Frida: Freie Vektor-Geodaten Osnabrück

:: Version 0.9::

Projektdokumentation

Monika Schunke

22. November 2002

Inhaltsverzeichnis

1 Einleitung

1.1 Hintergrund

Freie Vektor-Geodaten sind ausser für die USA nahezu nicht verfügbar. Lediglich einige globale Datensätze geringerer Auflösung sind entstanden (z.B. VMAPL0 und GSHHS). Alle Staaten ausser den USA geben die behördlich erhobenen Daten nicht frei und erlegen weitreichende Restriktionen bei der Weitergabe an Dritte auf.
Bei Freien Daten sind die Freiheiten analog zu Freier Software zu verstehen. Dies sind:

1. Die Freiheit, ein Programm für jeden Zweck einsetzen zu dürfen
2. Die Freiheit, untersuchen zu dürfen, wie ein Programm funktioniert, und es den eigenen Bedürfnissen anzupassen
3. Die Freiheit, Kopien für Andere machen zu dürfen
4. Die Freiheit, das Programm verbessern zu dürfen und diese Verbesserungen zum allgemeinen Wohl zugänglich zu machen

Die restriktive Handhabung jedoch verursacht eine Reihe von immensen Probleme. Eines ist die Schwierigkeit, Freie GIS Software mit interessanten Beispiel-Daten zu verbreiten. Bisher werden grundsätzlich US-Daten verwendet, für den europäischen Bereich wären entsprechende Daten wesentlich attraktiver.
Der politische Prozeß zu einer Freigabe der Geo-Daten ist in Deutschland und Europa noch nicht einmal begonnen worden. Zur Zeit wird lediglich die Verfügbarmachung, aber eben weiterhin unter den harten Restriktionen, sowie Vereinheitlichung diskutiert.

1.2 Ziele

Hauptziel war es, für eine europäische Stadt Vektor-Geodaten zu erzeugen und als Freie Geodaten allen Interessierten zur Verfügung zu stellen.
Die Wahl viel auf Osnabrück, da dies der Sitz von Intevation ist und die Stadt Osnabrück die Idee begrüsst und aktiv unterstützt. Allerdings wurde dabei sehr genau darauf geachtet, dass keine Daten die bereits Restriktionen unterliegen, von der Stadt beigesteuert werden.
Mit der Erstellung des Datensatzes werden konkret folgende Punkte anvisiert:

- Ein detaillierte Datensatz als Beispieldaten für Freie GIS Software. Die Software kann dann direkt mit den Daten ausgeliefert und somit ausprobiert werden.
- Verwendung bei kommerziellen Produkt-Demonstrationen. Unternehmen, die Dienstleistungen mit Freier GIS Software anbieten können mit den Osnabrücker Daten interessante Demos bauen und den Kunden problemlos zur Verfügung stellen.
- Test der Datenpflege: Wenn eine gute Heimatseite zu den Daten erstellt wird und Möglichkeiten zu Fehlerberichten und Mitarbeit geschaffen wird, kann der Datensatz dann längerfristig aktuell und attraktiv gehalten werden?
- Vorbild für weiter Projekte zur Erstellung Freier Vektor-Geodaten.
- Ausübung politischen Drucks zum Anstoss einer politischen Diskussion ob Freie Geodaten für Europa nicht von größerem vorbild als propritäre Daten sind.

1.3 Umsetzung

Das Projekt wurde im Rahmen eines Praktikums bei der Intevation GmbH durchgeführt.
Als Grundlage für die freien Vektordaten werden Orthofotos der Stadt Osnabrück verwendet, welche digitalisiert und attributiert werden sollen. Besonderer Wert wird dabei auf die (soweit möglich) vollständige Digitalisierung der Strassen der Stadt Osnabrück gelegt. Desweiteren sollten Öffentliche Gebäude und andere wichtige Objekte erfasst werden. Die digitalisierten Daten sollen dann weiterhin sinnvoll attributiert und unter der GNU GPL Lizenz der Öffentlichkeit zur Verfügung gestellt werden. Das Projekt soll in einem Zeitraum von 18 Wochen realisiert und die Daten später ständig aktualisiert werden.

1.4 Verwendete Programme

Bei den verwendeten Programme mit denen die Daten gewonnen und verarbeitet werden, handelt es sich ausschließlich um Freie Software.

- Um die Möglichkeiten im GIS Bereich aufzuzeigen wurde zur Digitalisierung der Daten das Programm GRASS verwendet, welches ein weitverbreitetes Programm zur Ver- und Bearbeitung von Raster- und Vektordaten ist.

  In GRASS ist es derzeit nur möglich ein Attribut und einen dazugehörigen Attributnamen zu vergeben, deswegen muß die Attributierung mit einem anderen Programm durchgeführt werden.

- Es wurde entschieden, für die Attributierung Thuban zu verwenden. Thuban ist ein neuentwickelter Geodatenbetrachter, welcher in seiner aktuellen Version ausreichende Funktionalität bietet um die Daten,bzw. die Datenbankdatei zu bearbeiten. Verwendet wurde die CVS Version vom 28.08.02, welche noch zusätzliche Funktionen zu der offiziellen Version 0.1.2 enthält.

- Die Basis GIS Anwendung wurde mit UMN Mapserver aufgesetzt. Ziel ist es Daten zu gewinnen die jedem zugänglich gemacht werden können und nach der GNU GPL frei verändert und bearbeitet werden können.

Diese Programme sind alle auf der FreeGIS CD enthalten.

1.5 Datengrundlage

Es werden digitale Orthofotos im .jpg Format mit zugehörigen Worldfile .jpgw verwendet, welche im Maßstab 1:5000 vorliegen. Die Auflösung liegt bei 16 cm und die zugrundeliegenden Luftbilder wurden im Jahr 1999 gemacht. Die Orthofotos wurden uns für den Zeitraum der Digitalisierung von der Stadt Osnabrück zur Verfügung gestellt.

2 Digitalisierung mit GRASS

Die Aufgabe in GRASS ist es, Vektordaten durch Digitalisierung ausgehend von digitalen Orthofotos zu gewinnen. Diese Orthophotos liegen in digitaler Form vor. Für die Durchführung werden georeferenzierte jpg formate mit entsprechendem Worldfile(jpgw) zur Verfügung gestellt.
Ein prinzipielles Wissen darüber wie man ein Projekt in GRASS anlegt, wird an dieser Stelle vorausgesetzt, einführende Tutorials dazu existieren bereits, siehe FreeGIS Tutorial Version 1.0 von Heiko

Kehlenbrink Kapitel 2 und im GRASS Buch OPEN SOURCE GIS: A GRASS GIS Approach von Neteler/Mitasova.
Es wurde mit der GRASS VERSION 5.0.0pre 5 digitalisiert. Bei dem Export der Daten wurde auf die GRASS VERSION 5.0.0 umgestiegen, da Probleme bei dem Export von Polygonen bestanden.
Unterschiedliche Themen werden als eigene Layer erfasst, d.h. Strassen, Grünflächen, Gewässer und Sehenswürdigleiten werden jeweils seperat digitalisiert.

2.1 Import der Daten

Da anfängliche Schwierigkeiten mit dem jpg Format existierten, wurde entschieden die jpg's in tiff Format umzuwandeln. Dies geschieht mit Hilfe des Paketes GDAL (Sammlung von Bibliotheken), welches auf dem Rechner installiert sein muss (es ist enthalten auf der FreeGIS CD). Das Programm gdal_translate muß ausgeführt werden. Dies fasst man am besten in einem Script zusammen. Dieses Script ist in der CVS Version unter Processing zu finden.

```
Dateiname: uebersetzung

set -x #(um die Vorgänge sichtbar zu machen)
for f in ‘ls luftbilder\_split/*.jpg‘; do
        gdal\_translate \$f tiff/‘basename \$f .jpg‘.tif
done
```

Danach müssen die jpgw's in tfw umgewandelt werden, dies geschieht mit einem einfachem cp Befehl.

```
cp 3494.jpgw 3494.tfw
```

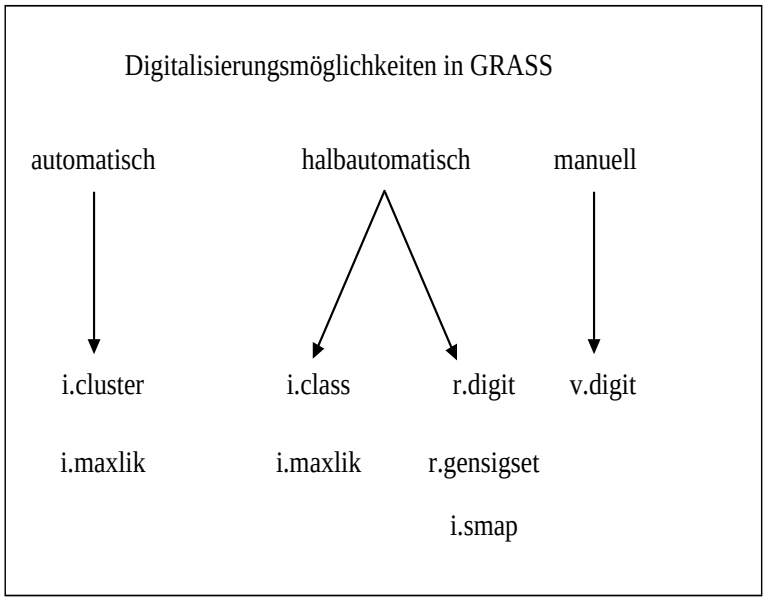
Man muß darauf achten, dass die tif und tfw Dateien zum Schluß in einem Verzeichnis stehen. In GRASS werden nur die .tif Datei angegeben und GRASS sucht dann automatisch im selben Verzeichnis, indem die .tif Dateien stehen, die jeweils zugehörige .tfw Datei.

Der Import der Daten erfolgt mit dem Modul **r.in.gdal**, dabei muß man darauf achten die -o Flagge anzugeben, da GRASS die aktuelle Projektion überschreiben muß. Das Orthofoto wird nun in den 3 Bändern Rot, Grün und Blau eingelesen. Damit das Bild auch in Echtfarben dargestellt werden kann, müssen die Werte in den einzelnen Farbtabellen der Bänder auf Grau gesetzt werden. Dies geschieht mit dem Modul **r.colors** für jedes einzelne Farbband. Zuvor sollte man die Region mit **g.region** auf eines der drei Bänder setzen. Nun kann man sich das Bild mit **d.rgb** auf dem Monitor, welcher zuvor mit **d.mon** gestartet wurde, ausgeben lassen. Wenn man mit nur einem Bild weiterarbeiten möchte, muß man die einzelnen Farbbänder vereinen. Dies ist für die gewählte manuelle Digitalisierung zwingend erforderlich. Um die Bilder zu vereinen muß das Modul **r.composite** ausgeführt werden. Danach kann man sich das Bild mit **d.rast** anzeigen lassen und die einzelnen Bänder mit **r.remove** löschen.
Diese einzelnen Schritte kann man wiederum sehr gut in einem Skript zusammenfassen und ausführen. Das Script ist in der CVS Version unter processing zu finden. Bei dem untenstehenden Beispiel ist es nur noch erforderlich den Dateinamen und den Namen des Luftbildes auf der Kommandozeile einzugeben, z.Bsp. luftimport 3494 :

```
Dateiname: luftimport
Ausführung: luftimport 3494

set -x
r.in.gdal -o input=/home/monika/os-data/raster/aerial-photos/tiff/\$1.tif
output=r\$1

g.region raster=r\$1.1
r.colors r$1.1 color=grey
r.colors r$1.2 color=grey
r.colors r$1.3 color=grey
d.mon start=x0
d.rgb red=luftbild.1 green=luftbild.2 blue=luftbild.3
r.composite r_map=r$1.1 g_map=r$1.2 b_map=r$1.3 output=r$1
d.rast map=$1 #oder d.rgb red=r$1.1 green=r$1.2 blue=r$1.3

g.remove rast=r$1.1
g.remove rast=r$1.2
g.remove rast=r$1.3
```

2.2 Methoden der Digitalisierung in GRASS

GRASS bietet verschiedene Methoden um Rasterbilder in Vektorbilder umzuwandeln. Um sich für eine Methode zu entscheiden, wird versucht mit allen vorhandenen Methoden ein Ergebnis zu erzielen. Die Ergebnisse dieser Versuche werden nun kurz aufgeführt. Im Anschluss daran wird die manuelle Digitalisierung mit **v.digit**, welche für die Aufgabe am geeignetsten erschien ausführlicher vorgestellt. Im folgenden noch eine Übersicht über die Methoden der Digitalisierung.



2.2.1 Automatische Digitalisierung

Bei dieser Variante der Digitalisierung werden die 3 Farbbänder des Rasterbildes benötigt. Die Vektoren werden mit Hilfe einer Clusteranalyse erzeugt und dann erstellt.

Zuerst müssen die drei Bänder des Luftbildes mittels **i.group** gruppiert werden, dabei muß noch eine Subgruppe definiert werden. Danach wird die Clusteranalyse mit **i.cluster** durchgeführt. Es können dabei verschiedene Parameter eingestellt werden und es wird ein „Result signature file“ angelegt, indem die Daten der einzelnen Punkte gespeichert werden. Danach wird das Modul **i.maxlik** ausgeführt, welches die Klassifikation des Luftbildes durchführt. Als Ergebnis bekommt man eine klassifizierte Rasterkarte, welche schlecht bis gar nicht geeignet ist um eine Vektorkarte zu erzeugen, da die Klassifizierungen der Straßen sehr schlecht vorgenommen wurden. Deswegen braucht wird hier nicht auf die eigentliche Vektorisierung eingegangen, sie würde aber im Prinzip genauso funktionieren wie bei der halbautomatischen Methode.

2.2.2 Halbautomatische Digitalisierung

Bei der halbautomatischen Digitalisierung müssen Trainingsgebiete definiert werden, an denen sich die einzelnen Module zum auffinden von Klasseneigenschaften orientieren. GRASS bietet hierzu zwei verschiedene Methoden ein Vektorbild zu erzeugen.
Bei der ersten Variante wird mit **i.class** gearbeitet, dies ist ein Modul mit dem über die Trainingsflächen und deren Historgramme eine Klassifizierung vorgenommen wird. Vorher müssen die Luftbilder erst über **i.group** gruppiert werden. Nachdem das Modul ausgeführt wurde, muß wie auch bei der automatischen Digitalisierung **i.maxlik** ausgeführt werden. Dabei werden die einelnen Pixel der jeweiligen Klasse zugeordnet.
Mit diesem Modul, gelingt es jeweils nur eine Klassifizierung durchzuführen, d.h. es können nicht mehrere Klassen festgelegt werden. Dieses Ergebnis ist für die Anwendung nicht brauchbar, da in einem Luftbild durch Schattenbildung,z.Bsp. die Strassen ziemlich hell aber auch ziemlich dunkel dargestellt sein können.
Die zweite Möglichkeit eine Klassifizierung durchzuführen ist über **r.digit**. Bei diesem Modul müssen manuell Trainingsgebiete festgelegt werden. Vorher müssen die Luftbilder erst über **i.group** gruppiert werden. Bei dem Aufruf von **r.digit** muß ein Monitor mit der Luftbildkarte geöffnet sein, dann wird A für neue Area ausgewählt. Nun kann man mit der Maus ein Trainingsgebiet festlegen. Dabei muß die Karte schon soweit herangezoomt sein, dass man nur noch mit Mausklicks das Gebiet festlegen muß. Bei jedem Aufruf von **r.digit** kann nur eine Klasse festgelegt werden, d.h. um mehrere Klassen zu erzeugen, muß **r.digit** mehrfach ausgeführt werden. Jedesmal wenn **r.digit** ausgeführt wird, legt GRASS eine neue Rasterkarte an. Um mehrere dieser Rasterkarten zu vereinen muß **r.mapcalc** mit folgenden Parametern ausgeführt werden:

```
r.mapcalc
result='if(isnull(digit.1),digit.3,digit.1)'.
```

Wenn man diese Trainingsgebiete nun vorliegen hat, führt man **r.gensigset** aus. Mit diesem Modul wird die Clusteranalyse anhand der Trainingsgebiete durchgeführt. Um dann die Punkte wieder einer Klasse zuzuordnen, wird in diesem Fall **i.smap** ausgeführt.
Als Resultat bekommt man eine Rasterkarte mit einer brauchbaren Klassifizierung für die Straßenachsen. Man sollte die Klassen der Häuser auf „no data“ Werte setzen, da sie so nicht mit als Vektoren umgewandelt werden. Die Umwandlung dieser „no data“ Klassen ist auch nicht nötig , da bei der Umwandlung unbrauchbare Ergebnisse entstehen würden.

Die eigentliche Umwandlung in eine Vektorkarte wird nun mit **r.thin** und anschließend mit **r.line** durchgeführt. Wobei **r.thin** eine Skelettierung der Rasterkarte durchführt und die „no data“ Punkte herausnimmt. Das Modul **r.thin** kann immer ohne Probleme ausgeführt werden, es erzeugt jedoch eine Karte bei der noch zuviel kleinste Linien vorhanden sind, sodass man Probleme bei ausführen von **r.line** bekommt. Es tritt dann folgender Fehler auf: „bitte führen sie **r.thin** durch“. Somit kommt man zu dem Ergebnis, dass man die Vektorisierung auch mit der halbautomatischen Methode nicht durchführen kann.

2.2.3 weitere Methoden und Übersicht

Ein weitere Möglichkeit die Digitalisierung durchzuführen ist die Funktion **r.poly**. Dabei bekommt man auch kein brauchbares Ergebnis, da man **v.support** zwar ausführen kann es sich aber, wahrscheinlich aufgrund der Größe, nicht in absehbarer Zeit ausführen lässt.

2.2.4 Manuelle Digitalisierung in GRASS

Die manuelle Digitalisierung erfolgt mit dem Modul **v.digit** und ist nach einer geringen Eingewöhnungszeit recht gut beherrschbar. Vorher sollten aber noch ein paar Vorarbeiten gemacht werden.

Luftbilder zusammenfügen: Da eine große Anzahl von Luftbildern digitalisiert werden soll, ist es sinnvoll mehrere Luftbilder zusammenzufassen. Dies macht die manuelle Digitalisierung ein wenig komfortabler und spart Zeit, da nicht so viele Überschneidungsbereiche berücksichtigt werden müssen.
Es gibt 2 verschiedene Möglichkeiten Rasterbilder zu verschmelzen, die erste ist mit dem Modul **r.mapcalc**. Zuvor muß man die Region auf die Eckpunkte der neuen Rasterkarte einstellen. Das Modul **r.mapcalc** kann unterschiedlichste Berechnungen durch arithmetische Ausdrücke an max. 2 Rasterkarten durchführen, sowie neue Karten berechnen und erstellen. In dem Anwendungsbeispiel geschieht dies mit einer if-Bedingung. Nach Ausführung der Verschmelzung ist ein Rasterbild mit 2 Farbtabellen entstanden. Da ein solches Bild mehr Rechnerzeit benötigt,wenn man es anzeigen und bearbeiten möchte, ist es nötig die Farbtabelle auf eins der vorherigen Bilder zu setzen. Dies geschieht mit dem Modul **r.colors**.

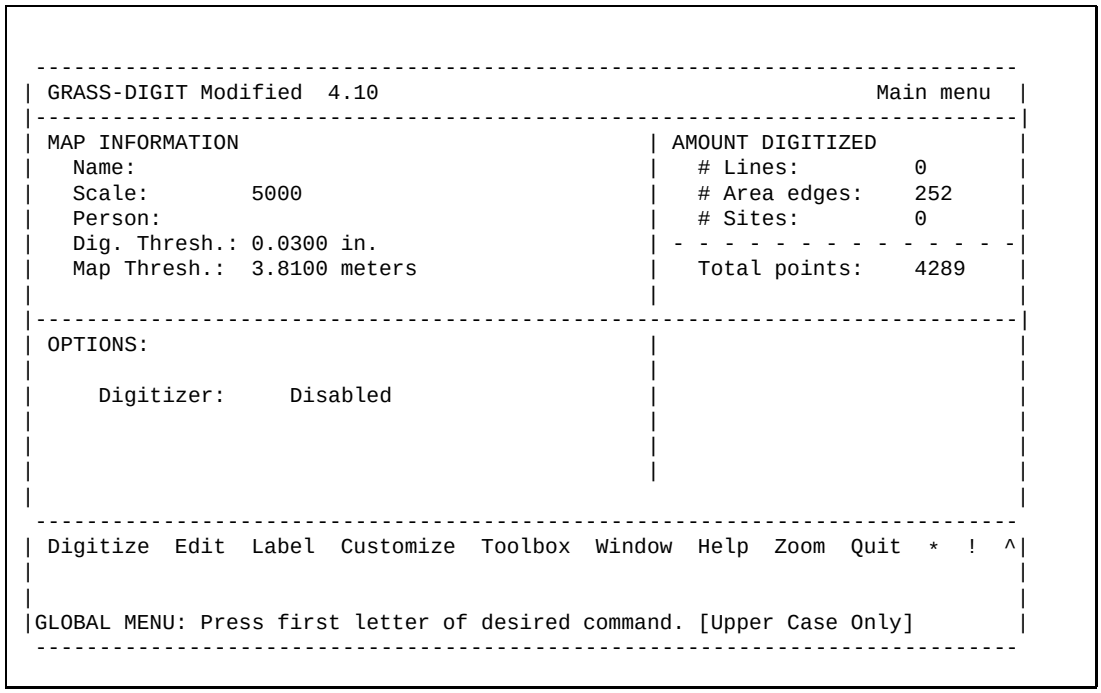
```
g.region n=5794000 s=5793000 e=3436000 w=3434000
r.mapcalc result = 'if(isnull(r3493),r3593,r3493)'
r.colors map=result rast=r3493
```

Die zweite Möglichkeit Rasterbilder zu verschmelzen funktioniert über das Kommando **r.patch**. Dabei ist es möglich mehrere Rasterkarten gleichzeitig zu verschmelzen. Man sollte aber nicht mit mehr als 4 Karten arbeiten, da sonst die Zeit zur Berechnung der Karte sehr groß ist. Vorher ist es auch nötig die Region mit **g.region** auf die Eckpunkte der Region zu setzen. Hinterher muß wie bei **r.mapcalc** die Farbtabelle der neuen Karte mit **r.colors** angepasst werden.

```
g.region n=5794000 s=5793000 e=3436000 w=3432000
r.patch input=r3293,r3393,r3493,r3593 output=comp.1
r.colors map=comp.1 rast=r3293
```

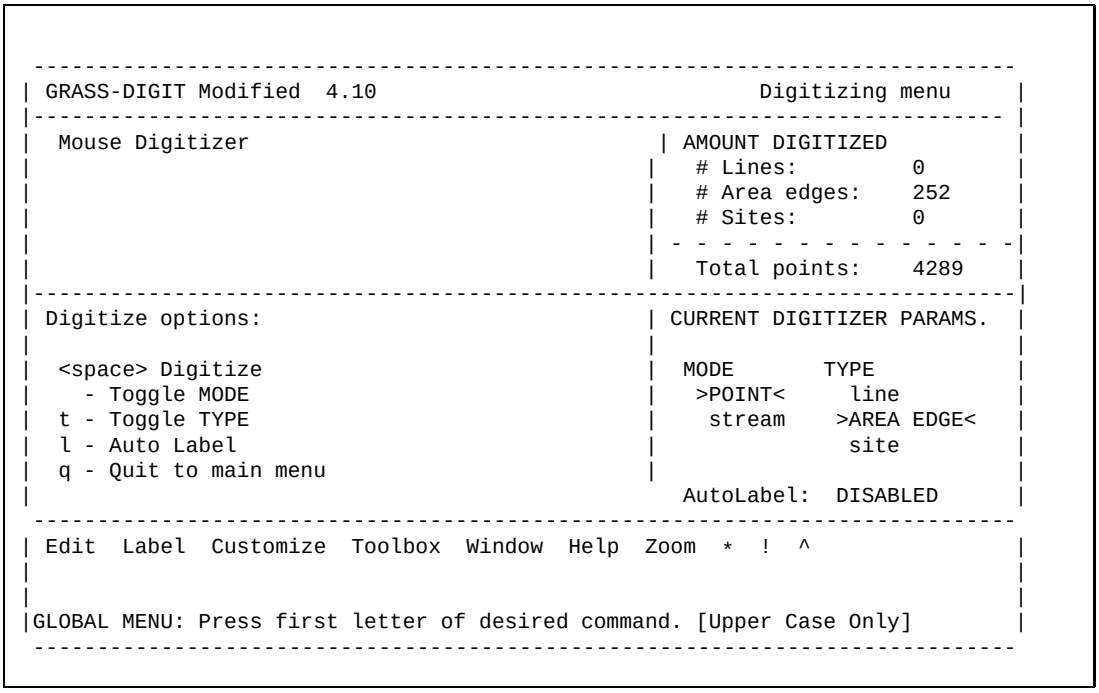
Man kann eine große Menge an Luftbildern (ca.23) verschmelzen, aber man sollte sich sehr gut überlegen wie man die Bereiche wählt. Ab einer Anzahl von ca.16 Rasterbildern, merkt man in **v.digit**, dass spürbar mehr Rechenzeit benötigt wird um das Bild aufzubauen.

Vektor-Digitalisierung am Bildschirm: Nun kann man anfangen mit dem Modul **v.digit** zu arbeiten, zuvor muß ein Monitor mit **d.mon start=x0** gestartet werden. Als erstes sollte man wählen wie man digitalisieren möchte, dabei wird die „3“ für „none“, also Bildschirmdigitalisierung angegeben. Danach wird nach einer Vektorkarte gefragt. Hier kann man eine neue Vektorkarte anlegen, indem man einfach den Namen der Karte angibt. Der nächste Bildschirm der sich öffnet beinhaltet wichtige Basisinformationen für die Vektorkarte, z.Bsp. den Maßstab und die Kartengrenzen. Diese sollten auch richtig angegeben werden, da sonst keine korrekte Anzeige der Karte möglich ist und auch die „Snappingwerkzeuge“ nicht richtig funktionieren. Wenn man alle benötigten Eingaben getätigt hat, gelangt man in das Hauptmenü. Die Navigation darin ist ein wenig gewöhnungsbedürftig, aber nach einer Weile kommt man damit sehr gut zurecht. Um in die Untermenüs zu kommen muß man jeweils den Anfangsbuchstaben des Menüs als Großbuchstabe und um die Untermenüs zu verlassen muß man q eingeben. Das Zoommenü ist von allen Untermenüs aus erreichbar.



Als erstes soll das Rasterbild, von welchem digitalisiert werden soll, in den Hintergrund geladen werden. Dafür muß man in das Untermenü Customize wechseln (C) und darin „Select a Backdrop Cellmap“ (B) wählen, danach einfach noch den Namen des Rasterbildes angeben und schon wird es auf dem Monitor ausgegeben. Nun kann man mit (D) in das eigentliche Digitalisierungsmenü wechseln. Mit drücken der Leertaste springt man in den eigentlichen Digitalisierungsvorgang. Aber zuvor sollte man noch einige Grundeinstellungen überprüfen.
Zuerst sollte man sich sicher sein ob man Linien, Flächen oder Punkte digitalisieren möchte. Die Grundeinstellung „TYPE“ liegt bei area. Es ist zu beachten, dass bei jedem Neustart von **v.digit** diese Grundeinstellung wieder vorgenommen wird. Möchte man also Linien digitalisieren, so muß man mit

Toggle Type (t) den TYPE auf line stellen. Desweiteren sollte darauf geachtet werden das AutoLabel DISABLED ist, da man keine Attributierung vornehmen möchte.



Wenn man diese Einstellungen überprüft hat, kann es mit der Leertaste losgehen. Es öffnet sich ein Menü, zur Orientierung wie man mit der Maus, Linien bzw. Punkte setzen muß und zoomen kann. Dies ist recht leicht verständlich und nach kurzer Eingewöhnungszeit schnell zu bedienen.
An dieser Stelle sollen noch ein paar grundlegende Regeln für die Digitalisierung mit v.digit in GRASS genannt und erläutert werden.

1. Es muß darauf geachtet werden, dass an sich kreuzenden Linien immer ein Knotenpunkt gesetzt wird. Da es im nachhinein schwer möglich ist diese Knotenpunkte zu erzeugen. Es existieren zwar Werkzeuge wie **v.trim** und **v.spag**, aber in der von uns verwendeten Version funktioniert **v.trim** leider nicht. **v.spag** setzt beim ausführen an jede Kreuzung einen Knotenpunkt, dies ist aber für die Anwendung nicht erwünscht, da z.Bsp. an Brücken kein Knotenpunkt gesetzt werden soll, um für eine spätere Anwendung z.Bsp. für Netzverfolgungen keine falschen Verbindungen zu schaffen. Deswegen ist es zwingend erforderlich wirklich an jede Kreuzung die keine Brücke o.ä. ist, einen Knotenpunkt zu setzen, dies ist einfacher als dies im nachhinein noch von Hand zu bearbeiten.

2. Es existiert ein Snappingwerkzeug um Linien auch wirklich in einem Punkt enden zu lassen. Die Snapdistance kann man unter (C) (s) einstellen, sie ist aber in Ihrer Voreinstellung vollkommen ausreichend. Man kann durch die Farbgebung, sehr gut nachvollziehen, ob Linien wirklich in einem Punkt enden. Und man sollte darauf achten, dass auch wirklich gesnappt wird, sonst muß man dies im nachhinein noch per Hand verändern. Dies ist wiederum sehr Zeitaufwendung und kann durch das Snappingwerkzeug vermieden werden.

3. Die Farbgebung der Linien und Punkte kann man unter (C) (C) einstellen. Man kann die Einstellungen so vornehmen, dass z.Bsp. Punkte mit nur einer Linie farbig anders dargestellt werden als Punkte

mit mehreren Linien. Dies ist sehr wichtig für die Snapfunktionen und kann helfen Fehler bei der Digitalisierung aufzudecken. Aber auch für Linien können diese Farbeinstellungen sehr wichtig sein, z.Bsp. kann man sehr gut überprüfen, ob man auch als Type line oder area eingestellt hat.

4.Für Nachbesserungen oder Aktualisierungen existiert ein Untermenü zur Bearbeitung der digitalisierten Elemente (E). Man kann Linien wieder löschen (r), diese sind dann allerdings nur für den Nutzer gelöscht, in der Datei aber noch vorhanden. Um sie endgültig zu löschen, muß **v.spag** mit der -i Flagge ausgeführt werden. Es ist darauf zu achten, dass **v.spag -i** angegeben wird, sonst werden noch Knotenpunkte an sich kreuzenden Linien gesetzt, obwohl man dies gar nicht möchte. Es können auch Linien gebrochen und somit Knotenpunkte gesetzt (b) oder verschoben (m) werden. Eine manuelle Nachbearbeitung ist also jederzeit möglich, aber sehr zeitintensiv.

Wenn man mit dem digitalisieren fertig ist, kann das Modul **v.digit** verlassen werden. Um die gelöschten Linien aus der Vektorkarte zu entfernen, führt man **v.spag -i** aus. Um dann noch die Topologie der Elemente der Karte zu bilden muß **v.support** durchgeführt werden, geschieht dies nicht, kann die Karte weder exportiert noch weiterbearbeitet werden. Wenn man sich die Vektorkarte in einem Monitor anzeigen lassen möchte, sollte man die Eckpunkte des Monitors mit **g.region** wieder einstellen.

2.3 Export der Daten

Der Export der Daten erfolgt mit dem Modul **v.out.shape**. Dabei sollte bei Polygonen darauf geachtet werden, dass vorher alle Polygone mit **v.alabel** ein Attribut zugewiesen bekommen. Der Wert des Attributes wird nicht weiterverwendet, aber muß für GRASS existent sein um den Export durchführen zu können. GRASS legt dann die 3 zu dem Shapefile gehörenden Dateien (.shp .shx und .dbf) in dem Arbeitsverzeichnis an.

Anmerkung: Bei dem arbeiten mit GRASS sollte man darauf achten, dass die Region relativ oft verändert wird, unter anderem auch beim zoomen. Dadurch können schnell Fehler auftreten, wie z.Bsp. die Ausgabe eines leeren Bildschirmes. Möchte man also verschiedene Module benutzen sollte man immer kontrollieren, ob noch die richtige Region eingestellt ist und vorsichtshalber nochmal **g.region** durchführen.

Digitalisierung in GRASS **Übersicht der verwendeten Module**
Import r.in.gdal g.region r.colors r.composite
Digitalisierung d.mon g.region r.patch r.colors v.digit g.region v.spag -i v.support
Export (v.alabel) v.out.shape (nur bei Polygonen)
Visualisierung d.mon g.region d.rast d.vect

3 Attributierung mit Thuban

Die Aufgabe der Attributierung ist es, den gewonnenen Geo-Objekten, in diesem Fall die Strassenachsen und andere Objekte wie Gewässer und Sehenswürdigkeiten, sinnvolle Eigenschaften zuzuweisen. Dies geschieht in .dbf Dateien, welche zu den Shapefiles dazugehören. Auf das Datenmodell wird im folgenden noch eingegangen.

Um die .dbf Datei bearbeiten zu können,stehen 2 Werkzeuge unter

`http://www.usf.uos.de/~fkoorman/software/dbftools.de.html`

zur Verfügung. Um sich die Dateien gut anschauen zu können, muß man sie mit dbf2txt in Textdateien umwandeln. Eine Textdatei kann man am besten in einem Editor bearbeiten. Man fügt die noch fehlenden Attributspalten ein und benennt sie um.
Wenn man sich die .dbf Dateien genauer anschaut, stellt man fest, dass sie nur aus 2 Spalten bestehen, nämlich der ID, welche Fortlaufend durchnummeriert ist und einer Spalte genannt CAT_ID in der keine Werte stehen. Nach dem bearbeiten der jeweiligen .txt Datei wandelt man die Datei mit txt2dbf wieder in ein .dbf Datei um.

Nun kann man mit der eigentlichen Attributierung anfangen. Zunächst muß Thuban aufrufen und die verschiedenen Layer die man angezeigt haben möchte hineinladen. Dann kann man sich die dazugehörige Tabelle anzeigen lassen. Mit dem Identify Shape kann man sich die jeweilige Linie anzeigen lassen und dort auch die Veränderungen vornehmen. Die Attributierung führt man dann an jeder Linie einzeln durch.

4 Das Datenmodell

Für die vorliegenden Daten wurde folgendes Datenmodell verwendet. In der .dbf Datei der Strassen wurden 5 Attribute spezifiziert. Dies sind der eindeutige Schlüssel (strShapeID), der Strassenname (strID), welcher als Zahl gehalten wird, um doppelte Datenhaltung zu vermeiden. Die Strassennamen sind in der Datei strassen_namen.dbf enthalten. Desweiteren der Strassentyp (strTypID), welcher auch als Zahl vergeben wird und auf die Datei strassen_typen.dbf zurückgreift. Es werden die Ebenen (strEbene), für Brücken gespreichert und es wird ein Attribut Spuren (strSpuren) angelegt, welches in unserer Anwendung aber nicht weiter attributiert wird.
Da es in der aktuellen Mapserver Version 3.6 nicht möglich ist, einen join von mehreren .dbf Dateien durchzuführen, ist man gezwungen die Dateien mit den Strassennamen und den Strassen, möglichst schon vor der Mapserveranwendung zu Joinen. Dies ist nötig, damit man sich später auch die Strassennamen anzeigen lassen kann.Gegebenenfalls muß dies auch für andere Layer, wie z.Bsp. die Sehenswürdigkeiten durchgeführt werden.
An dieser Stelle soll dieser Join exemplarisch gezeigt werden. Zum bearbeiten der .dbf Files müssen diese mit

```
dbf2txt -d, strassen.dbf > strassen.txt
dbf2txt -d, strassen_namen.dbf > strassen_namen.txt
```

in Textdateien umgewandelt werden, diese sehen dann in etwa folgendermaßen aus.

```
strassen.txt                                              strassen_namen.txt

#strShapeID,strID,strTypID,strSpuren,strEbene               #strNameID,strName
1,1,0,0,0                                                   0,nicht attributiert
2,2,3,0,0                                                   1,kein Name vorhanden
3,4,2,0,0                                                   2,Georgstraße
4,0,1,0,0                                                   3,Hasestraße
5,3,4,0,0                                                   4,Torwall
```

Nun muß ein join durchgeführt werden, bei dem die Strassennamen an die strassen.txt angefügt werden. D.h. die 2. Spalte der strassen.txt und die 1.Spalte der strassenname_tab.txt müssen miteinander gejoint werden.

```
strassenjoin.txt
#strShapeID,strID,strTypID,strSpuren,strEbene,strName
1,1,0,0,0,kein Name vorhanden
2,2,3,0,0,Georgstraße
3,4,2,0,0,Torwall
4,0,1,0,0,nicht attributiert
5,3,4,0,0,Hasestraße
```

Um den Join ausführen zu können, muß in der strassen.txt die 2. Spalte an die erste Stelle gesetzt werden und dann nach dieser Spalte sortiert werden. Zuvor werden noch die Überschriftenzeilen aus den beiden Dateien gelöscht. Dies geschieht in der Kommandozeile mit:

```
grep -v "^#" strassen.txt > strassen1.txt
grep -v "^#" strassen_namen.txt > strassen_namen1.txt
sed -e "s/\(.*\),\(.*\),\(.*,.*,.*\)/\2,\1,\3/" strassen1.txt > strassensed.txt
```

nun kann man nach der ersten Spalte sortieren, wobei man darauf achten muß, dass das Ergebnis in eine Datei geschrieben wird und das eine numerische Sortierung vorgenommen wird.

```
sort strassensed.txt -o strassen_sort -n
```

Da die strassenname_tab schon sortiert ist kan nun der Join erfolgen, dabei muß darauf geachtet werden, dass auch alle Werte die nicht in der 2. Datei vorkommen mit übernommen werden.

```
join -t, -n -a 1 strassen_sort strassen_namen1.txt >strassenjoin.txt
```

Ist dies geschehen muß man noch in einem Editor die Spalten der strassenjoin.txt zurücktauschen und

wieder nach der ersten Spalte sortieren. Danach muß man nur noch die Überschriftenzeile einfügen und die Textdatei in .dbf umwandeln und dem zugehörigen Shapefile mitgeben.

```
sed -e "s/\(.*\),\(.*\),\(.*,.*,.*,.*\)/\2,\1,\3/" strassenjoin  > strassen_join.txt
sort strassenjoin.txt -o ergebnis.txt -n
echo \#strShapeID,strID,strTypID,strSpuren,strEbene,strName > ergebnis.txt
cat strassen_join.txt >> ergebnis.txt
txt2dbf -I10 -I10 -I10 -I10 -I10 -C50 -d, ergebnis.txt ergebnis.dbf
```

Die Attributierung der einzelnen Layer wurde wie folgt durchgeführt:

gewaesserflaechen: als Polygone	
gpShapeID	gpNameID
Integer[4]	Char [50]

gewaesserlinien: als Linien	
glShapeID	glNameID
Integer[4]	Integer [4]
glNameID ist mit gewaesserlinien_namen.dbf verknüpft	
gewaesserlinien_namen.dbf	
glNameID	glName
Integer[1]	Char [50]

gruenflaechen: als Polygone		
gfShapeID	gfTypID	gfName
Integer[4]	Integer [2]	Char[50]
gfTypID ist mit gruenflaechen_typen.dbf verknüpft		
gruenflaechen_typen.dbf		
gfTypID	gfTypName	
Integer[1]	Char [50]	

poi(Points of interest): als Punkte		
poiShapeID	poiTypID	poiName
Integer[4]	Integer [4]	Char[50]
poiTypID ist mit poi_typen.dbf verknüpft		
poi_typen.dbf		
poiTypID	poiTypName	
Integer[1]	Char [50]	

strassen: als Linien				
strShapeID	strID	strTypID	strSpuren	str Ebene
Integer[5]	Integer [4]	Integer[4]	Integer[2]	Integer[2]
strID ist mit strassen_namen.dbf verknüpft				
strassen_namen.dbf				
strNameID	strName			
Integer[4]	Char [50]			
strTypID ist mit strassen_typen.dbf verknüpft				
strassen_typen.dbf				
strTypID	strTypName			
Integer[2]	Char [50]			

5 Mapserveranwendung

Die Aufgabe besteht darin die gewonnenen Freien Geodaten in geeigneter Weise als Web Mapping Anwendung zur Verfügung zu stellen. Als Programm dient der UMN Mapserver,welcher aus mehreren Komponenten besteht. Auch hier gibt es sehr gute Einführungen auf den Heimatseiten des Mapservers http://mapserver.gis.umn.edu/index.html Als erstes wird eine HTML Seite benötigt, von der der Mapserver angestoßen werden muß. Dabei müssen bestimmte Parameter an den Mapserver übergeben werden,wie z.Bsp. wo sich das Programm befindet und welche Layer dargestellt werden sollen. Am besten löst man dieses Problem mit einem Link oder mit einem Button:

```
 <form method=get action="/cgi-bin/mapserv">
    <input type="hidden" name="map" value="/spare/mapserver/
     osnabrueck/monika/osnabrueck.map">
    <input type="hidden" name="layers" value="sonststrassen,
     nebenstrassen,khauptstrassen,bundesstrassen,
autobahn,Beschriftung">
    <input type="hidden" name="program" value="/cgi-bin/mapserv">
    <input type="hidden" name="map_web_imageurl" value="/tmp/">
    <p align=center>
    <input type="submit" value="Stadtplan Osnabr&uuml;ck">
 </form>


<a href="/cgi-bin/mapserv?map=%2Fspare%2Fmapserver%2Fosnabrueck
%2Fmonika%2Fosnabrueck.map&layers=sonststrassen+nebenst
rassen+hauptstrassen+bundestrassen+autobahn+Beschriftung&program
=%2Fcgi-bin%2Fmapserv&map_web_imageurl=%2Ftmp%2F"><b>Mapserver</b></a>
```

Um nun diesen Link ,bzw. diesen Button ausführen zu können benötigt der Mapserver ein sogenanntes Mapfile(hier osnabrueck.map). In diesem werden Grundeinstellungen des Mapservers eingestellt und Legende, Maßstab, Layer, Beschriftung usw. festgelegt. Desweiteren wird ein Template benötigt, indem die Darstellung der Ergebnisse des Mapservers geregelt wird. Dies ist eine HTML Datei mit Möglichkeiten auf den Mapserver zuzugreifen. Möglichkeiten zur Gestaltung und Funktionen des Mapfiles und Templates findet man auf unter http://mapserver.gis.umn.edu/doc36/
Im folgenden soll nicht weiter auf die einzelnen Mapserverdateien eingegangen werden, da diese Erläuterungen zu aufwendig sind.Die Mapserverandwendung kann am Anfang sehr einfach aufgebaut sein und ist dann sehr leicht beliebig erweiterbar. Ausführliche Hilfe wie eine Mapserverandwendung aufgesetzt werden kann, findet man auf der Homepage. Die einzelnen Templates und das Mapfile zu der Beispielanwendung finden sich auf der Downloadseite. Die Mapserverandwendung kann am Anfang sehr einfach aufgebaut sein und ist dann sehr leicht beliebig erweiterbar.

6 Aktualisierung

Bei der Beschreibung der Aktualisierung wird nun davon ausgegangen, dass man die Daten im Generate Format (.gen .att) vorliegen hat. Wie man Daten vom Generate Format in Shapefiles umwandeln kann und umgekehrt wird im folgenden Kapitel beschrieben. Es können nun mehrere Fehler auftreten, einfache Attributfehler oder Koordinatenfehler. Desweiteren kann es sein, dass Änderungen in den Attributklassen durchzuführen sind.

6.1 Umwandlung zwischen Generate- und Shapefiles

Sowohl für die Umwandlung von Generate- in Shapeformat, als auch von Shape- in Generateformat existieren einfache Programme. Um Generate- in Shapeformat umzuwandeln, muß das Programm gen2shp ausgeführt werden. Zu finden ist es unter: http://intevation.de/ jan/gen2shp/gen2shp.html
Der umgekehrte Fall von Shape- in Generateformat existiert, ist aber noch nicht veröffentlicht. Er wird aber in absehbarer Zeit von Intevation zur Verfügung gestellt.

6.2 Attributfehler

Wenn man einen Fehler in der Attributierung feststellt, kann dieser relativ leicht behoben werden.
Es gibt 2 Varianten die Änderungen vorzunehmen. Die erste Variante ist, die Attribute einfach in der Textdatei zu ändern. Dazu ist es erforderlich die ID des zu ändernden Objektes zu wissen. Dann kann in der zugehörigen .att Datei mittels einem Editor die Änderung des Attributes vorgenommen werden. Bei der zweiten Variante, wandelt man seine .gen und .att Dateien in Shapeformat um und lädt sie dann in Thuban. Nun können mit dem Identity Tool die Änderungen an den Attributen vorgenommen werden. Da bei der Attributierung die .gen Datei nicht verändert wird, kann die alte .gen Datei beibehalten werden. Um die .att Datei zu bekommen wandelt man sich die neue .dbf Datei einfach mit dbf2txt um und speichert sie als .att Datei.

```
dbf2txt  -d, strassen.dbf > strassen.att
```

6.3 Koordinatenfehler

Die Behebung von Koordinatenfehlern, gestaltet sich im Moment noch ein wenig kompliziert, da GRASS nicht die Möglichkeit bietet mehrere Attribute einzulesen. So muß nach dem eigentlichen beheben der Fehler noch ein Join durchgeführt werden, der es ermöglicht die neuen Linien (Punkte oder Polygone) mit den Attributen zusammenzufügen. Dies wird im Kapitel Join beschrieben.
Die Behebung der Koordinatenfehler erfolgt in GRASS mit dem Modul **v.digit**. Zuvor muß man die Vektorkarte mit **v.in.shape** in GRASS importieren. Als Attribut muß man die Spalte ID an GRASS übergeben und als Label Name die Spalte mit dem Namen. GRASS übernimmt nur diese beiden Spalten aus der .dbf Datei. Nun kann wie schon vorher beschrieben die Digitalisierung oder Änderungen vorgenommen werden. Also die Region mit **g.region** festlegen, **v.support** ausführen und dann **v.digit** durchführen. Am Ende nicht vergessen nochmals **v.support** auszuführen um die Topologie der Elemente zu bilden. Danach werden die Daten mit **v.out.arc** exportiert um dann gleich ein Generate Format zu erhalten. GRASS übergibt nun nach aussen seine (in GRASS selbst nicht zu erkennende) ID und die CAT_ID.Die Dateien die bei dem Export der Daten aus GRAS entstehen heißen .lin und .txt. Sie müssen noch von Hand in .gen und .att umbenannt werden. Für den Export von Punktdaten gibt es nur die Möglichkeit diese als ASCII oder als Shapedateien zu erhalten. Diese muß man dann in das Generate Format umwandeln. Man hat jetzt also eine alte und eine neue .att Datei und diese beiden Dateien müssen nun miteinander gejoint werden. Danach hat man seine Daten wieder im Textformat vorliegen und kann damit arbeiten.

6.3.1 Join

Das zusammenfügen von einer neuen Datei mit nur einem Attribut mit der alten Datei mit mehreren Attributen, soll an dieser Stelle durch ein Beispiel veranschaulicht werden. Es kann sowohl für die

.dbf Dateien der Shapefiles durchgeführt werden, sowie für die .att Dateien des Generate Formates. Der einzige Unterschied ist das man bei den .dbf Dateien noch jeweils eine Umwandlung von dbf2txt und txt2dbf durchführen muß.
Bei diesem Beispiel wurde die 2.Linie in 2 neue Linien 4 und 5 geteilt. GRASS schreibt, die neu entstandenen Linien ans Ende seiner ID Liste und verschiebt, die freigewordenen Plätze in der Liste werden durch die bestehenden ID's ersetzt, bleiben aber in der bestehenden Reihenfolge erhalten.

```
alte.att                                neue.att

#ID,Str_ID,Str_typ,Ebene                #ID, CAT_ID
1,2,0,0                                 1,1
2,324,3,0                               2,3
3,22,2,0                                3,4
4,0,1,0                                 4,2
                                        5,2
```

Nun muß ein join durchgeführt werden, bei der in der neuen Datei die 2. Spalte durch die 2.-4. Spalte der alten Datei ersetzt wird, d.h.die erste Spalte der alten muß mit der zweiten Spalte der neuen Datei gejoint werden. Das Ergebnis soll wie folgt aussehen:

```
nachdemjoin.att
#ID,Str_ID,Str_typ,Ebene
1,2,0,0
2,22,2,0
3,0,1,0geteielt

4,324,3,0
5,324,3,0
```

Im folgenden wird der Ablauf des joinens für diesen speziellen Anwendungsfall beschrieben.

die beiden Spalten der neuen Datei müssen vertauscht und dann nach der ersten Spalte sortiert werden.

```
sed -e "s/\(.*\),\(.*\)/\2,\1/"  neu.att|sort -n  >neu_sorted
```

Nun wird der eigentliche Join durchgeführt mit dem -a Flag, dafür dass alle zeilen übernommen werden sollen und - n für alle Spalten. Desweiteren wird die erste Zeile mit „#“ herausgelöscht

```
join -a 1 -t "," -n  neu_sorted alt.att | grep -v '#'  > xtest.att
```

Nun werden die erste Spalte, da sie ja nicht mehr gebraucht wird herausgeschnitten und an die Nullerzeilen werden auch 4 Spalten angehängt

```
sed -e "s/\(^0,[0-9]*\)/\1,,,/" xtest.att | cut -d "," -f 2-5  | \
 sort -n  > x1test.att
```

Nun muß noch die Überschriftenzeile und gegebenenfalls andere Kommentarzeilen eingefügt werden.

```
echo \#Id,Str_ID,Str_typ,Ebene > ergebnis.att
cat x1test.att >> ergebnis.att
```

Nun hat man die neuen Daten in der ergebnis.att gespeichert und muß diese Datei nur noch mit der gewünschten .gen Datei verknüpfen.
Wenn man nun möchte kann man Attributfehler beseitigen, entweder durch Umwandlung in Shape-format und dann durch Bearbeitung in Thuban oder man nimmt die Änderungen direkt an der .att Datei vor.

6.4 Attributklassenfehler

Möchte man Änderungen an den Attributklassen durchführen, so kann man dies durch Umwandlung der jeweiligen .dbf Datei in eine .txt Datei und umgekehrt durchführen. Dies soll an einem Beispiel kurz erläutert werden. Bei den Grünflächen soll zu den bestehenden Attributen Wiese und Wald noch das Attribut Erholungsflächen hinzugefügt werden. Zuerst muß die Datei gruenflaechen_typen.dbf in eine Textdatei umgewandelt werden.

```
dbf2txt  -d, gruenflaechen_typen.dbf > gruenflaechen_typen.txt
```

Dann kann das Attribut mit Hilfe eines Editors eingefügt werden, sodass die Datei um das Attribut erweitert wird.Danach erfolgt die Umwandlung in eine .dbf Datei.

```
txt2dbf -I10 -C40 -d, gruenflaechen_typen.txt  gruenflaechen_typen.dbf
```