neuen Namen ein. Hier soll als Beispiel eine Reihenmesskammer der Firma Zeiss benutzt werden, als Namen wählen den passenden Namen, beispielsweise „rmk-top“. Nun erscheint folgender Bildschirm:

```
     Please provide the following information:
+-----------------------------------------------------------+
 Camera Name                                  DBA SYSTEMS CAMERA_
 Camera Identification                        _____________________
 Calibrated Focal Length mm.                  0____________________
 Point of Symmetry: X-coordinate mm.          0____________________
 Point of Symmetry: Y-coordinate mm.          0____________________
 Maximum number of fiducial or reseau marks 0____________________
+-----------------------------------------------------------+
```

Als *Camera Name* geben Sie beispielsweise „Zeiss“ an (der vorhandene Eintrag wird überschrieben), als *Camera Identification* „RMK-TOP“ ein. Als *Calibrated Focal Length* wird nun die mit dem Luftbild mitgelieferte (oder die notfalls der Literatur entnommene typische) Kammerkonstante angegeben (vgl. S. 177). Die kalibierte Kammerkonstante ist messkammerspezifisch und wird vom Hersteller individuell ausgemessen. Verwendet werden soll hier der typische Wert von 305,07mm (als 305.07 wegen der U.S.-amerikanischen Notation einzugeben). Als Koordinaten für den *Point of Symmetry* wird jeweils 0 angegeben, er ist der Ursprung der Bildkoordinaten. Ist in der Kamerabeschreibung ein abweichender Wert (auch in Millimetern) angegeben, der vom Hersteller durch Messung ermittelt wurde, ist dieser einzusetzen. Die Anzahl der Rahmenmarken (*Maximum number of fiducial or reseau marks*) ist aus dem Luftbild leicht ersichtlich, i.a. sind vier Rahmenmarken, jeweils in den Bildrandmitten gelegen, vorhanden.

So sieht das Formular typischerweise aus:

```
     Please provide the following information:
+-----------------------------------------------------------+
 Camera Name                                  Zeiss RMK-TOP__
 Camera Identification                        _____________________
 Calibrated Focal Length mm.                  305.07_____________
 Point of Symmetry: X-coordinate mm.          0____________________
 Point of Symmetry: Y-coordinate mm.          0____________________
 Maximum number of fiducial or reseau marks   4____________________
+-----------------------------------------------------------+
```

Der Bildschirm sieht in unserem Beispiel also so aus:

```
      Please provide the following information:
+-------------------------------------------------------------+
 Camera Name                                  Zeiss_____________
 Camera Identification                        RMK-TOP___________
 Calibrated Focal Length mm.                  305.07____________
 Point of Symmetry: X-coordinate mm.          0_________________
 Point of Symmetry: Y-coordinate mm.          0_________________
 Maximum number of fiducial or reseau marks 4_________________
+-------------------------------------------------------------+
```

Nun müssen in einem neuen Bildschirm die Koordinaten der Rahmenmarken angegeben werden. Als Bezug wird dabei der *Point of Symmetry*, also die Bildmitte (bzw. die laut Hersteller leicht verschobene Bildmitte, s.o.) genommen. Die Numerierungsreihenfolge ist in Abbildung 43 angegeben. Bei einem idealen 23cm $*$ 23cm Luftbild befinden sich die hellen Punkte in den Rahmenmarken 113mm vom Bildmittelpunkt (vgl. S. 178) entfernt. Diese Werte sollten ausgemessen werden bzw. liefert der Kamerahersteller mit. In Abbildung 43 sind die Marken und der zu messende Abstand zum Bildmittelpunkt dargestellt. Entsprechend des Koordinatensystems mit dem Ursprung in der Bildmitte ergeben sich folgende Werte für unser Luftbild (alle Angaben in Millimetern):

```
                Please provide the following information:
+-------------------------------------------------------------+
          Fid#     Fid Id          Xf              Yf

            1      1_____      113_______      0_________
            2      2_____      -113______      0_________
            3      3_____      0_________      113_______
            4      4_____      0_________      -113______

                              Next: end__
+-------------------------------------------------------------+
```

Damit sind alle Daten der Reihenmesskammer erfasst.

5. Compute image-to-photo transformation

An dieser Stelle werden die Bildkoordinaten (in Pixeln des gescannten Bildes in der GRASS-*location*) mit den Luftbildkoordinaten („analoge Koordinaten“ in Millimetern) verknüpft.
Sollte doch noch kein GRASS-Monitor gestartet worden sein, kann das Modul `i.ortho.photo` zu diesem Zweck verlassen und anschließend neu aufgerufen werden. Jetzt geht es im GRASS-Monitor weiter mit der Auswahl des zu bearbeitenden (unentzerrten) Luftbilds. Automatisch werden die Kameraparameter eingeblendet. Das Orthofotomodul erwartet nun die Zuordnung der Rahmenmarken in der Tabelle zu den Marken im Bild (Digitalisierung). Damit zeigt sich, dass die Nummerierung prinzipiell individuell sein kann, sofern hier richtig zugeordnet wird.

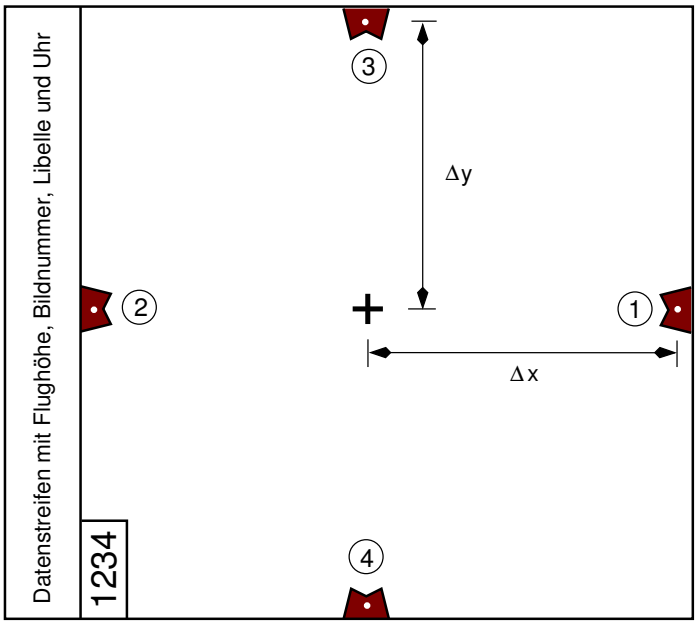


Abbildung 43: Rahmenmarken im Luftbild

Mit der Zoom-Funktion wird nun Marke 1 (rechter Bildrand) vergrößert, dann innerhalb der Marke der helle Markierungspunkt. Ist hier kein Punkt erkennbar, war die Auflösung beim Scannen zu niedrig. In diesem Fall muss leider von vorne begonnen werden. Nun wird der Mittelpunkt des Markierungspunktes in der vergrößerten Rahmenmarke mit der Maus markiert und per Doppelklick der zugehörige Punkt in der Tabelle angewählt. Im Textfenster (xterm) müssen Sie den Punkt mit „y“ (yes) bestätigen, sofern die Zuordnung sinnvoll ist. Die Markierung im Monitor färbt sich grün, wenn der Passpunkt von Ihnen akzeptiert wird.
In dieser Weise wird mit allen Marken verfahren. Der Menüpunkt „Analyze“ im Monitor erlaubt abschließend die Beurteilung der Digitalisierung. Der RMS-error sollte gegen Null streben bzw. höchstens bei Eins liegen. Ein ungenau digitalisierter Punkt kann durch Doppelklick mit der Maus in der entsprechenden Passpunktzeile in der „Analyze“-Tabelle ausgeschaltet werden, um ihn erneut nach Zoomen der jeweiligen Marke zu digitalisieren.
Sind alle Rahmenmarken erfolgreich zugeordnet, geht es mit „Quit“ wieder in das Hauptmenü von `i.ortho.photo`

6. Initialize exposure station parameters

Menüpunkt (6) wird nur für Schrägaufnahmen benutzt und wird hier nicht weiter erläutert. Es kann also gleich mit Menüpunkt (7) fortgesetzt werden.

7. Compute ortho-rectification parameters

Nachdem xy-*location* und das unreferenzierte Luftbild verknüpft sind (Menüpunkt (5)), können jetzt Passpunkte gesetzt werden, um das Luftbild zu geocodieren. Sinnvolle Ergebnisse erhält

man bereits mit zwölf gut verteilten Passpunkten (SHAPIRO ET AL. 1991). Die zugehörigen Geländehöhen, die ja essentiell sind für die Herstellung eines Orthofotos, werden automatisch aus der Höhendaten-Rasterdatei in die internen Berechnungen übernommen.

Rufen Sie also Menüpunkt (7) auf und wählen das zu bearbeitende Luftbild im GRASS-Monitor aus. Mit „PLOT CELL“ und Anklicken der rechten Monitorhälfte kann dort die topographische Karte (TK) angezeigt werden. Nun müssen (wie bei `i.points`) korrespondierende Passpunkte durch geschicktes Zoomen und Mausdigitalisierung in Luftbild und TK digitalisiert werden. Ist ein Passpunkt jeweils im Luftbild und in der TK erfolgreich markiert, wird dieses mit „y“ (yes) im X-terminal bestätigt. Als leichte Schwierigkeit ergibt sich anfänglich die Drehung des Luftbilds um $\pm$ 90° zur TK. Doch nach kurzer Gewöhnungszeit sollte dieser Umstand keine Probleme mehr bereiten.

Sinnvolle Passpunkte sind aufgrund der Generalisierungen in den topographischen Karten Kreuzungsmitten und die Mitten von Einzelobjekten (Baumsignaturen etc.). Vorsicht ist bei Gebäuden geboten: Im Luftbild sind meistens sowohl Dachkante als auch Fußpunkt zu sehen – aufgrund der Parallaxe im Bild dürfen höchstens die Fußpunkte verwendet werden, aber nicht die verschobenen Dachkanten.
Wenn Sie ein „echtes“ Orthofoto herstellen können und eine Höhenmodell mit integrierten Gebäudehöhen sowie eine grundrisstreue Karte haben, sind die Hausdächer ebenfalls zu geocodieren. In diesem Fall werden sowohl die Gebäudefußpunkte als auch die Dachecken referenziert.

Sind mehr als vier Passpunkte gesetzt, kann ihre Lagequalität mit dem Menüpunkt „Analyze“ untersucht werden. Nun erscheint eine Tabelle aller Passpunkte mit zugehörigem Lagefehler (RMS-error). Dieser RMS-error errechnet sich für die Passpunkte aus den aktuellen Transformationsgleichungen (von Kamera etc. abhängig). Der Wert wird für die Zielkoordinaten errechnet, ist also in Zentimetern oder Metern je nach Ziel-*location* angegeben. Hat ein Passpunkt eine zu starke Abweichung vom theoretischen Wert, der sich aus den Transformationsgleichungen ergibt, erscheint diese Zeile in roter Farbe. Soll ein Passpunkt inaktiviert werden, da seine Qualität nicht ausreichend ist, kann er per Doppelklick in der ANALYSE-Tabelle „stillgelegt“ werden. Akzeptabel ist ein summierter RMS-error im Wert einer Ziel-*location*-Grundeinheit, z.B. von einem Meter, wenn in der Ziel-*location* eine Rasterzelle diese Seitenlänge hat.

Als Option für eingemessene Passpunkte (z.B. mit GPS) können alternativ auch Koordinaten eingegeben werden. Dazu wird der jeweilige Passpunkt im Luftbild markiert und anschließend als „input method“ statt „screen“ das „keyboard“ gewählt.

Mit „Quit“ gelangt man nach Digitalisierung einer ausreichenden Anzahl von Passpunkten (mindestens zwölf) wieder in das Hauptmenü.

8. Ortho-rectify imagery files

Abschließend erfolgt die vollständige Entzerrung des Luftbildes. Dazu wird Menüpunkt (8) gewählt. Im nun erscheinenden Bildschirm geben Sie den neuen Namen für das zu transformierende Orthofoto in der Ziel-*location* an. Waren vorher mehrere Luftbilder in die Bildgruppe aufgenommen worden (Menüpunkt (1)), sind hier mehrere Luftbilder aufgelistet. Natürlich gelten die gesetztenPasspunkte nur für das aktuell bearbeitete Luftbild, deshalb geben Sie ausschließlich den Namen dieses Bildes an. Nun werden zunächst Transformationsgleichungen aufgestellt („Computing equations...“), dann geht es weiter. Mit *„1. Use the current window in the target location“* beginnt der Rechner mit der Entzerrung. GRASS meldet sich per email, wenn der Prozess abgeschlossen ist. Bis dahin kann man GRASS verlassen, neu aufrufen (nur nicht mit der Ziel-*location*) oder parallel andere Arbeiten am Rechner erledigen.

Typische Rechenzeiten sind für einen Linux-PC/ 200MHz/ 64MB: rund 0,6Mcell/min. So braucht GRASS für ein gescanntes Luftbild mit $2500 * 2500$ Punkten in 8bit mit 2,50m Auflösung rund neun Minuten.
Bei $9000 * 7500$ Punkten und einer Zielauflösung von 1m dauert es mit einem Linux-PC/ 200MHz/ 64MB wesentlich länger: rund 2 Stunden (rund 0.55Mcell/min). Glücklicherweise handelt es sich um ein sich quasi selbst lösendes technologisches Problem.

Ist die email im elektronischen Briefkasten eingetroffen, können Sie GRASS mit der Ziel-*location* starten und das fertige Orthofoto betrachten. Eine Überlagerung mit der topographischen Karte zeigt die Qualität der Entzerrung an.

13 Hinweise zur Programmierung in GRASS

GRASS bietet aufgrund des modularen Aufbaus großes Potenzial zur Programmierung. Es wird in „Script-Programmierung“ und „C-Programmierung“ unterschieden. Seit 1999 gibt es auch erste Entwicklungen für eine JAVA-Schnittstelle.[1]
Für die Script-Programmierung sind keine expliziten Programmierkenntnisse notwendig, es handelt sich vielmehr um die automatisierte Anwendung von GRASS-Modulen in Verbindung mit weiteren UNIX-Werkzeugen. Eine interessante Anwendung dieser Fähigkeit von GRASS ist das Internet-GRASS „GRASSLinks“. Es basiert auf UNIX-Shell-Scripts (CGI-Scripts), die eine dynamische Seitengenerierung unter Verwendung von GRASS-Daten durchführen. So kann GRASS ohne große GIS-Kenntnisse „fernbedient“ werden. Für die Kartenausgabe im Browser werden Vektordaten in Rasterdaten umgewandelt.
Die C-Programmierung erfordert dagegen umfassendere Kenntnisse, sie soll im Rahmen dieses Handbuchs nur am Rande angesprochen werden.

13.1 Script-Programmierung

Eine angenehme Erweiterung (auch für GRASS-Anfänger) stellt die Möglichkeit dar, einzelne Rechenschritte automatisiert ablaufen zu lassen. Diese Scripte sind als UNIX-Shell-Scripte im ASCII-Textformat zu schreiben. Darin lassen sich GRASS-Module mit den entsprechenden Parametern aufrufen und so beispielsweise geostatistische Berechnungen automatisiert durchführen. Der UNIX-Befehl `$ history` ermöglicht, vorher eingegebene Befehle zu betrachten bzw. zu speichern („copy“-„paste“ mit linker und mittlerer Maustaste in „xedit“ oder einen anderen Editor).

Ein Beispiel soll das Potenzial der Script-Programmierung verdeutlichen: Die Berechnung allgemeiner geostatistischer Parameter für Rasterbilder kann auf folgende Weise durchgeführt werden (nach ALBRECHT 1992, leicht verändert). Speichern Sie das Beispiel z.B. als „`statistik.sh`“ ab; die UNIX-Ausführungsrechte sind mit `$ chmod u+x statistik.sh` zu setzen:

```
#!/bin/sh
#
# Univariate Statistik für GRASS-Rasterdaten
#
if test "$GISBASE" = ""; then
```

[1] `http://www.vtt.co.jp/staff/sorokin/jni/` von Alexandre Sorokine

```
 echo "You must be in GRASS GIS to run this program." >&2
 exit 1
fi

eval `g.gisenv`
: ${GISBASE?} ${GISDBASE?} ${LOCATION_NAME?} ${MAPSET?}
LOCATION=$GISDBASE/$LOCATION_NAME/$MAPSET

input="$1"
r.stats -1 input="$input" | awk 'BEGIN {sum = 0.0 ; sum2 = 0.0}
NR == 1{min = $1 ; max = $1}
       {sum += $1 ; sum2 += $1 * $1 ; N++}
       {
        if ($1 > max) {max = $1}
        if ($1 < min) {min = $1}
       }
END{
print "Anzahl der Rasterzellen (Stichproben)  N =",N
print "Kleinster Wert          MIN =",min
print "Hoechster Wert          MAX =",max
print "Variationsweite           v =",(max - ((min * -1) * -1))
print "Mittelwert              MEAN =",sum / N
print "Varianz                   S2 =",(sum2 - sum * sum / N) / N
print "Standardabweichung         S =",sqrt((sum2 - sum * sum/N) / N)
print "Variationskoeffizient      V =",(((sqrt((sum2 - sum * sum/N)
       * 100)/ N)) / (sqrt(sum*sum)/N))
}'
```

Als Parameter wird dem Script, das nach dem Start von GRASS aufgerufen werden kann, der Dateiname der zu analysierenden Rasterdatei übergeben. Das Script ist in modifizierter Form als `$ r.univar` in GRASS 5 enthalten. Hier wird die Ausgabe des GRASS-Moduls über „UNIX-Piping“ an das UNIX-Programm „awk“ übergeben. Innerhalb der „awk“-Umgebung werden die entsprechenden Berechnungen durchgeführt (vgl. auch Abschnitt 6.17).

Das nächste Script dient der Berechnung des Schwerpunkts einer Fläche. Sie können es beispielsweise zur Bestimmung des Gebietsschwerpunkts in der Hydrologie einsetzen. Dazu berechnen Sie mit `$ r.watershed` die Einzugsgebiete und maskieren alle „basins“ bis auf das Gebiet Ihres Interesses (r.centroid arbeitet nur mit einer Fläche). Sie können das folgende Script als Textdatei speichern oder aus dem Internet[2] laden:

```
#!/bin/sh
# calculates centroid of raster area (center of gravity)
# Useful for watersheds etc.
```

[2] Sie finden es unter: `http://www.geog.uni-hannover.de/users/neteler/handbuch/`

```
#
# Markus Neteler
# V 1.1 - 12. Feb. 2000

if test "$GISBASE" = ""; then
 echo "You must be in GRASS GIS to run this program." >&2
 exit 1
fi

eval `g.gisenv`
: ${GISBASE?} ${GISDBASE?} ${LOCATION_NAME?} ${MAPSET?}
LOCATION=$GISDBASE/$LOCATION_NAME/$MAPSET

TMP=`g.tempfile pid=$$`
if [ $# = 0 ] || [ $1 = "" ]
    then
        g.ask type=any element=cell unixfile=$TMP
        . $TMP
    elif [ $1 = "help" ]
        then
            echo "Calculates center of gravity (centroid) of raster area"
            echo "USAGE: r.centroid [raster map]"
            exit
    elif [ $1 = "-help" ]
        then
            echo "Calculates center of gravity (centroid) of raster area"
            echo "USAGE: r.centroid [raster map]"
            exit
    else
        name=$1
fi
if [ ! "$name" ]
then
   exit
fi

# example: calculate watershed (minimum size: 1000 cell units)
#  r.watershed elevation=dgm25 basin=basin threshold=1000
# now select your watershed by masking:
#  r.mapcalc "MASK=if(basin == 6)"
# check it with: d.rast basin
# here we go for centroid calculation:
# centroid is
#              N
```

```
#  x_c = 1/A * SUM (x_i * a_i)
#              i=1
#
#               M
#  y_c = 1/A * SUM (y_i * a_i)
#              i=1
# with
#  N: total number of cells in x direction
#  M: total number of cells in y direction
#  x_i: distance of cell center from left boundary
#  y_i: distance of cell center from upper boundary
#  a_i: area of ith cell

# calculate area
AREA=`r.surf.area $name |grep plan |cut -d':' -f2 |awk '{printf "%.2f", $1}'`
export AREA
MORETHANONE=`echo $AREA| cut -d' ' -f2| wc -w`
if [ $MORETHANONE -gt 1 ]
then
 echo "ERROR: more than one area in this map!"
 echo "Use r.mask to masking area of interest"
 exit
fi

# determine resolution
EWRES=`awk ' /e-w/ { print $3}' $LOCATION/WIND`
export EWRES
NSRES=`awk ' /n-s/ { print $3}' $LOCATION/WIND`
export NSRES

if [ -f $LOCATION/../PERMANENT/PROJ_UNITS ] ; then
  UNITS=`cat $LOCATION/../PERMANENT/PROJ_UNITS |grep units |cut -d' ' -f2`
else
  UNITS="cellunits"
fi

echo "Basin area: $AREA $UNITS^2"
echo "Resolution $EWRES, $NSRES"

echo "Calculating x_min and x_min of area..."
#calculate x_min
XMIN=`r.stats -1gnq $name |cut -d ' ' -f1 | awk 'BEGIN{min = 0.0}
NR == 1{min = $1}
       {if ($1 < min) {min = $1}}
```

```
END{print min}'`

#calculate y_min
YMIN=`r.stats -1gnq $name |cut -d ' ' -f2 | awk 'BEGIN{min = 0.0}
NR == 1{min = $1}
       {if ($1 < min) {min = $1}}
END{print min}'`

echo "Calculating centroid..."

# calculate x_c:
r.stats -1gnq $name |cut -d ' ' -f1 | gawk 'BEGIN{
    sum = 0.0 ; calc = 0.0 ; xmin2 = 0.0
    ewres = '$EWRES' ; nsres = '$NSRES'
    xmin  = '$XMIN'  ; area  = '$AREA'}
 NR == 1{xmin2 = xmin * 1.0 ; ewres2 = ewres * 1.0 ; nsres2 = nsres * 1.0}
        {calc = ($1 - xmin2) * ewres2 * nsres2}
        {sum = sum + calc}
 END{printf "Center of gravity x_c: %.2f\n", sum/area + xmin2}'

# calculate y_c:
r.stats -1gnq $name |cut -d ' ' -f2 | gawk 'BEGIN{
    sum = 0.0 ; calc = 0.0 ; ymin2 = 0.0
    ewres = '$EWRES' ; nsres = '$NSRES'
    ymin  = '$YMIN'  ; area  = '$AREA'}
 NR == 1{ymin2 = ymin * 1.0 }
        {calc = ($1 - ymin2) * ewres * nsres}
        {sum = sum + calc}
 END{printf "Center of gravity y_c: %.2f\n", sum/area+ymin2}'

echo ""
rm -f $TMP
```

Das Script zeigt Ihnen, wie UNIX-Kommandos und GRASS-Module verknüpft werden können. Über das „UNIX-Piping“ tauschen Module Informationen aus, an manchen Stellen wird auch eine temporäre Textdatei erzeugt und in einem weiteren Schritt wieder analysiert. Einzelwerte und Zeichenketten können auch in Variablen abgelegt werden. Wenn Sie selbst programmieren, ist es ratsam, zunächst möglichst viele Hilfsausgaben (mit „echo $VARIABLE“) einzubauen, um den Programmablauf zu kontrollieren.

Nützlich sind die Module `$ g.ask` (um die Nutzer komfortabel nach den zu analysierenden GRASS-Dateien zu fragen) bzw. `$ g.findfile` (um direkt eine Datei in der GRASS-Datenbank zu finden und zu verarbeiten).

Weitere Scripte sind beispielsweise bei ALBRECHT 1992 und SHAPIRO 1992 beschrieben. Außerdem können Sie sich die in GRASS enthaltenen Scripte anschauen (wenn Sie GRASS aufgerufen haben, wechseln Sie in `$ cd $GISBASE/scripts`).

13.2 Automatisierte Benutzung von GRASS

GRASS kann komplett scriptgesteuert (und damit automatisiert) benutzt werden, indem zuerst die passenden Umgebungsvariablen gesetzt werden (Beispiel für die „bash-shell“):

```
echo "MAPSET: innenstadt"                  >  ~/.grassrc5
echo "LOCATION_NAME: hannover"             >> ~/.grassrc5
echo "DIGITIZER: none"                     >> ~/.grassrc5
echo "GISDBASE: /home/neteler/grassdata5" >> ~/.grassrc5

DIGITIZER=none
GISBASE=/usr/local/grass5
GISDBASE=/home/neteler/grassdata5
GISRC=/home/neteler/.grassrc5
LOCATION=/home/neteler/grassdata/hannover/innenstadt
PATH=$PATH:/usr/local/grass5/bin:/usr/local/grass5/scripts:\
          /usr/local/grass5/garden/bin

export DIGITIZER GISBASE GISDBASE GISRC LOCATION PATH
```

Nun können innerhalb dieser Shell alle GRASS-Module genutzt werden, die sich über Parameter ansteuern lassen. Generell erhalten Sie die Parameterliste eines Moduls, indem Sie „help“ als Parameter angeben oder die Anleitung lesen.
Nach Setzen dieser Variablen kann GRASS auch in CGI- und PERL-Scripten eingebunden werden. Sie können sich ein umfangreiches Beispiel, basierend auf dem freien UMN MapServerm, im Internet anschauen: GRASSLinks[3] ist ein komplette Online-GIS mit MapServer/GRASS.

13.3 Hinweise zur C-Programmierung für GRASS

Alle relevanten Aspekte der C-Programmierung für GRASS sind im GRASS 5.0-Programmierhandbuch (GRASS DEVELOPMENT TEAM 2001) beschrieben). Hier soll nur überblicksartig gegliedert werden, wie GRASS-Module aufgebaut sind. Generell empfiehlt es sich, vorhandene Module (über 400) exemplarisch anzuschauen und davon zu lernen. Diese Möglichkeit bietet nur ein „open-source“-GIS wie GRASS. Der prinzipielle Aufbau der Module ist immer gleich, jedes Modul ist in einem Verzeichnis im GRASS-Quellcode abgelegt.

[3] `http://grass.itc.it/start.html`

Die aktuelle Struktur sieht so aus:
GRASS-GIS-Bibliothek:

```
- man/                  # Modulbeschreibungen
- src/CMD/              # interne Scripts zur Kompilierung
- src/include/          # Header-Dateien
- src/libes/            # GIS-Bibliotheksroutinen
- src/display/devices   # Bildschirmtreiber
- src/fonts/            # Zeichensätze
- src/front.end/        # interne Routinen für interaktiven Modus der Module
```

Module (Standardbaum):

```
- src/display/          # Module zur Bildschirmausgabe auf GRASS-Monitor
- src/general/          # Dateiverwaltungsmodule
- src/imagery/          # Bildverarbeitungsmodule
- src/mapdev/           # Vektormodule
- src/misc/             # verschiedene Module
- src/paint/            # Paint-Treiber
- src/ps.map/           # Postscript-Treiber
- src/raster/           # Rastermodule
- src/scripts/          # Scripte
- src/sites/            # Punktdatenmodule
- src/tcltkgrass/       # graphische Tcl/Tk-Oberfläche
```

Zulieferungen externer Institutionen:

```
- src.contrib/
```

Module mit gekoppelten Simulationsmodellen und Anbindungen:

```
- src.garden/
```

Externe Programme, die für GRASS interessant sind:

```
- src.related/
```

Die vorhandenen GRASS-Module bauen auf der „GRASS programming library“ auf, die eine Vielzahl von Funktionen bietet und sich folgendermaßen gliedert (in eckigen Klammern der typische Wortanfang für die jeweiligen Bibliotheksroutinen):

- GIS-Bibliothek: Datenbank-Routinen (GRASS-Dateien verwalten), Speicherverwaltung, Parser (String-Analyse), Projektionen usw. [G_]
- Vektor-Bibliothek: Verwaltung von Flächen-, Linien- und Punktvektoren [Vect_, V2_, dig_]

- Rasterbibliothek: Verwaltung von Rasterdaten [R_]
- Punktdatenbibliothek: Verwaltung von Punktdaten [G_]
- Anzeige-Bibliothek: Graphische Datenausgabe am Bildschirm [D_]
- Treiber-Bibliothek: Druckertreiber
- Bilddaten-Bibliothek: Zur Verwaltung bildverarbeitungsspezifischer Dateien [I_]
- Segment-Bibliothek: Zur segmentierten Datenbearbeitung [segment_]
- Vask-Bibliothek: Steuerung der Cursortasten etc. [V_]
- Rowio-Bibliothek: Zur gleichzeitigen Zeilenanalyse in Rasterdaten [rowio_]

Die angebotenen Routinen sind zum Teil sehr mächtig, sie bieten beispielsweise die Möglichkeit, direkt geodätische Distanzen aus angegebenen Koodinatenpunkten zu berechnen oder Vektorflächen abzufragen (z.B. Punkt-in-Polygon).

Module bestehen aus den C-Programmdateien (∗.c), den Header-Dateien (∗.h) und einem „Gmakefile“. GRASS hat seine eigene „make“-Routine: `$ gmake4.2` bzw. `$ gmake5`. Die Datei „Gmakefile“ enthält Anweisungen über die zu übersetzenden Dateien und die zu verwendenen Bibliotheken (GRASS-Bibliotheken und UNIX-Bibliotheken). Sie hat einen bestimmten Aufbau, der einzuhalten ist. Ein einfaches Beispiel (wichtig: Einrückungen mit Tabulator und nicht mit Leerzeichen!) soll den typischen Aufbau demonstrieren:

```
PGM=i.sat.reflectance
HOME=$(BIN_CMD)

LIBES= $(IMAGERYLIB) $(GISLIB) $(VASKLIB) $(VASK)
DEPLIBS=$(DEPIMAGERYLIB) $(DEPGISLIB)
OBJ = main.o\
        open.o\
        atmos.o\
        sun_pos.o\
        correction.o\
        histogram.o\
        history.o

$(HOME)/$(PGM): $(DEPLIBS)
        $(CC) $(LDFLAGS) -o $@ $(LIST) $(LIBES) $(MATHLIB) $(XDRLIB)

$(LIST): global.h
```

```
$(IMAGERYLIB): #
$(GISLIB): #
$(VASKLIB): #
```

Die Zeile „$(HOME)/$(PGM)...“ enthält die Compileranweisungen, darüber in der Datei werden Variablen gesetzt. Hier nicht gesetzte Variablen sind in `grass5/src/CMD/head/head` automatisch definiert. Diese „head“-Datei wird ab GRASS 5 mit dem „configure“ Script vor dem ersten Kompilieren plattformspezifisch erzeugt. Die Variable $(HOME) gibt an, wohin die Binärdatei (also das Modul) kopiert wird: in den Standardpfad für GRASS command-line-Module `grass5/etc/bin/cmd/`.
Das eigentliche Programm wird üblicherweise thematisch gegliedert in mehreren Dateien gespeichert, die entsprechend im „Gmakefile“ bei den Objekten aufgelistet werden müssen. Zur eigentlichen C-Programmierung kann im Rahmen dieses Handbuchs nichts gesagt werden. Die Befehle der GRASS GIS-Library lassen sich direkt im Quellcode verwenden. Eine Parameterabfrage bei Modulen wird über die GRASS-typische Parameterprogrammierung ermöglicht. Ein kurzes Beispiel für ein Rastermodul (Datei main.c):

```
/* Conversion of LANDSAT TM digital numbers to radiances
 * (c) 2000 Markus Neteler, Hannover, Germany
 */

#include "gis.h"
#include <stdio.h>
#include <strings.h>
#include <math.h>

int main(int argc, char *argv[])
{
    int nrows, ncols;
    int row, col;
    int i;
    struct
    {
        struct Option *group, *date, *result;
    } parm;
    struct GModule *module;

    G_gisinit (argv[0]);
```

```
    module->description =
      "conversion of LANDSAT TM digital numbers to radiances." ;

    parm.group = G_define_option();
    parm.group->key = "group";
    parm.group->type = TYPE_STRING;
    parm.group->required = YES;
    parm.group->description = "Imagery group of images to be converted";

    flag.quiet = G_define_flag();
    flag.quiet->key = 'q';
    flag.quiet->description = "Run quietly";

    if (G_parser(argc,argv))
        exit(1);
    group         = parm.group->answer;
    open_files();  /* In weiterer Datei definiert */

    nrows = G_window_rows();
    ncols = G_window_cols();

    /* go rowwise and colwise through image */
    for (row = 0; row < nrows; row++)        /* rows loop */
    {
          for (col = 0; col < ncols; col++)             /* cols loop */
           {
              result_cell[col] = calc_new_pixel;  /* in weiterer Datei */
           }  /* end cols loop */

           G_put_d_raster_row (fd, (DCELL *) result_cell);
    } /* end rows loop */

    G_close_cell (fd);
    exit(0);
}
```

Die Berechnung erfolgt zeilen- und spaltenweise (for-Schleife). Es ist sinnvoll, das Gesamtmodul in einzelne, thematisch sortierte Dateien zu zerlegen, um die Programmpflege zu vereinfachen.

14 Zitierte Literatur

ALBERTZ, J. (1991): Grundlagen der Interpretation von Luft- und Satellitenbildern. Eine Einführung in die Fernerkundung. Darmstadt

ALBRECHT, J. (1992): GTZ-handbook GRASS. Vechta
Im Internet: `http://grass.itc.it/gdp/`

BÄHR, H.P. & T. VÖGTLE (Hrsg.) (1991): Digitale Bildverarbeitung. 2. Aufl., Karlsruhe

BAHRENBERG, G., E. GIESE & J. NIPPER (1990): Statistische Methoden in der Geographie. Band 1, Stuttgart

BAHRENBERG, G., E. GIESE & J. NIPPER (1992): Statistische Methoden in der Geographie. Band 2, Stuttgart

BARTELME, N. (1995): Geoinformatik. Modelle, Strukturen, Funktionen. Heidelberg

BIENLEIN, J., L. DRESCHLER-FISCHER & H. SPITZER (Hrsg.) (1996): Fernerkundung und Bildverarbeitung. CENSIS-REPORT 1996. Universität Hamburg

BIERHALS, E. (1988): CIR-Luftbilder für die flächendeckende Biotopkartierung. Informationsdienst Naturschutz Niedersachsen 5/88. Hannover, S. 77-104

BILL, R. (1996): Grundlagen der Geo-Informations-Systeme. Analysen, Anwendungen und neue Entwicklungen. Band 2, Heidelberg

BRANDON, R.J., T. KLUDT & M. NETELER (1999): Archaeology and GIS – The Linux Way. Using GRASS and Linux to analyze archaeological data. Linux Journal, July 1999

BURROUGH, P.A. & R.A. McDONNELL (1998): Principles of Geographical Information Systems. Spatial Information Systems and Geostatistics. Oxford

BUZIEK, G. (Hrsg.) (1995): GIS in Forschung und Praxis. Hannover

CERL (1993): GRASS 4.1 Installation Guide.
Im Internet: `http://grass.itc.it/gdp/`

DESMET, P.J.J. & GOVERS, G. (1996): Comparison of Routing Algorithms for Digital Elevation Models and their Implications for Predicting Ephemeral Gullies. International Journal of GIS, 10: 311-331

FOX, J. (1989): RIM Installers manual. University of Washington

FOX, J. (1989): RIM Users manual. University of Washington

HAKE, G. & D. GRÜNREICH (1994): Kartographie. 7. Aufl., Berlin

HARMON, V. & M. SHAPIRO (1992): GRASS tutorial: Image processing. CERL, Champaign. Illinois.
Im Internet: `http://grass.itc.it/gdp/imagery/imagery.ps.gz`

HILDEBRANDT, G. (1996): Fernerkundung und Luftbildmessung: für Forstwirtschaft, Vegetationskartierung und Landschaftsökologie. Heidelberg.

HILDEBRANDT, R. (1992): Digitale und visuelle Interpretation von LANDSAT-TM- und SPOT-Daten sowie Luftbildern in Hinblick auf die Geologie in der Region des Jebel Marra (Provinz Darfur, Sudan). Berlin

JACOBS, H. (1998): Analyse fernerkundlicher Scannerdaten. In: Bähr, H.-P., Th. Vögtle (Hrsg.): Digitale Bildverarbeitung. Anwendung in Photogrammetrie, Kartographie und Fernerkundung. Heidelberg

JENSEN, J.R. (1996): Introductory Digital Image Processing. A remote sensing perspective. 2nd ed., New Jersey

JENSON, S.K. & J.O. Domingue (1988): Extracting topographic structure from digital elevation model data for geographic information system analysis. Photogram. Engr. and Remote Sens. 54: 1593-1600

JOHNSTON, C.A. (1998): Geographic Information Systems in Ecology. Methods in Ecology. Oxford

LARSON, M., M. SHAPIRO & S. TWEDDALE (1991): Performing map calculations on GRASS data: r.mapcalc programming tutorial. CERL, Champaign, Illinois

LILLESAND, TH.M. & R.W. KIEFER (1994): Remote sensing and image interpretation. New York

LÖFFLER, E. (1994): Geographie und Fernerkundung. 2. Auflage, Stuttgart

MANDELBROT, B.B. (1987): Die fraktale Geometrie der Natur. Basel

McCAULEY, J.D. & B.A. ENGEL (1994): Comparison of Scene Segmentations: SMAP, ECHO and Maximum Likelihood. IEEE Trans. on Image Proc. Vol. 6(2), pp. 1-4

MITAS, L. & H. MITASOVA (1999): Spatial Interpolation. In: P. LONGLEY, M.F. GOODCHILD, D.J. MAGUIRE & D.W. RHIND (Eds.): Geographical Information Systems: Principles, Techniques, Management and Applications, pp. 481-492

MITASOVA, H. & L. MITAS (1993a): Interpolation by regularized spline with tension: I. Theory and implementation, Mathematical Geology No. 25, pp. 641-656

MITASOVA, H. & J. HOFIERKA (1993b): Interpolation by regularized spline with tension: II. Application to terrain modeling and surface geometry analysis, Mathematical Geology No. 25, p. 657-667

Im Internet: http://skagit.meas.ncsu.edu/~helena/

MITASOVA, H., J. HOFIERKA, M. ZLOCHA & R.L. IVERSON (1996): Modelling Topographic Potential for Erosion and Deposition Using GIS. International Journal of Geographical Information Systems, 10, 629-641

NEIDIG, C.A., D. GERDES & CH. KOS (1991): GRASS 4.0 map digitizing manual: v.digit. CERL, Champaign, Illinois

NETELER, M. (1997): Introduction to GRASS GIS Software. Madras (Indien)/Hannover
Im Internet: http://grass.itc.it/gdp/neteler/

NETELER, M. & H. MITASOVA (2002): Open Source GIS: A GRASS GIS Approach. Boston, Dordrecht. Online supplement: http://mpa.itc.it/grasstutor/

RASE, W.-D. (1998): Visualisierung von Planungsinformationen: Modellierung und Darstellung immaterieller Oberflächen. Bundesamt für Bauwesen und Raumordnung, Forschungen H. 89, Bonn

REDSLOB, M. (1998): Radarfernerkundung in niedersächsischen Hochmooren. Diss. Inst. f. Landschaftspfl. u. Natursch., Univ. Hannover

RESEARCH SYSTEMS, INC. (1997): ENVI 3.0 Tutorial. The environment for visualizing images, Version 3.0. Better solutions consulting, Boulder, Colorado

RIPLEY, B.D. (1996): Pattern recognition and neural networks. Cambridge

SAURER, H. & F.-J. BEHR (1997): Geographische Informationsysteme. Eine Einführung. Darmstadt

SCHOWENGERDT, R. (1997): Remote sensing: Models and methods for image processing. 2nd ed., Tucson, Arizona

SEEL, K. (1995): Mitteilung in news://info.grass.user

SHAPIRO, M. & J. WESTERVELD (1991): GRASS 4.1 Programmer's manual. CERL, Champaign, Illinois

SHAPIRO, M. & J. WESTERVELD (1991): GRASS 4.1 User's manual. CERL, Champaign, Illinois

SHAPIRO, M. & J. WESTERWELD (1992): r.mapcalc. An algebra for GIS and image processing. CERL, Champaign, Illinois

RAYMOND, E. (1999): The Cathedral & the Bazaar. Musings on Linux and Open Source by an accidental revolutionary. Cambridge

ÜBERLA, K. (1968): Faktorenanalyse. Eine systematische Einführung für Psychologen, Mediziner, Wirtschafts- und Sozialwissenschaftler. Berlin

WADSWORTH, R. & J. TREWEEK (1999): Geographical Information Systems for Ecology. An Introduction. Essex

WISCHMEIER, W.H. & D.D. SMITH (1978): Predicting rainfall erosion losses - A guide to conservation planning. Agriculture Handbook No. 537, Science and Education Administration, U.S. Dep. of Agriculture, Washington, D.C.

WOOD, J. (1996): The Geomorphological characterisation of Digital Elevation Models. Diss., Department of Geography, University of Leicester, U.K.
Im Internet: `http://www.geog.le.ac.uk/jwo/research/dem_char/thesis/index.html`

A Anhang

A.1 Antworten auf häufig gestellte Fragen

Die folgenden Fragen und Antworten sollen helfen, typische Fehler zu erkennen und zu beheben. Meistens „hängt“ es nur an Kleinigkeiten, GRASS bereitet glücklicherweise äußerst selten schwerwiegende Probleme.

- *Größe des Display-Fensters einstellen:* In die .ishell¿rc-Datei (z.B. .bashrc oder .cshrc) muss Folgendes eingetragen werden:

  export GRASS_WIDTH=500

  export GRASS_HEIGTH=600

  Die Zahlen können den individuellen Bedürfnissen angepasst werden (wichtig sind die Unterstriche in den Variablen).

  Diese Werte können auch in TclTkGRASS angegeben werden („CONFIG“ ⇝„Options“ ⇝„Display dimensions“). Ein eventuell bereits geöffneter GRASS-Monitor ist nun zu schließen und erneut zu öffnen, um die veränderte Größe zu bekommen. Alternativ können Sie auch das Script `$ d.monsize` benutzen.

- *Größe des CELL-Treibers einstellen:* In die .ishell¿rc-Datei (z.B. .bashrc oder .cshrc oder global in /etc/profiles) muss Folgendes eingetragen werden:

  export GRASS_WIDTH=500

  export GRASS_HEIGTH=600

  Die Zahlen sind den individuellen Bedürfnissen anzupassen (wichtig sind die Unterstriche in den Variablen).

  Diese Werte können auch in TclTKGRASS angegeben werden („CONFIG“ ⇝„Options“ ⇝„Display dimensions“).

- *WARNING: can’t read range file for ...:*
  Problem: Diese Meldung erscheint in GRASS 5 bei der Benutzung von Rastermodulen, die r.stats verwenden.

  Die Statistik der Datei liegt nicht korrekt berechnet vor, da Sie vermutlich mit GRASS 5 eine GRASS-Datenbank bearbeiten, die mit GRASS 4 erstellt wurde. Rufen Sie r.support zur

Lösung des Problems auf: Geben Sie die Datei entsprechend an, dann bei „Edit headers:“ no, bei „Update stats...“ yes, um die Kartenstatistik neu zu berechnen, bei den weiteren Fragen „return“. Der Fehler sollte damit für diese Karte behoben sein.

- *d.rast:*
  (a) Problem: Die Rasterdatenausgabe mit d.rast erzeugt nur weiße oder schwarze Flächen, anstatt das Bild auszugeben.

  Es bestehen verschiedene Möglichkeiten zur Problemlösung:

  1. Nach `$ r.in.sunrast`, `$ r.in.gif`, `$ r.in.tiff`, ist der Aufruf `$ r.support` nötig, um für die importierte Karte die korrekten Randkoordinaten anzugeben (also beide Menüpunkte in `$ r.support` aufrufen).
  2. GRASS 4.x: Bei Aufruf des GRASS-Monitors erscheint die Meldung: *„Can't set color 0“* bzw. der Monitor wird weiß (nur bei 8bit-Bildschirmen). Dann besteht ein Konflikt zwischen einem anderen Anwendungsprogramm und GRASS – beide Programme versuchen, dieselben Farben zu verwenden. Also müssen der bereits gestartete GRASS-Monitor (nicht GRASS!) und das problematische Anwendungsprogramm geschlossen werden (i.a. handelt es sich um „netscape“ oder „xv“). Nun werden zuerst der GRASS-Monitor und anschließend das problematische Anwendungsprogramm wieder gestartet. Vermutlich wird dieses nun eine Fehlermeldung ausgeben, die aber nicht vermeidbar ist. Zumindest arbeitet nun die Grafikausgabe unter GRASS. Deshalb sollte der GRASS-Monitor vor anderen graphischen Anwendungsprogrammen aufgerufen werden. Wird der 24bit-Treiber von GRASS benutzt, sollte dieses Problem nicht mehr existieren.
  3. Bei der Darstellung von Karten/Images erscheint die Meldung: *„Map in wrong projection“*. Es wurde GRASS verlassen und mit einer neuen *location* wieder gestartet, dabei aber der GRASS-Monitor nicht geschlossen. Dadurch hat der Monitor noch die Projektionsart der vorherigen *location*. Lösung: GRASS-Monitor schließen und neu starten (mit d.mon).

  (b) GRASS 4.x-Problem: Das Zoomen funktioniert nicht richtig.

  Vermutlich wurde vergessen, das Modul `$ d.erase` vor der erneuten Bildausgabe aufzurufen (v.a. nach `$ g.region`).

  (c) Problem: Bei dem Versuch, Karten mit d.rast darzustellen, erscheint die Meldung: „WARNING: [rasterfile] in mapset [mapset] in different projection than current region.“

  Es wurde eine neue *location* mit anderer Projektion aufgerufen (durch das Verlassen von GRASS und einen Neustart), aber dabei der Monitor nicht geschlossen. Dieser hat also noch die vorige Projektionsart. Einfach den Monitor mit d.mon schließen und neu starten.

- *d.vect:*
  Problem: Bei dem Versuch, Karten mit d.vect darzustellen, erscheint die Meldung: „WARNING: [vectorfile] in mapset [mapset] in different projection than current region.“

  Hier wurde eine neue *location* mit anderer Projektion aufgerufen (durch das Verlassen von GRASS und einen Neustart), aber dabei der Monitor nicht geschlossen. Dieser weist also noch die vorige Projektionsart auf. Einfach den Monitor mit d.mon schließen und neu starten.

- *i.vpoints:*
  Problem: Nach dem Verlassen des Moduls kommt es manchmal zu Störungen in nachfolgenden Ausgaben von Raster- oder Vektordaten.
  Zur Behebung dieses Problems muss der Monitor mit `$ d.frame -e` „aufgeräumt“ werden.

- *s.in.ascii:*
  Problem: Sie möchten eine XYZ-organisierte Datei (z.B. Geländehöhen) importieren und erhalten ungefähr die Meldung: „- line 44467 ** invalid format **“.
  In dieser Zeile ist das Format anders als in der übrigen Datei. Prüfen Sie mit einem Texteditor die entsprechende Zeile nach. Handelt es sich um die letzte Zeile, dann achten Sie darauf, dass keine Leerzeichen enthalten sind.

- *r.mapcalc:*
  Problem: Das Rasterbild hat nur einen Wert.
  Wahrscheinlich haben Sie als Dateinamen nur einen Zahlenwert benutzt. Entweder muss dieser Name dann in Anführungsstrichen stehen oder mindestens ein Buchstabe hinzugefügt werden. Ansonsten wird der Dateiname als Wertzuweisung interpretiert.

- *r.mask:*
  Problem: Das Modul r.mask lässt sich nur interaktiv bedienen. Wie setzt man in Scripten eine Maske?
  Lösung: Wenn eine Maske nicht-interaktiv gesetzt werden soll (z.B. in Scripten), kann sie mit g.copy gesetzt werden: `g.copy rast=meinemaske,MASK`

- *v.digit:*
  (a) Problem: Manchmal werden keine Vektoren beim Menüpunkt „Bulk remaining labels“ gelabelt (alle unspezifizierten Vektoren mit Labeln versehen).
  Hier hilft folgender Trick:
  Man digitalisiert mit der Maus einen beliebigen Vektor an eine freie Stelle, lässt dann „label“-„Bulk remaining labels“ erfolgreich laufen und löscht die eben neu digitalisierte Linie wieder.

  (b) Problem: Keine Ausgabe bereits digitalisierter Vektoren nach Aufruf von v.digit.
  Der Fehler besteht häufig darin, dass im Startformular von `$ v.digit`
  „map scale“ mit 1:0 anstelle des entsprechenden Kartenmaßstabs angegeben ist.

- *v.in.arc:*
  Problem: Das Modul weigert sich, ARC/INFO-Daten zu importieren.
  Üblicherweise wurde vergessen, die zu importierenden Dateien in der GRASS-Database im einzurichtenden Unterverzeichnis `../arc` abzulegen. Vgl. dazu Abschnitt A.4.

- *v.in.dxf3d:*
  Problem: Das Modul arbeitet bei Linienvektoren korrekt, bei Polygonen allerdings nicht.
  Die Lösung besteht im Anwenden von `$ v.line2area` nach dem DXF-Import. Anschließend folgt `$ v.alabel`, das mit laufender Nummer die Vektoren labelt. Die Attribute können anschließend mit `$ v.support` hinzugefügt werden.

- *v.to.rast:*
  Problem: Die Vektor-Umwandlung in das Rasterformat liefert keine Ergebnisse.
  Lösung: Vektoren können nur in Raster umgewandelt werden, wenn sie ein Label haben. Diese können mit `$ v.alabel` oder `$ v.digit` vergeben werden.

A.2 Kurzübersicht der wichtigen GRASS-Befehle

Die folgende Übersicht über rund 250 Befehle ist eine Zusammenstellung aus verschiedenen Publikationen, Handbüchern und den Online-Anleitungen. Ausführliche Hilfe gibt es bei den meisten Modulen mit dem Befehl:
`$ g.manual GRASS-Befehl`
Alternativ bzw. zusätzlich kann (fast) jedem Befehl auch die Option „help“ übergeben werden.

Die Online-Modulbeschreibungen finden Sie im Internet unter:
`http://grass.itc.it/gdp/`
Neben den hier besprochenen Modulen existieren im Internet etliche weitere Pakete, die Oberflächenanalyse, Erosionsmodellierung und insbesondere auch hydrologische Simulation in GRASS ermöglichen. Vgl. dazu auch „Bezugsadressen“ im Abschnitt 2.1.

A.2.1 Punktdatenbefehle

s.delaunay: Berechnung einer Delaunay-Triangulation aus Sites-Daten
s.in.ascii: importiert Sites-Dateien (Punktinformationen). Anordnung in jeder Zeile:
`Eastcoordinate Northcoordinate Category` (getrennt durch Leerzeichen), also
`Rechtswert Hochwert Z-Wert`, geeignet für den Import digitaler Höhenmodelle
s.info: gibt Angaben über Attribute einer Punktdatenkarte
s.kcv: teilt eine Punktdatenkarte in zufällige Teilgebiete für eine „k-fold cross validation“
s.label: erzeugt eine „label“-Datei für ps.map
s.medp: Modul zur Glättung von Sites-Daten durch Verschieben der Punkte auf das nächste Gitterkreuz (Vektorgittererstellung mit v.mkgrid)

s.menu: Modul zur Verwaltung und Bearbeitung von Sites-Daten

s.normal: führt zahlreiche geostatistische Normalitätstests durch: Schiefe (skewness) und Steilheit (kurtosis) einer Stickprobenverteilung (sample), Geary's a-statistic und eine angenäherte Normaltransformation, extreme Normalabweichung, D'Agostino's D-statistic, modified Kuiper V-statistic, modified Watson U^2-statistic, Durbin's Exact Test (modified Kolmogorov), modified Anderson-Darling statistic, modified Cramer-Von Mises W^2-statistic, Kolmogorov-Smirnov D-statistic (modifiziert für Normaliätstest), Chi-Square test statistic (gleiche Wahrscheinlichkeitsklassen) und die Zahl der Freiheitsgrade, Shapiro-Wilk W Test, Weisberg-Binghams W" (ähnlich zu Shapiro-Francia's W'), Royston's Erweiterung von W für große Stichproben, Kotz Separate-Families Test für Lognormalität gegen Normalität

s.out.ascii: Export von Sites-Daten in das ASCII-Format
(Benutzung: `$ s.out.ascii sites=karte > exportdatei`)

s.perturb: führt Perturbationen von Punktdatenkoordinaten durch

s.probplot: erzeugt (für gnuplot) einen „probability plot" von Punktdatenattributen

s.qcount: berechnet Anzahl und Durchmesser von Quadraten, die ohne Überlappung die Punktdaten abdecken

s.rand: erzeugt zufällige Punktkarte

s.sample: erzeugt aus Punktkarte eine Rasterkarte per Interpolation (Methode wählbar: bilinear, cubic convolution oder nearest neighbor)

s.surf.idw: Oberflächeninterpolation, um „Löcher" (Wert 0) in einem Höhenmodell über einen gewichteten Durchschnitt zu füllen (Ergebnis ist eine Rasterkarte); Auflösungserhöhung möglich, dazu Auflösung vorher in g.region hochsetzen

s.surf.rst: topographische Oberflächenanalyse eines Höhenmodells (Ergebnisse als Fließkommakarten). Berechnung von Interpolation, Hangneigung, Exposition, Profilkrümmung horizontal und vertikal (in Richtung der stärksten Hangneigung) und Durchschnittskrümmung möglich (Interpolation über „regularized spline with tension" – Algorithmus)

s.to.vect: Konvertierung von Sites-Daten in das Vektorformat

s.to.rast: Konvertierung von Sites-Daten in das Rasterformat

s.univar: berechnet univariate Statistik aus Punktdaten: Punktanzahl, Mean, Standardabweichung, Variationskoeffizient, Mininum, Erstes Quartil, Median, Drittes Quartil und Maximum

s.voronoi: Berechnung von Thiessen-Polygonen aus Sites-Daten (wird abgelöst durch das Modul s.geom)

A.2.2 Vektordatenbefehle

v.alabel: Belegung aller ungelabelten Vektoren mit einem anzugebenden Wert

v.apply.census: Berechnungen mit demographischen Daten von CENSUS

v.area: Angaben zu Umfang und Fläche selektierter Vektorflächen

v.autocorr: Berechnungen zur Autokorrelation vektorisierter Flächen

v.cadlabel: Zuweisung von Labeln zu Vektorhöhenlinien, die im DXF-Format importiert wurden (auch mit v.digit möglich)

v.circle: setzt Vektorkreise mit definiertem Radius oder Fläche um Punkte (Sites)

v.clean: löscht unbenutzte Vektorinformationen in einer Vektordatei

v.cutter: erzeugt neue Vektordatei aus zwei Polygonkarten (Verschneidung)

v.digit: Digitalisier- und Vektor-Editierprogramm für Maus oder Digitalisierbrett

v.import: Import von ASCII oder binären Digital Line Graph (DLG)-Dateien

v.in.arc: liest aus ARC/INFO im „ungenerate-Format“ exportierte Vektordaten ein (vgl. Abschnitt A.4)

v.in.ascii: importiert eine ASCII-Vektordatei als binäre Vektordatei (Kategorien in der GRASS-Database unter `../dig_cat` und `../dig_att` ablegen)

v.in.dlg: importiert eine ASCII USGS DLG-3-Optional-Datei als eine binäre Vektordatei

v.in.dxf: importiert eine DXF-Datei als eine binäre oder ASCII-Vektordatei (Probleme gibt es mit den vermutlich zu Standard-DXF inkompatiblen SURFER 6-Dateien)

v.in.dxf3d: konvertiert DXF-Dateien mit Z-Werten in das GRASS Vektorformat

v.in.poly: erzeugt eine Vektordatei aus Polygonen mit spezifiziertem Radius um angegebene Mittelpunkte

v.in.shape: importiert Daten im ESRI SHAPE Format

v.in.tig.basic: importiert TIGER-Daten nach GRASS

v.in.tig.lndmk: importiert Census „Landmark“ features nach GRASS

v.in.transects: importiert „transects“ nach GRASS (Linien mit Koordinaten, Winkelabweichung oder Richtung und Länge)

v.line2area: konvertiert Vektorlinien zu Flächen (Script für v.in.dxf3d)

v.mkgrid: erzeugt eine Datei mit einem Vektorgitter

v.mkquads: erzeugt eine Sites-Liste oder ein region-definition-file für ein USGS 7.5-Minuten-Viereck (Winkelminuten)

v.out.arc: exportiert binäre Vektordaten ins „generate-Format“ von ARC/INFO

v.out.ascii: exportiert eine binäre Vektordatei in das ASCII-Format (liegt dann im aktuellen Verzeichnis)

v.out.dlg: exportiert eine binäre Vektordatei als ASCII USGS DLG-3-Optional-Datei

v.out.dxf: exportiert eine binäre Vektordatei als DXF-Datei

v.out.idrisi: exportiert eine binäre Vektordatei als ASCII IDRISI-Datei

v.out.moss: exportiert eine binäre Vektordatei als MOSS-Datei

v.patch: kombiniert Vektordaten zu einer neuen Vektorkarte

v.proj: projiziert eine Vektorkarte aus einer Quell-*location* in die aktuelle *location* mit neuer Projektion anhand der Projektionsdefinition der aktuellen *location*

v.prune: beseitigt überflüssige Knoten aus einer Vektordatei bei gegebener Toleranz (Threshold value)

v.reclass: erzeugt aus Vektordaten eine neue Karte mit reklassifizierten Kategorien

v.spag: korrigiert „Spaghetti-digitalisierte“ Vektordateien (fügt fehlende Knoten an Kreuzungen ein etc.)

v.stats: Ausgabe statistischer Daten über eine Vektordatei

v.support: erzeugt benötigte topologische Informationen für die Datenverwaltung (nach Import etc.)

v.surf.rst: erzeugt eines digitalen Höhenmodells mit regularisierten Spannungssplines aus Vektorhöhenlinien

v.to.rast: konvertiert eine Vektordatei in das GRASS-Rasterformat (Voraussetzung: alle Vektoren müssen ein Label haben, setzbar mit v.alabel oder v.digit)

v.to.sites: konvertiert eine Vektordatei in das GRASS-Sitesformat

v.transform: transformiert eine ASCII-Vektorkarte vom xy-System in das UTM-Koordinatensystem anhand von Passpunkten

v.trim: entfernt kleine Vorsprünge, überflüssige Kürzest-Vektoren und unnötige Knoten aus einer Vektordatei (z.B. nach Konvertierungen mit r.poly oder r.line)

A.2.3 Rasterdatenbefehle

r.average: Berechnung des Durchschnitts der Werte in einer Rasterdatei in allen Gebieten, die in einer Basisdatei (Maskendatei, z.B. erstellbar mit r.digit, mit zwei Kategorien, eine davon 0) festgelegt wurden

r.basins.fill: erzeugt eine Rasterkarte mit Wasser-Teileinzugsgebieten

r.bilinear: bilineare Interpolation von Rasterzellen zur Auflösungsverbesserung (nur für regelmäßige Raster)

r.binfer: Expertensystem auf Grundlage der Bayes'schen Entscheidungsregel für Landnutzungsklassifikationen (vgl. allgemeine Erläuterungen in BIENLEIN et al. 1996)

r.buffer: Bufferfunktion v.a. für Linienstrukturen in Rasterdateien (Zuweisung von bestimmten Werten bei den nächsten Nachbarn dieser Struktur über eine Entfernungsrechnung)

r.cats: druckt Kategorien und Labels von Rasterdateien

r.clump: rekategorisiert Rasterzellen durch Vereinheitlichung von nebeneinanderliegenden Zellen

r.cn: erzeugt eine Karte der SCS Curve numbers (rasterzellenorientiert).

r.coin: tabelliert die Koinzidenz von Kategorien zweier Rasterdateien

r.colors: erzeugt oder modifiziert die Farbtabelle einer Rasterdatei (dagegen Editieren einzelner Rasterzellen mit d.rast.edit)

r.colors.paint: wie r.colors, erlaubt aber die Benutzung von Farbtabellen für definierte Geräte (Devices)

r.combine: Überlagerung benutzergewählter Kategorien verschiedener Rasterdaten mit boolescher Algebra

r.contour: erzeugt eine Vektorhöhenlinienkarte aus einer Rasterkarte (Höhenmodell)

r.cost: Berechnung der kumulativen Kosten bei der Bewegung von einer zu einer anderen Zelle auf Basis einer Kostenfläche (enthält die Kosten bei Bewegung von einer Zelle zur nächsten)

r.covar: Berechnung der Kovarianz/Korrelationsmatrix für Rasterdaten

r.cross: Berechnung eines räumlichen Kreuzprodukts verschiedener übereinanderliegender Rasterflächen (Bestimmung aller Permutationen der Kategorien der Eingabedaten) – damit gut geeignet zur Plausibilitätskontrolle bei Flächenverschneidungen zusammen mit r.cats

r.describe: Ausgabe der Kategorien als Kurzliste

r.digit: erlaubt die Digitalisierung von Vektorlinien, -flächen und -kreisen in eine Rasterkarte, die entsprechenden Gebiete werden als Rasterdatei gespeichert (z.B. als Vorlage für r.mask oder r.average)

r.drain: Berechnung des günstigsten Pfades durch eine Rasterkarte auf Basis der kumulativen Kosten aus r.cost (z.B. zur Streckenplanung). Die Kostenfläche aus diesem Modul wird hier als Höhenmodell verwendet (je höher, desto teurer die Bewegung)

r.flow: berechnet Tiefenlinien, Längen der Tiefenlinien und Tiefenliniendichten sowie LS-Faktor

r.grow: lässt alle Flächen um eine Zelle in jede Richtung wachsen

r.hydro.CASC2D: komplexes hydrologisches Modell zur Abflussberechnung in einem Wassereinzugsgebiet. Eingangsdaten: Niederschlagsverteilung (räumlich und zeitlich), Interzeption, Infiltration, Oberflächenabflussverlauf (routing) mit Bodenrauhigkeit auf Grundlage eines Höhenmodells

r.in.ascii: Import von Rasterdaten im ASCII-Format (z.B. Höhenmodell)

r.in.gdal: Import von Rasterdaten in über 20 Formaten (empfohlen)

r.in.ll: importiert eine Längen- und Breitengrad-referenzierte Rasterkarte in UTM-*location*

r.in.poly: importiert ASCII-Polygon- und Linienvektoren als Rasterkarte

r.in.ppm: Import von Rasterdaten im PPM-Format (24 bit)

r.in.sunrast: Import von Rasterdaten im Sunraster-Format (8 bit)

r.in.tiff: Import von Rasterdaten im TIFF-Format (8 bit)

r.in.utm: importiert Rasterkarten in UTM-Koordinaten

r.infer: erlaubt die Erzeugung einer Rasterkarte unter Berücksichtigung von spezifizierbaren Kategorie-Kriterien

r.info: Ausgabe der Grundinformationen zu einer Rasterdatei

r.le.dist: Landschaftsstrukturanalyse: Berechnung der Distanzen zwischen einzelnen Parzellen (verschiedene Methoden) (le: landscape ecology, Grundlage sind Rasterkarten, Satellitenbilder etc.)

r.le.null: Landschaftsstrukturanalyse: erzeugt eine Karte mit neutraler Struktur als „Null-Hypothese“ für Signifikanztests

r.le.patch: Landschaftsstrukturanalyse: Berechnung von Attributen, Größe, Kerngröße, Form, Komplexität und Umfang von Parzellen

r.le.pixel: Landschaftsstrukturanalyse: Berechnung von Reichhaltigkeit, „Shannon Index“, „Inverser Simpson Index“, Textur etc.

r.le.rename: Landschaftsstrukturanalyse: Umbenennung von Dateinamen

r.le.setup: Landschaftsstrukturanalyse: Festlegung von Untersuchungsraum und Untersuchungsart (stationär, moving windows etc.)

r.le.trace: Landschaftsstrukturanalyse: Anzeige der Parzellengrenzen, Form, Größe etc.

r.line: automatische Vektorisierung von Linienstrukturen in einer Rasterdatei (vgl. r.poly und r.thin)

r.linear.regression: berechnet die Koeffizienten einer linearen Regression aus einer ASCII-Datei, die im aktuellen Verzeichnis liegt

r.los: Sichtlinienanalyse (los = line of sight) von einem bestimmten Standpunkt aus unter bestimmtem Höhenwinkel der Blickrichtung

r.mapcalc: umfangreiches Rechenmodul für arithmetische Berechnungen (zugehöriges Handbuch: LARSON ET AL. 1991, SHAPIRO ET AL. 1992)

r.mask: erlaubt die Ausmaskierung bestimmter Gebiete in einer Rasterdatei (dazu Anlegen einer Rasterdatei mit den Werten 0 und 1 nötig, diese wird intern mit der ersten Rasterdatei multipliziert) . Wenn eine Maske nicht-interaktiv gesetzt werden soll (z.B. in Scripten), kann sie mit g.copy gesetzt werden: `g.copy rast=meinemaske,MASK`

r.mask.points: prüft, ob gegebene Punkte in einer gesetzten Maske liegen

r.median: berechnet den Median aller Rasterzellen in einer Coverkarte, die geographisch in Bezug zu den Zellen derselben Kategorie in einer Basiskarte stehen

r.mfilter: filtert eine Rasterdatei mit einem als ASCII-Datei abzulegenden Matrixfilter

r.mode: berechnet den Modus aller Rasterzellen in der Coverkarte, die geographisch in Bezug zu den Zellen derselben Kategorie in der Basiskarte stehen

r.ndvi.model: berechnet den NDVI (normalized difference vegetation index) über eine Regressionsrechnung aus einer ASCII-Datei, die im aktuellen Verzeichnis liegt

r.neighbors: Berechnung von Durchschnitt, Median, Modus, Minimum, Maximum, Diversität und Verteilung in einer definierbaren Matrix um eine Zelle über das gesamte Bild als neue Karte (das Rechenfenster „wandert“, Matrix maximal 25x25 Pixel, größere Matrizen sind mit r.mapcalc programmierbar)

r.out.ascii: Rasterexport im ASCII-Format mit Header

r.out.mpeg: Rasterexport im MPEG-Format

r.out.ppm: Rasterexport im PPM-Format (24 bit)

r.out.ppm3: Rasterexport nach Grundfarben getrennter Rasterdateien (RGB) in eine neue Datei im PPM-Format (je 8 bit)

r.out.tga: Rasterexport im TGA-Format (24 bit)

r.out.tiff: Rasterexport im TIFF-Format (8 bit)

r.patch: fügt Rasterbilder zu einem neuen Rasterbild zusammen (Überlagerung sowie gegenseitiges Ausfüllen von „no data“-Gebieten möglich)

r.poly: automatische Vektorisierung von Polygonstrukturen in einer Rasterdatei (vgl. r.line und r.thin)

r.profile: Ausgabe von Kategorien bzw. deren Median oder Durchschnitt, die auf einer benutzerdefinierten Linie liegen

r.random: erzeugt eine Rasterdatei mit zufällig verteilten Punkten

r.rational.regression: berechnet die Koeffizienten einer rationalen Regression aus einer ASCII-Datei, die im aktuellen Verzeichnis liegt, über Iterationen

r.reclass: Reklassifizierung einer Rasterdatei über Zuweisung generalisierter Informationen zu den einzelnen Kategorien

r.report: Ausgabe statistischer Informationen zu einer Rasterdatei

r.resample: Resampling einer Rasterdatei (Anpassung der Auflösung an die aktuelle Auflösung in einer Region (setzbar mit g.region), vgl. auch r.surf.idw2) nach nearest-neighbour-Methode

r.rescale: Reskalierung von Kategorien in bestimmten Wertebereichen (von min.alt bis max.alt nach min.neu bis max.neu, Rest wird 0, auch zur Histogrammverkürzung geeignet (vgl. r.colors))

r.rescale.eq: wie r.rescale, alternativer Algorithmus, vgl. Modulanleitung

r.slope.aspect: Berechnung von Hangneigung (slope) und Exposition (aspect) aus einem Höhenmodell. Die Expositionskarte kann zur Schummerung in d.3d verwendet werden

r.stats: Ausgabe statistischer Informationen zu einer Rasterdatei

r.sun: Berechnet eine Karte des Sonnenstrahlungs-Einfallswinkels zu vorgegebener Zeit, geographischer Position, Höhe, Hangneigung, Exposition und Abschattung durch das jeweils umliegende Gebiet

r.support: zur „Nachbehandlung“ einer Rasterdatei nach Import (setzen der Randkoordinaten, Berechnung der Kartenstatistik) usw.

r.surf.area: Berechnung von Oberflächen bei geneigten Flächen (Höhenmodell etc.)

r.surf.contour: Berechnung eines Höhenmodells aus einer gerasterten Höhenlinienkarte (mit v.to.rast)

r.surf.fractal: dient der Erzeugung synthetischer Höhenmodelle durch Angabe ihrer fraktalen Dimension D

r.surf.gauss: dient der Erzeugung von Flächen mit Werten gemäß einer Gaußschen Verteilungskurve

r.surf.idw: Oberflächeninterpolation, um „Löcher“ (Wert 0) in einem Höhenmodell über einen gewichteten Durchschnitt zu füllen (für *locations* mit Koordinaten in Längen-/Breitenangabe, Algorithmus: inverse distance weighted)

r.surf.idw2: Oberflächeninterpolation, um „Löcher“ (Wert 0) in einem Höhenmodell über einen gewichteten Durchschnitt zu füllen (für *locations* mit Koordinaten in xy- oder Gauß-Krüger-Projektion, Algorithmus: inverse distance weighted). Ebenso geeignet, um die Auflösung im Höhenmodell hochzuinterpolieren (vorher mit g.region Auflösung erhöhen, vgl. auch r.resample, r.resamp.rst, s.surf.rst, v.surf.rst)

r.surf.random: dient der Erzeugung von Flächen mit Zufallswerten

r.thin: Verdünnung von Rasterlinien als Vorbereitung zum automatischen Vektorisieren (mit r.line, r.poly)

r.transect: Ausgabe von Kategorien bzw. deren Median oder Durchschnitt, die auf einem benutzerdefinierten Transect liegen

r.univar: berechnet univariate Statistik aus Rasterdaten: Zellenanzahl, arithmetisches Mittel, Standardabweichung, Variationskoeffizient, Mininum, Median und Maximum

r.volume: Berechnung des (kubischen) Volumens von Zellengruppen (gleiche Kategorie) und Ausgabe als Tabelle oder Sites-Datei (dort an die Mittelpunkte der Zellengruppen gesetzt)

r.water.outlet: Modul zur Berechnung von Wassereinzugsgebieten an frei definierbaren Punkten im DGM

r.watershed: umfangreiches Modul zur Berechnung von Wassereinzugsgebieten (Gesamtgebiete)

r.weight: Berechnung einer neuen Rasterdatei aus einzelnen, unterschiedlich gewichteten Kategorien anderer Rasterdateien (nicht interaktiv)

r.weight.new: Berechnung einer neuen Rasterdatei aus einzelnen, unterschiedlich gewichteten Kategorien anderer Rasterdateien (interaktiv)

r.what: Ausgabe von Kategorien einer Rasterkarte an benutzerdefinierten Stellen (manuell anzugeben oder per ASCII-Datei, vgl. mausgesteuerte Abfrage mit d.what.rast)

A.2.4 Bildverarbeitungsbefehle

i.cca: Berechnung der kanonischen Komponententransformation (maximale Trennung zwischen den Signaturen in Rasterbildern, vgl. i.pca)

i.class: Modul zur überwachten Klassifizierung (Eingrenzen von Testflächen, Test auf Signaturverteilung, Klassenbildung)

i.cluster: Modul zur un- und teilüberwachten Klassifizierung (über Berechnung von Clustermittelwerten und Kovarianzmatrizen)

i.colors: Zuweisung einzelner Kanäle (Satellitenbilder) zu den Farben Rot, Grün und Blau (mit Ausgabe im GRASS-Monitor, vgl. d.rgb)

i.composite: erzeugt ein Farbkomposit aus drei Kanälen (Satellitenbilder)

i.fft: Fouriertransformation einer Rasterdatei im Bildbereich (Luftbild, Satellitenbild) in den Frequenzbereich (reales und imaginäres Spektrum) z.B. zu Filterzwecken (vgl. i.ifft, i.shape)

i.gensig: automatische Erzeugung von Signaturen für i.maxlik (statt i.cluster oder i.class) aus einer „Trainingsrasterkarte“ (Testflächenkarte – ground truth training map, mit v.digit und v.to.rast, bzw. v.in.transects oder r.digit erstellbar) für eine überwachte Klassifizierung

i.gensigset: automatische Erzeugung von Signaturen für i.smap aus einer „Trainingsrasterkarte“ (Testflächenkarte – ground truth training map, mit v.digit und v.to.rast, bzw. v.in.transects oder r.digit erstellbar)

i.group: fasst die zu analysierenden Satellitenbilder in einer Bildgruppe zusammen

i.his.rgb: Farbtransformation aus Farbe (Hue), Intensität (Intensity), Sättigung (Saturation) nach Rot, Grün, Blau

i.ifft: inverse Fouriertransformation einer Rasterdatei im Frequenzraum zurück in den Bildraum (beispielsweise nach einer Filterung, vgl. i.fft)

i.in.erdas: importiert ERDAS-Daten im ERDAS 7.4 Format (4, 8, 16bit Rasterdaten)

i.maxlik: Zuweisung der Signaturen in einer Satellitenbildszene über eine „maximum likelihood analysis“ (vgl. i.smap); als Grundlage werden die Ergebnisse aus i.cluster (un- und teilüberwacht),

i.class (überwacht) oder i.gensig (überwacht) verwendet, jedes Pixel wird einer Klasse zugeordnet – radiometrisch/statistisches Klassifikationsverfahren (spektral pattern analysis)

i.ortho.photo: Modul zur Herstellung von Orthofotos aus Luftbildern mit einem Höhenmodell

i.pca: Hauptkomponententransformation (minimale Trennung zwischen den Signaturen in Rasterbildern, vgl. i.cca)

i.points: Modul zum Setzen von Passpunkten (per Tastatur als Koordinatenpaar oder per Maus) für eine Affin- oder eine Polynomtransformation (i.rectify). Nebeneinander können zwei Rasterkarten dargestellt werden: zu geocodierende Rasterkarte und Referenz-Rasterkarte aus der geocodierten *location* (vgl. i.vpoints)

i.quantize: erzeugt eine neue Rasterdatei mit einer Farbtabelle basierend auf den in einer Bildgruppe vorkommenden Farbwerten für Rot, Grün und Blau

i.rectify: Modul zur Affin- oder Polynomtransformation zur Entzerrung von in sich verzerrten Rasterbildern etc. auf topographische Karten

i.rgb.his: Farbtransformation aus Rot, Grün, Blau nach Farbe (Hue), Intensität (Intensity), Sättigung (Saturation)

i.shape: Berechnung eines fouriertransformierten Rasterbildes über eine Gauß-Filterung (Rückrechnung mit i.ifft)

i.smap: Zuweisung der Signaturen in einer Satellitenbildszene über eine „sequential maximum a posteriori (SMAP) estimation“ (vgl. i.maxlik). Als Grundlage werden die Ergebnisse aus i.gensigset verwendet, die Pixel werden im regionalen Zusammenhang betrachtet – geometrisch/radiometrisch/ statistisches Klassifikationsverfahren (spatial pattern analysis)

i.tape.mss: Extraktionsmodul für LANDSAT-MSS-Bilder auf 0,5 Zoll-Bändern

i.tape.mss.h: Extraktionsmodul für die Headerinformationen der LANDSAT-MSS-Bilder auf 0,5 Zoll-Bändern

i.tape.other: Extraktionsmodul für SPOT-, NHAP-Bilder etc. auf 0,5 Zoll-Bändern

i.tape.tm: Extraktionsmodul für LANDSAT-TM-Bilder auf 0,5 Zoll-Bändern

i.tape.tm.fast: schnelles Extraktionsmodul für LANDSAT-TM-Bilder auf 0,5 Zoll-Bändern

i.target: Zuweisung einer Ziel-*location* und -*mapset* für Transformationsrechnungen (i.rectify)

i.texture: Texturanalyse in verschiedenen Richtungen (Berechnung von Kontrast, Korrelation, Varianz, Durchschnitt, etc.)

i.vpoints: Modul zum Setzen von Passpunkten (per Tastatur als Koordinatenpaar oder per Maus) für eine Affin- oder eine Polynomtransformation (i.rectify). Nebeneinander können eine Raster- und eine Vektorkarte dargestellt werden: Entzerrung einer Rasterkarte auf eine Vektorkarte (vgl. i.points)

i.zc: Berechnung von Strukturrändern in Rasterbildern („edge detection mit zero crossing“) über Fourier- und Laplacetransformation in Verbindung mit einem definierbaren Gaußfilter

A.2.5 Display-Befehle

d.3d: Erzeugung dreidimensionaler Ansichten von Rasterdaten als Blockbild (z.B. perspektivische Überlagerung eines Höhenmodells mit Rasterdaten möglich), wird von nviz abgelöst

d.ask: Modul für shell-Scripte, um nach Dateien zu fragen (benutzerdefinierbares Menü im Grafikmonitor)

d.barscale: zeigt einen Maßstabsbalken im Grafikmonitor

d.colormode: Umschaltung zwischen eigener (rasterdateiabhängiger) und standardisierter Farbtabelle für den Grafikmonitor

d.colors: Direkte Modifikation der im Grafikmonitor dargestellten Farben

d.colortable: Anzeige der Farbtabelle eines Rasterbildes

d.display: Menüführung zur Datendarstellung im Grafikmonitor

d.erase: löscht den aktiven Grafikmonitor (v.a. nach g.region nötig)

d.font: Auswahl des Zeichensatzes für den Grafikmonitor

d.frame: Handhabung von Display-Rahmen (frames) im Grafikmonitor

d.geodesic: zeigt eine geodätische Linie (kürzeste Verbindung zwischen zwei Punkten entlang des Großkreises) bei *locations* mit Koordinaten in Längen-/Breitenangabe

d.graph: Ausgabe einfacher Grafiken auf dem Grafikmonitor aus ASCII-Datei mit Zeichenbefehlen (xy-Koordinaten des Grafikmonitors, vgl. d.mapgraph)

d.grid: Ausgabe eines Gitters im Grafikmonitor

d.his: Anzeige eines kombinierten Rasterbildes aus drei Rasterdateien für Farbe (Hue), Intensität (Intensity), Sättigung (Saturation)

d.histogram: Anzeige eines Histogramms einer Rasterdatei in Torten- oder Balkendarstellung im Grafikmonitor

d.icons: Ausgabe von Punktdaten als Symbole im Grafikmonitor (Erzeugung mit p.labels, Koordinatenangaben idealerweise in Datei, dafür Koordinatensuche mit d.where)

d.label: erzeugt und zeigt Textlabel an (unkomfortabel, vgl. d.labels)

d.labels: erzeugt und zeigt Textlabel an für PPM-Karten (vgl. p.map)

d.legend: Ausgabe einer Legende zu einer Rasterdatei

d.linegraph: zeigt Liniengrafen an (eine Datei für x-Werte, eine zweite für y-Werte nötig)

d.mapgraph: Ausgabe einfacher Grafiken auf dem Grafikmonitor aus ASCII-Datei mit Zeichenbefehlen (geographische Koordinaten, vgl. d.graph)

d.measure: Messung von Strecken oder Flächen per Maus im Grafikmonitor

d.menu: Modul für Shell-Scripts zur Erzeugung von Menüs im GRASS-Monitor

d.mon: Handhabung der Grafikmonitore (Start/Stop)

d.paint.labels: Ausgabe von Labels, die mit p.labels erzeugt wurden

d.pan: weiterbewegen der aktiven Region durch Festlegung eines neuen Regionzentrums per Maus

d.points: Ausgabe von Punkten an benutzerdefinierter Stelle

d.profile: Ausgabe eines Profilschnitts in einer Rasterdatei über eine benutzerdefinierte Linie (Transect)

d.rast: Ausgabe bzw. Überlagerung von Rasterdateien im aktiven Rahmen des Grafikmonitors

d.rast.arrow: Ausgabe von Pfeilen, um Fließrichtungen auf einer Oberfläche besser zu visualisieren

d.rast.edit: Interaktives Editieren von Rasterzellen per Maus, die editierten Rasterzellen werden in einer neuen Datei gespeichert. Verbinden lassen sich die alte und die neue Datei mit r.patch oder r.mapcalc

d.rast.num: Ausgabe der Kategorien über einer Rasterkarte

d.rgb: zeigt drei Rasterdateien als Rot-, Grün- und Blauüberlagerung

d.rhumbline: Ausgabe einer Linie zwischen definierbaren Punkten, die einem bestimmten Kompasswinkel folgt (für *locations* mit Koordinaten in Längen-/Breitenangabe)

d.save: Erzeugung eines shell-Scripts über bisher erfolgte Ausgaben auf dem Grafikmonitor, um diese Darstellung zu einem späteren Zeitpunkt einfach wiederholen zu können

d.scale: Ausgabe eines Maßstabs und Nordpfeils an definierbarer Stelle

d.site.labels: Anzeige bestimmter oder aller Site-Label

d.sites: Ausgabe von Sites im Grafikmonitor

d.text: Ausgabe von Text im Grafikmonitor

d.title: Ausgabe eines Titels im Grafikmonitor

d.vect: Ausgabe bzw. Überlagerung von Vektordateien im aktiven Rahmen des Grafikmonitors

d.vect.area: Ausgabe definierbarer und optional flächengefüllter Vektoren im Grafikmonitor

d.what.rast: Abfrage von Kategorien (Category values) in Rasterdateien im aktiven Rahmen an benutzerdefinierter Stelle per Maus (vgl. dateigesteuerte Abfrage mit r.what)

d.what.vect: Abfrage von Kategorien (Category values) in Vektordateien im aktiven Rahmen an benutzerdefinierter Stelle per Maus

d.where: Ausgabe von Koordinaten von Punkten, die mit der Maus ausgewählt werden können

d.zoom: komfortables Zoommodul (per Maus)

A.2.6 PPM-Ausgabebefehle

p.chart: druckt eine Farbtabelle mit Nummerierung auf einem Farbdrucker

p.colors: Zuweisung von druckerabhängigen Farben (vgl. p.chart) zu den Kategorien anstelle von RGB-Werten

p.icons: erzeugt und modifiziert Icons (Kartensymbole) als einzelne Icondateien mit anzugebendem Texteditor (z.B. „textedit“), Ausgabe mit d.icons möglich

p.map: Modul zur Erstellung von Karten (für PPM-Treiber (vgl. p.select), alternativ: vgl. ps.map)

p.map.new: vgl. p.map

p.ppm: liest eine PPM-Datei ein und gibt sie auf dem mit p.select auswählbaren Device aus

p.labels: Bearbeitung der Labelinformationen für die Kartengestaltung (Erzeugung mit d.labels)

p.select: wählt ein Druckerdevice aus (preview oder Speicherung als PPM-Datei, die PPM-Dateien können später mit xv in Postscript umgewandelt werden, bei der Erstellung von Karten mit p.map/ p.map.new sollte die Auflösung in dpi beachtet werden)

A.2.7 Postscript-Druckbefehle

ps.icon: erzeugt und modifiziert Icons (Kartensymbole) als einzelne Icondateien graphisch über den GRASS-Monitor

ps.map: Modul zur Erstellung von Karten (für Postscript-Treiber (vgl. ps.select), alternativ: vgl. p.map)

ps.select: wählt einen Postscript-Druckertreiber aus (Speicherung in Datei), Druckerdefinitionen können selbst angelegt werden (siehe Anhang)

A.2.8 Allgemeine Befehle

g.access: Modul zur Steuerung des Zugriffs auf einzelne *mapsets* durch andere Benutzer

g.ask: Abfrage der GRASS-Database (für shell-Scripts)

g.copy: Anlegen von Kopien einzelner Dateien in der GRASS-Database

g.filename: Ausgabe von Dateinamen (für shell-Scripts)

g.findfile: sucht nach Dateien in GRASS-Databases (für shell-Scripts)

g.gisenv: Ausgabe der GRASS-Umgebungsvariablen

g.list: Liste aller vorhandenen Dateien einer speziellen Sorte (Raster-, Vektordaten etc.)

g.manual: digital verfügbare GRASS-Modulanleitungen (Aufruf: `$ g.manual modulename`)

g.mapsets: ermöglicht den Zugriff auf andere *mapsets* in einer *location* (interessant für Teamarbeit)

g.region: Management der Grenzen und Auflösung der aktuellen *mapset* (anschließend immer d.erase aufrufen)

g.remove: Löschen von Dateien (Raster-, Vektordaten etc.) in der aktuellen *location*

g.rename: Umbenennen von Dateinamen in der aktuellen *location*

g.setproj: setzt die Projektion in der aktuellen *location*, wenn bei Einrichtung vergessen. Achtung: dabei keine Neuprojektion von Daten!

g.tempfile: erzeugt eine Temporärdatei und gibt ihren Namen aus (für Shell-Scripts)

g.version: Ausgabe der GRASS-Versionsnummer

A.2.9 Verschiedene Befehle und Kartenprojektions-Befehle

m.datum.shift: Modul zur Koordinatenberechnung bei Verschiebung des Datums eines Projektionsellipsoiden

m.dem.examine: Ausgabe einer Kurzbeschreibung von USGS Digital Elevation Model (DEM) Daten, gespeichert auf 0,5 Zoll-Bändern

m.dem.extract: Extraktion von USGS Digital Elevation Model (DEM)-Daten, gespeichert auf 0,5 Zoll-Bändern

m.dmaUSGSread: Extraktion von USGS Digital Terrain Elevation Data (DTED), gespeichert auf 0,5 Zoll-Bändern

m.dted.examine: Ausgabe einer Kurzbeschreibung von DMA Digital Terrain Elevation data (DTED)-Daten, gespeichert auf 0,5 Zoll-Bändern

m.dted.extract: Extraktion von DMA Digital Terrain Elevation Data (DTED)-Daten, gespeichert auf 0,5 Zoll-Bändern

m.examine.tape: Untersuchung der Parameter von 0,5 Zoll-Bändern

m.flip: „klappt“ Koordinaten in Datei im aktuellen Verzeichnis an der Ost-West-Achse „um“ (Tausch Nord-Süd, nur für externe ASCII-Daten)

m.gc2ll: konvertiert geozentrische in geographische Koordinaten

m.in.e00: importiert Daten im ESRI-E00 Format

m.in.pl94.db3: importiert demographische Datensätze von Census PL94-171 dBase3 CDROM-Dateien

m.in.stf1.db3: importiert demographische Datensätze von Census STF1A dBase3 CDROM-Dateien

m.in.stf1.tape: importiert demographische Datensätze von Census STF1A dBase3 Textdateien

m.linear.regression: berechnet aus einer ASCII-Datei im aktuellen Verzeichnis lineare Regression und verschiedene Vegetationsindizes

m.ll2gc: konvertiert geographische in geozentrische Koordinaten

m.ll2u: konvertiert geographische in UTM-Koordinaten

m.lulc.USGS: erzeugt Rasterdaten von Composite Theme Grid CTG-Dateien, die von m.lulc.read erzeugt wurden

m.lulc.read: extrahiert Landuse/Landcover-Daten in das ASCII Composite Theme Grid (CTG)-Datenformat vom USGS

m.proj: Umrechnungsmodul für verschiedene Koordinatensysteme. Dieses Modul kann (ab GRASS 5.0.x) als Koordinaten-Taschenrechner für 121 Projektionen benutzt werden

m.region.ll: konvertiert die aktuellen UTM-Koordinaten in einer Region in Längen-/Breitengrade

m.rational.regression: berechnet aus einer ASCII-Datei im aktuellen Verzeichnis lineare, rationale Regression und verschiedene Vegetationsindizes

m.rot90: rotiert Höhendaten, die entweder von m.dted.extract oder m.dmaUSGSread extrahiert wurden, um 90 Grad

m.tiger.region: sucht aus einer TIGER-Datei die geographischen Informationen (Region etc.) heraus

m.u2ll: konvertiert UTM zu geographischen Koordinaten

A.3 Die Struktur der GRASS-Datenbank

Die in GRASS gespeicherten Daten werden üblicherweise im HOME-Verzeichnis innerhalb eines eigenen Verzeichnisbaums als GRASS-Datenbank (*database*) gespeichert. Die Struktur sieht folgendermaßen aus (Beispiel):

/home/stefan/grassdata
/grassdata/location1
/grassdata/location1/PERMANENT/
/grassdata/location1/mapset1/
/grassdata/location1/mapset1/cell

/grassdata/location1/mapset1/dig
/grassdata/location1/mapset1/group/groupname/
/grassdata/location1/mapset1/...

/grassdata/location1/mapset2/
/grassdata/location1/mapset2/...

/grassdata/location2
/grassdata/location2/PERMANENT/
/grassdata/location2/mapset1/...

Die *mapset*-Verzeichnisse beinhalten jeweils Teilgebiete eines Projektgebiets, in PERMANENT sind die grundsätzlichen Projektionsinformationen und die *location*-Ausdehnung gespeichert. Vom direkten Bearbeiten der Daten in den Unterverzeichnissen ist dringend abzuraten, da die Struktur sonst nicht mehr konsistent ist. Die einzige Ausnahme ist bei der Datenumwandlung ARC/INFO nach GRASS (vgl. Abschnitt A.4) anzutreffen.
Arbeiten verschiedene Nutzer in einer GRASS-location, gelten folgende Rechte: In der *mapset* PERMANENT können alle Nutzer/-innen lesen und schreiben, in den anderen nur die individuellen Besitzer der jeweiligen *mapset*. Das Austauschverzeichnis ist also die PERMANENT-*mapset*, derweil die anderen *mapsets* jeweils nur einer Person gehören können. So lässt sich die Projektarbeit im Team gut organisieren, und es entstehen keine Konflikte bezüglich der Änderung von Dateien. Nähere Erläuterungen zur Bedeutung der verschiedenen Unterverzeichnisse finden sich bei ALBRECHT 1992 und im GRASS-Programmierhandbuch.

A.3.1 Löschen von GIS-Daten in GRASS

Einzelne Raster-, Vektordateien etc. werden mit `$ g.remove` entfernt. Das Modul ist selbsterklärend. Sollen einzelne Dateien vorher auf Datenbänder etc. gerettet werden, können sie mit `$ r.out.tiff`, `$ r.out.ppm`, `$ v.out.ascii` etc. exportiert werden (vgl. zum Export im GIF-Format Abschnitt 6.8).

Um eine *mapset* zu löschen, kann der entsprechende Menüpunkt in TclTkGRASS benutzt werden. Alternativ wird GRASS verlassen und das entsprechende *mapset*-Unterverzeichnis in der GRASS-Datenbank mit dem UNIX-„rm“-Befehl oder einem Dateimanager gelöscht.
Eine *location* kann dagegen nur außerhalb von GRASS entfernt werden, GRASS ist dazu also zu verlassen. Das UNIX-Kommando
`$ rm -rf <locationname>`
führt zum Entfernen der ganzen *location* (rekursives Löschen). Für „location“ ist Verzeichnisname (ohne Anführungszeichen), der dem *location*-Namen entspricht, einzusetzen. Das Befehl für das rekursive Löschen kann durch `$ unalias rm` von den Sicherheitsrückfragen bei jeder Datei „be-

freit“ werden (dann vorsichtig mit diesem Befehl umgehen, es gibt keine Wiederherstellungsfunktion!). Anschließend wird mit `$ alias rm='rm -i'` die Abfrage wieder aktiviert. Alternativ können Sie auch einen Dateimanager verwenden.

A.3.2 Kopieren einer GRASS-Datenbank

Wenn Sie die Daten eines Projektgebiets (*location*) auf einen anderen Rechner oder in ein neues Verzeichnis übertragen wollen, sollten Sie die Daten erst „einpacken“ und dann als Paket bewegen. Am Zielort wird der Datensatz wieder „ausgepackt“.
Wechseln Sie zunächst in das Verzeichnis der GRASS-Datenbank (GRASS vorher verlassen):

```
$ cd grassdata
```

Nun wird die gewünscht *location* mit dem „tar“-Packer in das Paket „meinelocation.tar“ eingepackt:

```
$ tar cvf meinelocation.tar location/
```

Sie können dieses Paket anschließend noch komprimieren:

```
$ gzip meinelocation.tar
```

Das Paket erhält die Dateiendung „.gz“ und sollte kleiner geworden sein. Nun können Sie dieses Paket an einen anderen Ort kopieren.

Dort sollten Sie wieder ein Verzeichnis „grassdata“ erzeugen. In dieses Verzeichnis wird das Paket kopiert und folgendermaßen ausgepackt:

```
$ gunzip meinelocation.tar.gz
$ tar xvf meinelocation.tar
```

Bitte verwechseln Sie keinesfalls die Optionen „c“ und „x“ bei „tar“, da mit „c“ das Paket überschrieben wird („c“ dient ja zur Erzeugung von Paketen).
Nach dem Auspacken der *location* auf dem Zielrechner steht die kopierte GRASS-Datenbank dort zur Verfügung und kann nach dem Aufruf von GRASS in der Eingangsmaske angegeben werden.

A.4 Konvertierung externer GIS-Formate für GRASS

Für viele Projektarbeiten ist es unerlässlich, Daten aus anderen GIS zusammenzutragen. GRASS bietet verschiedene Module zum Import dieser Daten an (für Idrisi, ERDAS, ARC/INFO, MOSS etc. und ASCII). Das einfachste Format ist das ASCII-Format, für das GRASS Schnittstellen (r.in.ascii, v.in.ascii, s.in.ascii) besitzt. Einige GIS kennen dieses Austauschformat allerdings nicht – möglicherweise kann dort jedoch das ARC/INFO-Format zum Export gewählt werden. Im Folgenden wird gezeigt, wie Daten aus ARC/INFO, sogenannte „coverages“, für GRASS lesbar exportiert und in GRASS importiert werden. Es wird allerdings nur auf Vektor- und Punktdaten eingegangen,

da der Im- und Export von Rasterdaten kein Problem darstellt (möglich über die TIFF-/ SUN-Raster-Bildformate bzw. ARC-ASCII-GRID etc.).

A.4.1 Export aus ARC/INFO

Die Konvertierung von Punkt- und Vektordaten beginnt in ARC/INFO. Dabei ergeben sich geringe Abweichungen zwischen der DOS und der UNIX-Version, auf die an den differierenden Stellen hingewiesen wird. Als Erstes muss überprüft werden, ob sich die Daten überhaupt verwenden lassen. Das coverage kann nur in GRASS importiert werden, wenn es sich um ein „polygon“ oder „line coverage“ handelt. Um diese Bedingung zu prüfen, muss man in die dem „coverage“ (das mit einem Unterverzeichnis identisch ist) übergeordnete Verzeichnisebene wechseln und ARC/INFO mit `arc` starten. Die entsprechende Information liefert dann der Befehl:

`arc> describe <coverage>`

also z.B. `arc> describe topo12`

Ist diese Bedingung erfüllt (liegt also ein „line“ oder „polygon coverage“ vor), kann der Export beginnen. Falls die Koordinaten des coverages noch nicht bekannt sind, können auch sie mit diesem Befehl abgefragt werden. Wenn nur „arcs“ und keine „polygons“ vorliegen, kann eine Konvertierung mit

`arc> clean <coverage> <coverageneu>`

vorgenommen werden. Die Daten werden in einem neuen „polygon coverage“ abgelegt.
Nun wird der „ungenerate“-Befehl benutzt, um die Vektordaten im ASCII-ARC-Format in Dateien abzulegen. Die allgemeine Syntax lautet:

UNIX: `arc> ungenerate <Datentyp> <coverage> <exportfile>`

DOS: `arc> ungen <Datentyp> <coverage> <exportfile>`

Je nach coverage bestehen nun zwei Möglichkeiten:

i. Export aus line coverages

Exportiert werden müssen nur die Linienvektoren. Als Endung für das Linien-Exportfile hat sich „.lin“ etabliert:

UNIX: `arc> ungenerate line topo12 topo12.lin`

DOS: `arc> ungen line topo12 topo12.lin`

Weiter geht es mit dem Export der Attribute.

ii. Export aus polygon coverages

Zuerst werden die Polygone (bzw. Arcs) exportiert, das coverage heißt hier „topo12“. Als Endung für das Polygon-Exportfile hat sich „.pol“ etabliert:

UNIX: `arc> ungenerate line topo12 topo12.pol`

DOS: `arc> ungen line topo12 topo12.pol`

Im zweiten Schritt geht es um die Punktdaten mit der Exportfile-Endung „.lab“, die nur bei „polygon coverages“ existieren:

UNIX: `arc> ungenerate point topo12 topo12.lab`

DOS: `arc> ungen point topo12 topo12.lab`

Weiter geht es mit dem Export der Attribute.

Export der Attribute

Im letzten Schritt steht die Speicherung der Attribute an (label text). Da diese Informationen im INFO-Teil von ARC/INFO abgelegt sind, gelten für DOS und UNIX unterschiedliche Vorgehensweisen.

(a) In *UNIX-ARC/INFO* muss das INFO-Programm zunächst gestartet werden:

```
arc> info
```

Der nun abgefragte User heißt „arc“. Die Attribute werden mit folgenden Befehlen als ASCII-Datei gespeichert (evtl. caps-lock einschalten, da Großschrift erforderlich):

```
ENTER COMMAND> SELECT TOPO12.PAT
ENTER COMMAND> OUTPUT ../topo12.txt
ENTER COMMAND> LIST PRINT
ENTER COMMAND> Q STOP
```

Die Datei `topo12.txt` sieht z.B. folgendermaßen aus:

```
$RECNO AREA PERIMETER TOPO12# TOPO12-ID TOPO12-NAME
1 -2.34543E+09 216,734.800 1 0
2 3.764578E+08 213,445.100 2 5 Nordstadt
3 26335673.000  43,567.120 3 6 Oststadt
4 5.684567E+04 233,235.400 4 2 Weststadt
```

usw.

Damit ist unter UNIX der Export abgeschlossen, ARC/INFO kann mit
`arc> quit` verlassen werden.

(b) Für *DOS-ARC/INFO* müssen die Attribute auf andere Art gespeichert werden. PC-ARC/INFO legt diese Daten im dBase-Format ab. Die Datei „pat.dbf“, die sich im coverage-Verzeichnis befindet, enthält die gewünschten Informationen. Sie muss in „dBase“ geladen („use dbf-Datei“) und als Fremddatei (Abgrenzung der einzelnen Werte durch Leerzeichen) gespeichert werden (F2=assist-Modul, das den Export menügesteuert ermöglicht). Diese Exportdatei kann dann nach UNIX kopiert (für GRASS) und mit einem ASCII-Editor (textedit, emacs etc.) bearbeitet werden. Es fehlt nämlich der header, also der Kopf der Datei, der in der UNIX-Version mitexportiert wird.
Dieser Dateikopf ist per Hand als erste Zeile einzutragen (in diesem Beispiel in topo12.txt):

```
AREA PERIMETER TOPO12# TOPO12-ID TOPO12-NAME
```

Damit ist der Export unter DOS abgeschlossen, ARC/INFO kann mit
`arc> quit` verlassen werden.
Ein Problem stellen Leerzeichen in den Attributen dar, wenn kein externes Datenbankprogramm benutzt wird. Das Importmodul unter GRASS akzeptiert in diesem Fall nur das erste Wort. Die einzige Lösung besteht im manuellen Austausch der Leerzeichen durch Bindestriche bzw. andere Zeichen in der Attributdatei (die Datei mit der Endung .txt, in diesem Fall topo12.txt).
Ein wichtiger Hinweis: GRASS ist ohne angeschlossene externe Datenbank (RIM, Postgrass) nur in der Lage, ein Attribut (Textlabel) pro Linie oder Punkt zu speichern. Es wird also nur das erste Attribut in jeder Zeile berücksichtigt, alle weiteren gehen verloren.

A.4.2 Import in GRASS

In GRASS muss eine *location* mit den entsprechenden Koordinaten vorhanden sein bzw. angelegt werden. Die drei Dateien (Endungen .lin, .lab, .txt), die von ARC/INFO exportiert wurden, werden nun in ein spezielles Verzeichnis kopiert. Dazu wird im GRASS-Datenbankverzeichnis unterhalb des *location* und *mapset*-Verzeichnisses ein Verzeichnis „arc“ angelegt. Die allgemeine Struktur ist:

`/home/<user>/grassdata/<location>/<mapset>/arc`

einfacher erreichbar durch: `$LOCATION/arc`

Dorthin werden die drei Exportdateien kopiert.
Nun können die Daten mit dem GRASS-modul `$ v.in.arc` eingelesen werden (folgende Beispiele sind in einer Zeile einzugeben!). Dabei sind die Spaltenzahlen für UNIX-ARC/INFO- und PC-ARC/INFO-Dateien unterschiedlich, da in der Attribut-Datei von PC-ARC/INFO die Spalte „$RECNO“ fehlt (hier wieder das Beispiel „topo12“, einzeilig einzugeben):

Bei Daten aus UNIX-ARC/INFO (in einer Zeile einzugeben):

```
$ v.in.arc type=polygon lines_in=topo12.pol points_in=topo12.lab
text_in=topo12.txt vector_out=topo12 idcol=5 catcol=5 attcol=6
```

Bei Daten aus DOS-ARC/INFO (in einer Zeile einzugeben):

```
$ v.in.arc type=polygon lines_in=topo12.pol points_in=topo12.lab
text_in=topo12.txt vector_out=topo12 idcol=4 catcol=4 attcol=5
```

Für ein „line coverage“ ist die Zeile entsprechend zu verändern. Das Modul `$ v.in.arc` kann auch interaktiv benutzt werden. Nach dem Einlesen ist
`$ v.support` auszuführen, um fehlende Informationen in die GRASS-Datenbank automatisch einzufügen:

```
$ v.support map=topo12 option=build
```

Mit dem Digitalisier-Modul `$ v.digit` kann die Datei betrachtet werden. Beim Ausfüllen des Informationsfelds ist darauf zu achten, dass als „map's scale“ der entsprechende Kartenmaßstab und nicht 1:0 angegeben wird.
Nun stehen die Vektor- und Punktdaten für weitere Arbeiten zur Verfügung.

A.4.3 IDRISI-Export nach GRASS über ARC/INFO

Das Programm IDRISI bietet die Möglichkeit, Vektordateien im ARC/INFO-Format zu speichern. Der vorangegangene Abschnitt zeigt, dass der Import von ARC-Daten kein Problem darstellt. Die zu überwindende Hürde ist allerdings die fehlende Attributdatei (Labeldatei), die IDRISI nicht exportieren kann. Diese benötigten Daten können mit ARC/INFO berechnet werden. Dazu müssen also die Vektoren zunächst aus IDRISI im ARC-Format gespeichert werden und in einer Datei im aktuellen Verzeichnis liegen. Dann wird ARC/INFO aufgerufen und ein neues „coverage“ erzeugt (Befehle für UNIX und DOS identisch) und die Datei importiert:

```
arc> generate testcover
generate> INPUT idrisiexport-datei
generate> line
generate> quit
arc> build testcover line
```

Das weitere Vorgehen unterscheidet sich je nach Betriebssystem.

(a) *UNIX:*
Die Attribute werden über das INFO-Modul extrahiert. Nun muss das INFO-Programm gestartet werden:

```
arc> info
```

Der nun abgefragte User heißt „arc“. Es ist die Groß- und Kleinschreibung zu beachten.

```
info> SELECT TESTCOVER.AAT
info> OUTPUT ../label.txt
info> LIST PRINT
info> Q STOP
```

Die Label-Datei, die für den GRASS-Import benötigt wird, liegt nun im aktuellen Verzeichnis.
(b) *DOS:*
Die PC-ARC-Version (3.x) besitzt kein INFO-Modul, die Daten werden im dBase-Format verwaltet. Daher verlässt man ARC/INFO und wechselt in das Verzeichnis, das mit dem coverage-Namen identisch ist. Dort wird dBase gestartet und mit

```
use aat.dbf
```

die Attributdatei geladen. Über „F2“ startet das „Assist-Modul“, und die Label können als Fremddatei (Abgrenzung der einzelnen Werte durch Leerzeichen) gespeichert werden. In diesem Verzeichnis liegt dann die Labeldatei.

A.4.4 Import in den ESRI-Formaten SHAPE und E00

Seit 1999 gibt es zwei neue Importmodule für Daten in ESRI-Formaten:

- v.in.shape
- m.in.e00

Das erste Modul liest SHAPE-Dateien ein, das zweite das ARC/INFO-E00-Format. Das E00-Format besitzt den Vorteil, die Vektortopologie beizubehalten, die dagegen im SHAPE-Format verloren geht. Besteht eine Wahlmöglichkeit vor einer Datenübernahme aus einem anderen GIS, ist somit das E00-Format dem SHAPE-Format vorzuziehen.

A.5 Koordinatenumrechnung mit m.proj und Kartentransformation mit r.proj/v.proj

Ab GRASS 5.0.x können Sie Koordinaten umrechnen lassen sowie ohne Passpunktsuche Raster- und Vektorkarten von einer Projektion in eine andere transformieren. Für die Koordinatenumrechnung steht ein „Koordinatentaschenrechner“ zur Verfügung, der zwischen 121 Projektionsarten umrechnen kann. Es können entweder einzelne Koordinatenpaare manuell umgerechnet werden oder auch eine Liste von Koordinaten, die in einer Datei abgelegt ist.

Umrechnung von Koordinaten

Im Folgenden wird nun die Einzelumrechnung am Beispiel der Umrechnung vom UTM-System in das Gauß-Krüger-System vorgestellt:

Zuerst ist (in GRASS) das Umrechnungsmodul („Koordinatentaschenrechner“) zu starten:

```
$ m.proj
```

Der erste Menüpunkt („Conversion“) gestattet nach der Definition der Quell- und Zielprojektion die Umrechnung einzelner Koordinatenwerte. Nun wird nach der „INPUT PROJECTION“ gefragt. Für das UTM-System ist `utm` einzugeben. Das „INPUT projection ELLIPSOID“ ist `wgs84`.
Als Zone ist für Deutschland `32` einzugeben. „Would you like to use South Hemisphere?“: `n` (no). Die Karteneinheit „INPUT units“ ist `meters`.
Nun folgt die Definition der Zielprojektion: „OUTPUT PROJECTION“ ist für das Gauß-Krüger-System `tmerc`. Das „OUTPUT projection ELLIPSOID“ heißt `bessel`.
Als „OUTPUT Central Parallel“ ist `0N` (Äquator) anzugeben, der „OUTPUT Central Meridian“ hängt von der Lage Ihres Projektgebiets ab. Für Hannover ist der Bezugsmeridian 9° Ost, also `9E`. Der „OUTPUT Scale Factor“ bleibt bei `1`. Das „OUTPUT False Easting“ ist wiederum vom Bezugsmeridian abhängig, für 9° Ost wird `3500000` eingegeben. Als „OUTPUT units“ gelten auch hier `meters`.
Nun fragt m.proj das erste umzurechnende Koordinatenpaar ab (hier: im UTM-System, Beispiel-Koordinatenpaar E: 32427882, N: 5833098). Als „Easting“ wird also eingegeben: `427882` (die führende Zonenangabe 32 weglassen!). Als „Northing“ wird hier dann eingegeben: `5833098`.
m.proj rechnet das angegebene Koordinatenpaar nun in das Gauß-Krüger-System um und zeigt an:

```
Universe Transverse Mercator -> Transverse Mercator  Conversion:
    X_in (meters)   Y_in (meters)   X_out (meters)  Y_out (meters)
    ------------    -----------     -------------   -------------
    427882.00      5833098.00      3427861.97      5834831.02
```

Auf diese Art und Weise können prinzipiell beliebige Umrechnungen stattfinden.

Eine Alternative zur manuellen Koordinatenpaareingabe ist die Verarbeitung von ASCII-Dateien mit Koordinatenpaaren. Hier wird zeilenweise Rechts- und Hochwert angegeben (in dieser Reihenfolge, bei UTM keine Zonenangabe im Rechtswert!) und dann m.proj aufgerufen. Eine interessante Anwendung ist die automatisierte Koordinatenumrechnung von GPS-Punkten, also von im Gelände aufgenommenen Standort- oder Flächengrenzdaten. Die Daten müssen hierzu von optionalen Attributen befreit werden. Das kann entweder in einem Tabellenkalkulationsprogramm erfolgen oder mit den UNIX-Textwerkzeugen „cut“ und „paste“ (vgl. Abschnitt A.8.1).
Rufen Sie nun m.proj auf. Mit Menüpunkt (2) „Input/Output Selection“ können Sie anschließend über (1) eine Datei als Eingabemedium (Option „File“) und über (2) eine neu zu erzeugende Datei, die die transformierten Koordinatenpaare enthalten soll, als Ausgabemedium anwählen (wenn Sie hier (2) nicht spezifizieren, erhalten Sie stattdessen die umgerechneten Werte aus Ihrer Eingabedatei auf dem Bildschirm). Mit (3) kehren Sie zum Hauptmenü von m.proj zurück. Dann wählen Sie (1) „Conversion“ und definieren Sie die Quell- und Zielprojektion wie oben beschrieben.
Eine UTM-Eingabedatei (Bezug: Zone 32, WGS84) könnte folgendermaßen aussehen:

```
427882 5833098
```

```
652226 5833098
652226 5620763
427882 5620763
```

Die Umrechnung z.B. in das Gauß-Krüger-System (Bezug: tmerc, bessel, 0N, 9E, Scale Factor:1, False Easting: 3500000) liefert dann:

```
427882.00      5833098.00      3427861.97      5834831.02
652226.00      5833098.00      3652268.28      5834831.01
652226.00      5620763.00      3652268.33      5622435.34
427882.00      5620763.00      3427861.94      5622435.36
```

Beachten Sie: Mit m.proj werden keine Daten, sondern nur Koordinatenpaare transformiert. Mit dem UNIX-Textwerkzeug „cut“ können Sie wieder ganz einfach die letzten beiden Spalten aus der Ergebnisdatei extrahieren.
Ein Hinweis zur Eingabe von Gradangaben (z.B. Längen-Breitengradsystem): Sie können entweder Koordinaten im Sexagesimalsystem mit Grad, Minuten und Sekunden durch Doppelpunkte getrennt und mit Nord-, Süd-, West- bzw. Ostangabe angeben (z.B. 52:35:00N und 9:20:22E) oder Koordinaten im Dezimalgradsystem (Nord- und Ostwerte positiv, Süd- und Westwerte negativ) mit Punkt als amerikanischem Komma (z.B. -9.5 statt 9:30W). Ein „Gemisch“ wird von m.proj nicht akzeptiert.

Automatisierte Transformation von Raster- oder Vektordaten

Für eine Transformation von Raster- oder Vektordaten sind neben `$ i.rectify` (mit Passpunkten) die Module `$ v.proj` und `$ r.proj` (ohne Passpunkte, basierend auf den Projektionsformeln) vorhanden. Damit können Sie automatisiert Raster- und Vektorkarten in andere Projektionen transformieren.
Derzeit gibt es eine umgehbare Einschränkung: `$ v.proj` und `$ r.proj` arbeiten derzeit nur für die Transformation von gesamten *locations*. Selbstverständlich geben Sie an, welche Karte umgerechnet werden soll. Kartenausschnitte können jedoch (noch) nicht transformiert werden. Das bedeutet für eine erfolgreiche Rechnung, dass die Quell-*location* maximal so groß sein darf wie die Ziel-*location*, damit die neuen Randkoordinaten korrekt berechnet werden. Sie können also nicht aus einem Europadatensatz nur ein Land in eine *location* mit neuer Projektion bringen, sondern nur Gesamteuropa in eine neue *location*, die mindestens (in der neuen Projektion) so groß wie Europa ist. Als Ausweg ist aber möglich, das gewünschte Land in der alten Projektion auszuschneiden, zu exportieren und in einer neuen, kleineren *location* mit alter Projektion wieder zu importieren (die neuen Randkoordinaten können Sie ja ganz einfach ablesen, da ja noch nichts zu transformieren ist). Diese ausgeschnittene Karte lässt sich unter Voraussetzung, dass sie in die *location* mit neuer Projektion passt, transformieren.

Ein Beispiel: Sie haben die geologische Karte („geologie“) der Insel Naxos in Längen-/Breitengrad-koordinaten (*location:* „naxosll“, *mapset:* „naxosll“) vorliegen, möchten sie aber im UTM-System weiterbearbeiten. Dann rechnen Sie die Längen-/Breitengrad-Randkoordinaten der Längen-/Breitengrad-*location* „naxosll“ mit dem Modul `$ m.proj` in das UTM-System um (s.o.). Mit diesen vier berechneten UTM-Randkoordinaten (Naxos ist in UTM-Zone 34) erzeugen Sie nun eine neue UTM-*location* „naxosutm“ mit gewünschter Auflösung. Anschließend kann die geologische Karte „herübergeholt“ werden unter gleichzeitiger Koordinatentransformation. Sie befinden sich also mit GRASS in der UTM-*location* und geben ein:

```
$ r.proj in=geologie out=geologie location=naxosll mapset=naxosll
```

Nach einiger Rechenzeit liegt die Karte transformiert im UTM-System vor.
Vergleichbar läuft das Verfahren mit Vektorkarte und `$ v.proj` ab.

A.6 Definition von Postscript-Treibern in GRASS

Das Erstellen von Postscript-devices (DIN A0 – A4) ist recht einfach, es müssen nur die entsprechenden Blattgrößen und die Auflösung definiert werden. GRASS erzeugt mit dem ps.map-Befehl Postscript-Karten, die dann wie üblich unter UNIX gedruckt werden können.
Hier werden nun Definitionen für fünf verschiedene Blattgrößen gegeben. Es folgen 5 Dateien mit Formatangaben in der Reihenfolge a0, a1, a2, a3 und a4. Diese Dateien müssen, sofern noch nicht vorhanden, in:

```
/usr/local/grass42/etc/paint/ps.devices/
```

bzw.

```
/usr/local/grass5/etc/paint/ps.devices/
```

erzeugt werden (Inhalt: zwischen den Strichen). Das Verzeichnis `../ps.devices` muss eventuell hierfür angelegt werden. Die erste Zeile in der anzulegenden Datei ist die „level“-Zeile, der entsprechende Dateiname steht darüber über dem horizontalen Strich:

```
printer-a0:
 ----------------------
  level: 2
  page width: 33.07
  page height: 46.77
  top margin: 0.5
  bottom margin: 0.5
  left margin: 0.25
  right margin: 0.25
  resolution: 300
 ----------------------
```

```
printer-a1:
 ----------------------
  level: 2
  page width: 23.39
  page height: 33.07
  top margin: 0.5
  bottom margin: 0.5
  left margin: 0.25
  right margin: 0.25
  resolution: 300
 ----------------------
```

```
printer-a2:
 ----------------------
  level: 2
  page width: 16.54
  page height: 23.39
  top margin: 0.5
  bottom margin: 0.5
  left margin: 0.25
  right margin: 0.25
  resolution: 300
 ----------------------
```

```
clc-a3:
 ----------------------
  level: 2
  page width: 11.69
```

```
clc-a4:
 ----------------------
  level: 2
  page width: 8.27
```

```
  page height: 16.54                page height: 11.69
  top margin: 0.5                   top margin: 0.5
  bottom margin: 0.5                bottom margin: 0.5
  left margin: 0.25                 left margin: 0.25
  right margin: 0.25                right margin: 0.25
  resolution: 300                   resolution: 300
 ----------------------            ----------------------
```

Die Auflösung (resolution) kann je nach Drucker unterschiedlich gewählt werden.

A.7 Steuerungsdatei für ps.map: Beispiel „Moordaten.psmap“

Diese Datei ist eine beispielhafte Steuerungsdatei für die Kartengestaltung mit ps.map. Sie gibt eine Übersicht über die möglichen Steuerungsbefehle.

```
raster <Rasterdatei>         # z.B. tk25-rast
vector <Vektordatei>         # z.B. moorstrassen-vec
  color 1 <Farbe>            # Farbe des Labels 1: z.B. red
  color 2 <Farbe>            # Farbe des Labels 2: z.B. green
  width 1                    # Breite der Linien: hier z.B. 1 Pixel
  hcolor <Farbe>             # highlight-Farbe  - nicht notwendig
  hwidth <Farbe>             #                  - nicht notwendig
  masked y                   # Ausblendung (siehe r.mask), n fuer
                             #             "ueberall Zeichnen"
  style 1                    # 1-9 Stellen, Strichelchen etc.:
                             #          z.B. 0011 erzeugt --  --  --
                             # ohne style erhaelt man eine
                             # durchgehende Linie
end
region <Regiondatei>         # mit g.regions (nur Ausschnitt zeigen)
  color <Farbe>
  width 1
end
grid 10000                   # Gitter mit Maschenweite 10000m
  color <Farbe>              # Farbe: z.B. black
  numbers 10 <Farbe>         # jede 10. Zeile und Spalte des Gitters
                             #     ist numeriert in <Farbe>
  font fontname style        # z.B. Helvetica, Helvetica bold etc.
  fontsize fontname size     #
end
outline                      #  Umrahmung der Rasterdatei (nur,
                             #  wenn vorhanden)
  color <Farbe>
end
colortable y                 # Farbtafel erstellen als Legende (nur,
```

```
                              #     wenn Rasterdatei vorhanden)
comments <Kommentardatei>     # z.B. moorstrassen.comments
  where x y                   # Platzierung in inch vom linken Rand u. ob.
  font fontname style         # z.B. Helvetica, Helvetica bold etc.
  fontsize fontname size      # z.B. Helvetica 12
  color <Farbe>               # Farbe: z.B. orange
end
scale 1:25000                 # Massstab
end
setcolor 6,8,9 <Farbe>        # Bei Rasterdaten 6,8,9te Farbe auf <Farbe>
                              #    setzen (Farbtabelle aendern)
setcolor 10 <Farbe>           # ...darf mehrfach vorkommen!
vlegend                       # Druckt scale, grid und region-
                              #     Informationen zu Vektordaten
  where x y                   # Platzierung: z. B. where 5 2 - fuenf inch
                              #         von links, 2 von oben
end
legend <east><north>          # Platzierung der benutzerdef. Legende
  height 20                   # Farbblock f. Rasterb.: Hoehe (in Pixel)
  width 20                    # Farbblock f. Rasterb.: Breite (in Pixel)
  vlen 20                     # Strich f. Vektordaten: Laenge  (in Pixel)
  textcolor <Farbe>           #
  textsize                    # wird in geographischen Einheiten angegeben:
                              #        (m oder km)
  textwidth                   # in Pixeln
  xspace                      # Platz zw. Legendensymbol und Text (horz.)
  yspace                      # Platz zw. Legendensymbolen (vertikal)
  background <Farbe>          #
  border <Farbe>              #
  beginrast
    ramp label vertical       # "vertical" oder "horizontal", eine
                              #      Farbrampe auf labels bezogen
  ODER!!:
    catnum 1 Sandstein        # categories zugeordnet
    catnum 3 Kalkstein        # etc.
  end
  beginvect
    vectname <Datei> Bezeichnung   # Vektordaten integrieren in Legende
   end
  beginsite
    sitename  <Datei> Bezeichnung  # Sitesdaten integrieren in Legende
  end
end
text <east> <north> <blabla># Positioniert <blabla> nach Gauss-Krueger,
```

```
                            #     z.B. "Moorstrassenkarte"
  color <Farbe>
  width 1                   # Strichbreite in Pixel
  hcolor <Farbe>            # highlight-Farbe  - nicht notwendig
  hwidth 1                  #                  - nicht notwendig
  background <Farbe>        # Texthintergrund
  border <Farbe>            # Textrahmen
size 500                    # in METERN (Textgroesse veraendert sich
                            # mit Massstab!)
  ref  center               # Textlage zum Referenzpunkt
  opaque n                  # Festlegung, ob Vektor uebermalen darf
end
line <east> <north> <east> <north>  # Linie von Punkt zu Punkt
  color <Farbe>
  width                     # Breite in Pixel
  opaque n                  # Festlegung, ob Vektor uebermalen darf
end
point                       # Punkte oder icons (ps.icons) platzieren
  color <Farbe>
  icon diamond              # z.B. diamond
  size 15                   # in Punkt (wie Schriftgroesse)
  masked n                  # nur in Ausschnitt setzbar oder ueberall
end
barscale <east> <north>     # Platzierung der Massstabsleiste
   unit  km                 # km oder m - Unterteilung
   length 2                 # z.B. 2 km, in Einheiten der Unterteilung
   interval 1               # Ein Strich  pro Einheit(hier: km)
   style tick               # (a) dash oder (b) tick:
                            # (a) wie Eisenbahnsymbol, (b) Dauerlinie
                            #                   mit Intervallstrichen
   width 10                 # in Pixel, Leistenbreite
   color  <Farbe>
   textsize                 # in Einheiten (m oder km), wird mitskaliert
   textcolor  <Farbe>
   font  Helvetica
   background  <Farbe>
   border  <Farbe>
end
labels <Labeldatei>        # mit p.labels anlegen:
                           #         z.B. moorstrassen.labels
```

A.8 Allgemeine Hinweise

A.8.1 Benutzung der UNIX-Textwerkzeuge für GIS-Datenaufbereitung

Die UNIX-Textwerkzeuge „cat“, „cut“, „join“, „head“, „more“, „paste“, „sed“, „tail“ (in Ergänzung mit „awk“) bieten vielfältige Möglichkeiten zur einfachen Bearbeitung von Textdateien, insbesondere Texttabellen. Da Attributtabellen für GIS-Daten häufig in solchen Texttabellen gespeichert sind, soll hier die Bearbeitung einer Tabelle vorgestellt werden. Prinzipiell können die UNIX-Textwerkzeuge sogar annähernd Tabellenkalkulationsprogramme ersetzen.
In diesem Beispiel geht es um die Aufarbeitung der auf dem „GRASS Europe“-Server (Universität Hannover) angebotenen Legende zur SPEARFISH-Bodenkarte. Den SPEARFISH-Datensatz erhalten Sie ebenfalls dort. Die Legendendatei ist eine ASCII-Datei, die für die in der Bodenkarte enthaltenen Polygone weitere Attribute beinhaltet. Im Folgenden wird gezeigt, wie sich die Legende auf einfache Weise modifizieren lässt. Das Ergebnis ist eine Textdatei, die als Klassifizierungsregel-Datei für die Reklassifizierung der Bodenkarte dienen kann.
Zunächst soll der Legendeninhalt betrachtet werden. Die Anzeige einer Textdatei, hier also der Legende zur Bodenkarte erfolgt mit:

```
$ more soils_legend.txt
```

(in „more“: Blättern mit „Leertaste“, Verlassen mit „q“ (quit), Suchen mit „/“)

```
0:no data:
1:AaB:Alice fine sandy loam, 0 to 6
2:Ba:Barnum silt loam
3:Bb:Barnum silt loam, channeled
4:BcB:Boneek silt loam, 2 to 6
5:BcC:Boneek silt loam, 6 to 9
6:BeE:Butche stony loam, 6 to 50
7:BhE:Butche rock outcrop complex, 25 to 50
8:BkD:Butche Satanta loams, 6 to 25
9:CBE:Citadel association, hilly
[...]
```

Die Legende ist folgendermaßen aufgebaut: In der ersten Spalte steht die Polygonnummer, in der zweiten das Kürzel für die Bodenserie, in der dritten der ausgeschriebene Name mit Bodenart-Attribut, dann durch Komma getrennt die typische Hangneigung.
In einem ersten Schritt soll eine Reduzierung der Legende auf Nummer, Kürzel und Textattribut (ohne Hangneigungen) erfolgen, die Hangneigungen können statt dessen exakter aus dem SPEARFISH-Höhenmodell abgeleitet werden (cut schneidet Spalten aus einer Textdatei, Speicherung des Ergebnisses in neuer Datei):

```
$ cut -d',' -f1 soils_legend.txt > soils_legend2.txt
```

(d: delimiter=Komma, Anzeige von f: field=erstes Feld)

Die neue Datei können Sie wieder mit „more“ betrachten:

```
0:no data:
1:AaB:Alice fine sandy loam
2:Ba:Barnum silt loam
3:Bb:Barnum silt loam
4:BcB:Boneek silt loam
5:BcC:Boneek silt loam
6:BeE:Butche stony loam
7:BhE:Butche rock outcrop complex
8:BkD:Butche Satanta loams
9:CBE:Citadel association
[...]
```

Nun soll eine Selektion ausschließlich der Textattribute in dieser neu erzeugten Datei erfolgen (das Ergebnis von „cut“ können Sie dafür mit „¿“ in eine neue Datei umlenken, ansonsten wird es, wie Sie gesehen haben, auf dem Bildschirm ausgegeben):

```
$ cut -d':'  -f3 soils_legend2.txt

Alice fine sandy loam
Barnum silt loam
Barnum silt loam
Boneek silt loam
Boneek silt loam
Butche stony loam
Butche rock outcrop complex
Butche Satanta loams
[...]
```

Beachten Sie, dass die erste Zeile leer bleibt, da dass „no data“-Feld kein Textattribut hat. Alternativ können Sie die beiden vorangegangenen Schritte über „UNIX-Piping“ auch in einem Schritt durchführen, ohne eine neue Textdatei anlegen zu müssen:

```
$ cut -d',' -f1 soils_legend.txt |cut -d':'  -f3
```

Statt über Trennzeichen (beliebiges Zeichen) könne Sie auch an einer bestimmten Stelle ausschneiden. Dann ist/sind die Stelle(n) anzugeben (entsprechend auszuzählen):

```
$ cut -b1,2 soils_legend.txt

0:n
1:A
2:B
3:B
```

```
4:B
5:B
6:B
7:B
8:B
9:C
[...]
```

Sollen nur die ersten 4 Zeilen angezeigt werden, benutzen Sie ein anderes Kommando:

```
$ head -4 soils_legend.txt
```

```
0:no data:
1:AaB:Alice fine sandy loam, 0 to 6
2:Ba:Barnum silt loam
3:Bb:Barnum silt loam, channeled
```

Möchten Sie stattdessen nur die letzten 3 Zeilen anzeigen lassen:

```
$ tail -3 soils_legend.txt
```

```
53:WaA:Weber loam, 0 to 2
54:Wb:Winetti cobbly loam
55:water
```

Das kann auch kombiniert werden: Anzeige nur der 3.-5. Zeile (wieder unter Benutzung von „UNIX-Piping“):

```
$ head -5 soils_legend.txt | tail -3
```

```
2:Ba:Barnum silt loam
3:Bb:Barnum silt loam, channeled
4:BcB:Boneek silt loam, 2 to 6
```

Die Zeile 0 gilt bei UNIX immer für die Zählung. Wie gehabt können Sie das Ergebnis in eine neue Datei umlenken:

```
$ head -5 soils_legend.txt | tail -3 > legende_neu.txt
```

Wollen Sie mehrere Dateien aneinanderhängen, benutzen Sie „cat“:

```
$ cat teillegende1.txt teillegende2.txt > legende_komplett.txt
```

Das „paste“-Kommando erlaubt das horizontale Aneinanderfügen von Texten (spaltenweise), das „join“-Kommando sogar mittels Zuordnung über gemeinsame Spalteneinträge. „sed“ ersetzt Zeichenketten (Begriffe) durch andere, mit „awk“ können Sie mathematische Rechnungen programmieren, wenn Sie eine Zahlentabelle „hineinleiten“. Diese Werkzeuge sind sehr mächtig, unter Verwendung von „UNIX-Piping“ lassen sich komplexe Programme erzeugen.
Mit „man ¡kommando¿“ können Sie sich die Anleitungen zu oben genannten Werkzeugen ansehen.

A.8.2 Typische Farbwerte für topographische Karten

Die folgenden Werte eignen sich für eine manuelle Farbzuweisung, die bei der Klassifikation von Satellitenbildern nötig ist. Mit den folgenden Werten können den einzelnen Klassen Farbwerte zugewiesen werden, die mit den digital lieferbaren topographischen Karten (z.B. die CD-ROM vom Landesvermessungsamt Niedersachsen) übereinstimmen.
In GRASS werden diese Farben mit `$ d.colors` oder `$ r.colors` zugewiesen.

TK25	**R**	**G**	**B**
0 - no data	-	-	-
1 - weiß	255	255	255
2 - Wald, grün	184	240	153
3 - Wasserfläche, hellblau	192	230	255
4 - Höhenlinien, braun	179	102	26
5 - Gräben etc., dunkelblau	0	51	255
6 - Grundriss, schwarz	0	0	0
7 - Schrift, dunkelblau	0	0	255
8 - Baumsignatur, grün	0	192	0
9 - dunkelgrau	192	192	192
10 - hellgrau	217	217	217
11 - Grundriss, Schrift, schwarz	0	0	0

Mit `$ r.reclass` lassen sich auch einzelne Signaturen „herauspräparieren“, indem man sich in einer digital gelieferten topographischen Karte nur den entsprechenden Farbwert herausgreift. Auf die Farbwerte einer Rasterdatei kann zugegriffen werden, da sie als „categories“ abgelegt sind. So ist die Erzeugung einer Raster-Höhenlinienkarte möglich, indem alle Farben außer dieser (meistens Braun) auf Null gesetzt werden. Im Anschluss kann dann diese Karte mit `$ r.thin` und `$ r.line` vektorisiert werden. Gescannte Karten lassen sich auf diese Weise auch wieder auf ihre wenigen Standardfarben reduzieren. Vereinfacht können alternativ Farbbereiche mit `$ r.rescale` auf einen Wert gesetzt werden.

A.8.3 Informationen zu LANDSAT-TM-Satellitendaten

Bedeutung der LANDSAT-TM-Kanäle

Die Bedeutung der sieben Kanäle gliedert sich wie folgt (nach SEEL 1995 und sowie HILDEBRANDT 1992, erweitert nach BIRKNER in BUZIEK 1995):

1 Blau (0.45-0.52 μm, 30m Auflösung):

Für Studien in Küstenbereichen geeignet, da tiefes Eindringen in Wasser. Weiterhin Unterscheidung von bewachsenem und unbewachsenem Boden sowie Laub- und Nadelbäumen. Starke Störungen durch atmosphärisches Streulicht möglich.

2 Grün (0.52-0.60 μm, 30m Auflösung):

Zeigt Vitalität von Pflanzen für bodenkundliche Untersuchungen. Ratio tm2/tm4 zur Kartierung limonithaltiger Gesteine und Rottönung von Wüstensanden verwendbar.

3 Rot (0.63-0.69 μm, 30m Auflösung):

Gut geeignet zur Darstellung von Straßen und unbedecktem Boden. Außerdem verwendbar für die Untersuchung eisenhaltiger Gesteine sowie von Strukturen.

4 Nahes Infrarot NIR (0.76-0.90 μm, 30m Auflösung):

Zur Schätzung des Biomassenanteils geeignet. Unterscheidung von Wasserkörper und Vegetation.

5 Kurzwelliges Infrarot SWIR (1.55-1.75 μm, 30m Auflösung):

Zeigt den Wassergehalt von Pflanzen und Böden. Ermöglicht die Unterscheidung von Wolken und Schnee. Zeigt Straßen, unbedeckten Boden, Wasser. Guter Kontrast bei unterschiedlichen Vegetationstypen, keine Dunstbeeinflussung.

Am besten geeignet für geologische Untersuchungen: Ratio tm4/tm5 erlaubt Trennung von wasser- und eisenhaltigen Gesteinen, Ratio tm4/tm7 die Unterscheidung von Tonmineralen.

6 Thermisches Infrarot TIR (10.5-12.5 μm, 120m Auflösung):

Messung von Wärmestrahlung (auch nachts). Ermittlung von Stress in der Vegetation, zeigt unterschiedliche Bodenfeuchte an. Ermöglicht Tiefenmessungen in Seen sowie die Unterscheidung siliziumreicher Gesteine. Geringere Auflösung.

7 Kurzwelliges Infrarot SWIR (2.08-2.35 μm, 30m Auflösung):

Zeigt unbedeckten Boden, geeignet zur Vegetationsdifferenzierung (schlechter als Kanal 5). Besondere geologische Möglichkeiten: Absorptionsbänder für Schichtsilikate und Karbonate in diesem Kanal.

Farbkomposit-Übersicht

Die hier aufgeführte Übersicht der Bedeutung verschiedener Farbkomposite zeigt eine Vielfalt an Auswertungsmöglichkeiten mit LANDSAT-TM-Szenen (nach SEEL 1995, weitere Informationen vgl. LÖFFLER 1994). Zu überlegen wäre, für manche Kompositbildungen vorher eine Hauptachsentransformation durchzuführen (insbesondere bei dem Echtfarbkomposit).

Die Farbkomposite sind in der Reihenfolge blau, grün, rot zu erstellen, d.h. B = erster Kanal, G = zweiter Kanal, R = dritter Kanal. Eventuell sollte vorher eine Hauptachsentransformation mit `$ i.pca` durchgeführt werden.

123: Echtfarbbild, da die ersten drei Bänder den sichtbaren Bereich umfassen.

234: Empfindlich für grüne Vegetation (in Rot dargestellt), Nadelhölzer haben dunkleres Rot als Laubbäume. Straßen und Wasserkörper gut erkennbar.

243: Grüne Vegetation erscheint grün, aber Nadelwälder sind nicht so klar wie in 234.

247: Beste Kombination fur Waldbeurteilung. Gut geeignet zur Kartierung von Ernteflächen und Straßen.

345: Enthält die Hauptreflexionskanäle (VIS, NIR, SWIR). Grüne Vegetation erscheint grün, das SWIR zeigt Vegetationsstress, Straßen sind schlechter erkennbar. item[347:] Ähnlich wie 345, aber Darstellung verbrannter Flächen besser geeignet.

354: Wirkt wie ein Farbinfrarotbild.

374: Ähnlich wie 354.

457: Zeigt Bodentexturklassen (Ton, Schluff, Sand).

Verhältniskarten (Ratiokarten) aus LANDSAT-TM-Daten

Eine Alternativmethode zur Bildung von Farbkomposits zur Untersuchung spektraler Eigenschaften von Oberflächen ist die Ratiobildung. Dabei werden die Rasterzellenwerte zweier Farbkanäle mathematisch geteilt. Insbesondere bei geologischen bzw. bodenkundlichen Auswertungen können Mineralien der Erdoberfläche unterschieden werden, sofern keine Vegetation vorhanden ist (nach ENVI 3.0 TUTORIAL 1997, S. 287):

TM5/TM4: Tonminerale, Karbonat, Vegetation

TM3/TM1: Eisenoxid

TM2/TM4: Vegetation

TM3/TM4: Vegetation

TM5/TM4: Vegetation

Diese Formeln können Sie direkt in r.mapcalc eingeben (vgl. Abschnitt 11.5).
Es kann auch eine Kombination von Ratiobildung und Farbkomposite interessant sein (zuerst Berechnung in r.mapcalc, dann Kombination in i.composite):

TM5/TM7 $\rightsquigarrow$R

TM3/TM1 $\rightsquigarrow$G

TM2/TM4 $\rightsquigarrow$B

Hier werden Tonminerale und Karbonat in der Farbe Magenta erscheinen, Eisenoxide Grün sowie Vegetation in Rot.

B GNU Free Documentation License

Version 1.1, March 2000

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".
A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's

overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.
If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.
If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.
It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

- State on the Title page the name of the publisher of the Modified Version, as the publisher.
- Preserve all the copyright notices of the Document.
- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections entitled “History” in the various original documents, forming one section entitled “History”; likewise combine any sections entitled “Acknowledgements”, and any sections entitled “Dedications”. You must delete all sections entitled “Endorsements.”

Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ör any later versionäpplies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify

a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

GRASS Handbuch

Das in diesem Leitfaden vorgestellte Geographische Informationssystem GRASS (Geographical Resources Analysis Support System) ermöglicht die Bearbeitung von Rasterkarten, topologischen Vektor- und Fernerkundungsdaten. GRASS ist Freie Software unter der GNU General Public License (GPL).

Dieses Handbuch richtet sich sowohl an GIS-Erfahrene, die GRASS neu kennenlernen möchten, als auch an GIS-Anfänger. Daher sind der eigentlichen Beschreibung von GRASS ein Abschnitt über Geographische Informationssysteme im Hinblick auf GRASS und eine ausführliche Anleitung zum Thema GNU/Linux vorangestellt. Ein Schwerpunkt liegt auf der Datenintegration, da hier erfahrungsgemäß die meisten Fragen beim Umgang mit GIS auftreten. Einige Beispiele von GIS-Applikationen sollen Anregung für eigene Projekte geben. Das Buch bezieht sich hauptsächlich auf GRASS 4.x, gibt aber relevante Hinweise für das Arbeiten mit GRASS 5.0.

GRASS ist auf vielen Computer-Plattformen wie GNU/Linux, MS-Windows, Mac OSX und anderen Systemen einsetzbar und hat eine weltweite Nutzer- und Entwicklergemeinde.

GDF HANNOVER
Gesellschaft für Datenanalyse und Fernerkundung bR

Intevation GmbH

VERSION 5
GRASS HANDBUCH