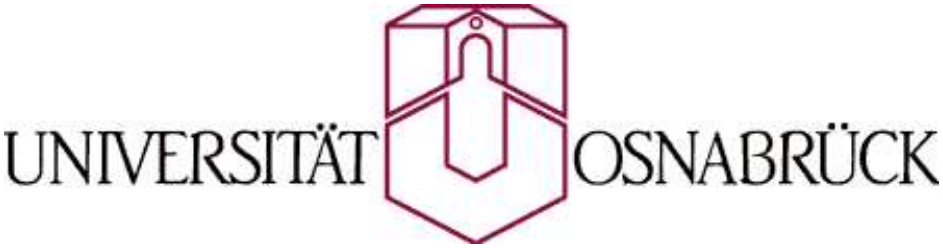

PlaceToBe

Webbasierte Hilfe zur Wohnortsuche
am Beispiel Osnabrück

Bachelorarbeit
von
Hanna Kraeusel

betreut von
Prof. Dr. Oliver Vornberger
Zweitgutachten von
PD Dr. Barbara Hammer

Fachbereich Mathematik/Informatik
Universität Osnabrück

22. Januar 2004

Vorwort

Diese Bachelorarbeit entstand an der Universität Osnabrück im Fachbereich Informatik. Der praktische Teil dieser Arbeit ist unter http://www.inf.uos.de/prakt/pers/dipl/hkraeuse.php erreichbar.

Danksagung

Folgenden Personen möchte ich hier für die Unterstützung bei der Erstellung dieser Bachelorarbeit danken:

- Herrn Prof. Dr. Oliver Vornberger für die Übernahme der Betreuung dieser Arbeit und seinen konstruktiven Hinweise und Anregungen
- PD Dr. Barbara Hammer für die Übernahme des Zweitgutachtens und ihre hilfreichen Hinweise
- Ralf Kunze für die sehr gute Betreuung während der Arbeit, für seine hilfreichen Vorschläge und Hinweise und das Korrekturlesen dieser Arbeit
- Dorothee Langfeld für die Unterstützung und das Korrekturlesen der Arbeit
- Friedhelm Hofmeyer für die technische Unterstützung
- meiner Familie für den Rückhalt den sie mir gegeben hat und das Korrekturlesen der Arbeit

GNU Lizensen

Das Produkt GRASS und die Daten von Frida unterliegen GNU Lizensen. GRASS und die Quellen von Frida unterliegen der *GNU General Public License (GPL)*. Der UMN Mapserver unterliegt der *Mapserver License* basierend auf der MIT Lizenz[1].

[1] http://www.opensource.org

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Kaum jemand verbringt sein ganzes Leben an ein und demselben Ort. Irgendwann muss fast jeder einmal an einen anderen Ort ziehen und die Arbeitsmarktsituation der Zukunft wird durch die Globalisierung noch mehr Mobilität erfordern. Bei solch einem Umzug ergeben sich viele Probleme. Etwa die Suche nach einem geeigneten Wohnort in der neuen Stadt, besonders wenn dem Suchenden die Stadt noch völlig unbekannt ist.
Individuelle Bedürfnisse bedingen verschiedene Wohnorte, z.B. werden Familien in der Nähe von Kindergärten und Schulen wohnen wollen und Kulturbegeisterte die Nähe zum Theater suchen.
Das Ergebnis dieser Bachelor-Arbeit soll dazu dienen, Wohnortsuchende, die sich in der neuen Stadt nicht auskennen, zu unterstützen einen Ort zu finden, an dem sie sich von Anfang an wohl fühlen können.

Aufgabe

Ziel ist es, eine Webapplikation zur Verfügung zu stellen, die solch eine Unterstützung bietet. Es werden Kriterien vorgegeben, mit deren Hilfe die Suche nach dem zukünftigen Wohnort eingegrenzt wird.
Der Benutzer wählt aus den vorgegebenen Kriterien einige aus, an Hand derer die Suche nach dem geeigneten Wohnort in der betreffenden Stadt (hier Osnabrück) durchgeführt wird. Die Auswahl erfolgt durch die Angabe der Prioritäten, mit der die Kriterien erfüllt sein sollen. Das Ergebnis, also der darauf passende Bereich in der Stadt, wird auf einem Stadtplan markiert.

Aufbau der Arbeit

Diese Bachelorarbeit ist in zwei Teile gegliedert. Im ersten Teil dieser Arbeit werden die nötigen Grundlagen vorgestellt. Im zweiten Teil wird die Realisierung der Aufgabenstellung mit Hilfe der in Teil I vorgestellten Grundlagen beschrieben.
Die Grundlagen in Teil I setzen sich zusammen aus dem Geografischen Informationssystem GRASS und Fuzzy Logic.
Der zweite Teil beinhaltet:

- die Beschaffung der benötigten Daten,
- die beispielhafte Ergänzung dieser,
- die Berechnung von Distanzen in den sogenannten Distanzkarten,
- das Erstellen eines Scripts zur Berechnung des eigentlichen Wohngebiets,
- die Darstellung des Ergebnisses.

Abschließend werden die Ergebnisse zusammengefasst und ein Ausblick gegeben, wie der Ansatz verbessert und weiter entwickelt werden kann.

Teil I

Grundlagen

Kapitel 2

Geografische Informationssysteme

Ein Geografisches Informationssystem (GIS) ist ein Werkzeug zur Erfassung, Verwaltung, Analyse und Darstellung raumbezogener Informationen. Raumbezogen bedeutet hier, dass die Informationen eindeutig geografisch lokalisierbar sind. Dabei werden die Eigenschaften und Objekte der realen Welt an Hand ihrer geografischen Lage gespeichert und lassen sich so zueinander in Beziehung setzen. Diese Informationen werden in Ebenen, den sogenannten Layern, thematisch getrennt gespeichert. Diese Ebenen kann man sich als durchsichtige Folien vorstellen, die beliebig übereinander gelegt werden können (siehe Abblidung 2.1).
Solche Informationen können Raster-, Vektor und Sachdaten sein (siehe Kapitel 2.1), wobei Rasterdaten georeferenzierte Landkarten oder Satellitenfotos sein können. Vektordaten werden als Punke oder Linien gespeichert und können mit Sachdaten verknüpft werden.
In einem GIS werden Daten, die über einen beobachtbaren Raum erhoben wurden auf Grundlage eines einheitlichen Bezugssystems (Landeskoordinaten) analysiert und dargestellt. Beim Import werden die Daten automatisch so umgewandelt, dass sie in das einheitliche Bezugssystem passen.
Schon in den 50er Jahren kam die Idee für rechnergestützte räumliche Präsentation und Überlagerung auf. In den 60er Jahren wurden dann die ersten Anwendungen der digitalen Bildbearbeitung (Rastertechnik) entwickelt. Das Konzept der unabhängigen Ebenen entstand gleichzeitig im *Harvard Laboratory for Computer Graphics and Spatial Analysis* und an der ETH Zürich. Der Begriff GIS entstand allerdings erst auf dem ersten großen GIS-Symposium 1970. Die Entwicklung in Harvard ging unter Anderem in der Firma ESRI[1] und in dem Produkt ArcInfo auf, die Entwicklung in der Schweiz zunächst in der Firma Digital, dann in der Firma Adasys[2]. ESRI ist nun praktisch welt-

[1] http://www.esri.com/
[2] http://www.adasys.ch/

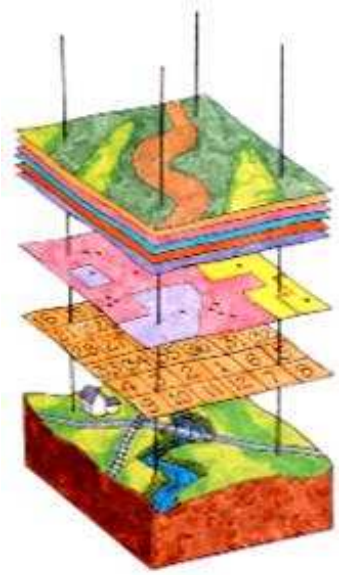

Abbildung 2.1: Ebenen in einem GIS

weiter GIS-Marktführer, wobei Adasys nur in der Schweiz Bedeutung erlangt hat [10].
Die wichtigsten Merkmale eines GIS sind zusammengefasst:

- der Raumbezug,
- die Möglichkeit der Kombination von Datensätzen,
- die Möglichkeit der Generierung neuer Informationen aus bestehenden Informationen durch verschiedene Rechenoperationen,
- die Möglichkeit der Anbindung von Sachdaten,
- die digitale Verfügbarkeit,
- die Maßstabsunabhängigkeit, resultierend aus dem einheitlichen Bezugssystem.

Der Schwerpunkt der GIS-Arbeit liegt auf der Datenauswertung. Eine grundsätzliche Methode ist die raumbezogene Datenabfrage im GIS oder in der gekoppelten Datenbank. Ein GIS ermöglicht die Analyse von Nachbarschaftsbeziehungen zwischen verschiedenen Objekten. Dies ist in einer reinen Datenbankanwendung dagegen nicht adäquat möglich.

Ein anderer Schwerpunkt liegt auf der Erzeugung neuer Daten aus gespeicherten Informationen über algebraische Funktionen oder logische Abfragen. Dadurch können z.B. Flächen mit bestimmten Eigenschaften ausgewiesen werden oder fehlende Werte in einer Oberfläche gleichmäßig verteilter Daten berechnet werden.
Mögliche Einsatzgebiete eines GIS sind der Umweltschutz, die Forstwirtschaft, der Verkehr oder in Gemeinden als prozessunterstützendes Werkzeug beim Kanalmanagement, der Liegenschaftsverwaltung und im Friedhofswesen oder, wie in dieser Bachelor-Arbeit, die mit dem GIS GRASS umgesetzt wurde, demonstriert, zur Unterstützung bei der Wohnortsuche.

2.1 Daten

Räumliche Daten sind Grundlage für die Arbeit mit einem GIS. In einem GIS kann mit vorhandenen Daten gearbeitet werden oder mit Daten, die neu erfasst, also digitalisiert wurden (siehe Abbildung 2.2). Vorhandene Daten können aus unterschiedlichen Quellen bezogen werden.
Die amtlichen Basisdaten der Vermessungsverwaltung sind die Automatisierte Liegenschaftskarte (ALK) und das Amtliche Topographisch-Kartographische Informationssystem (ATKIS). Die ALK kann als die digitale Flurkarte beschrieben werden. Sie wird teilweise von den Landesvermessungsämtern, teilweise von den (Kreis-)Katasterämtern erfasst. Sie liegt aber nur in wenigen Regionen bisher flächendeckend vor und die Erstellung verursacht hohe Kosten. ATKIS-Daten sind dagegen in den meisten Bundesländern mittlerweile flächendeckend verfügbar [3] [11].
Neben diesen Daten existieren auch Basisdaten aus verschiedenen lokalen, regionalen, bundesweiten und internationalen Projekten, in dessen Rahmen mehr oder weniger flächendeckend große Datenbestände erfasst wurden.
Außerdem hat sich ein neuer Dienstleistungssektor entwickelt. Es werden nachfrageorientiert Fachdaten erfasst und zum Verkauf angeboten. Besonders häufig sind dies statistische Daten und Adressdaten für Geo-Marketing-Zwecke. Ein Beispiel für diesen Marketing-Sektor ist die *DDS Digital Data Services GmbH*[4], die digitale Geografiedaten und Potentialdaten (statistische Daten) zum Verkauf anbietet.
Vorraussetzung für die Nutzung von Fremddaten ist die Kenntnis über das Bezugssystem, z.B. das Koordinatensystem, da nur so Daten mit unterschiedlichem Bezugssystem gemeinsam genutzt werden können und die Daten richtig analysiert werden können.

[3] Bezug über die Landesvermessungsämter
[4] http://www.dds.ptv.de/html/geo.frame.html

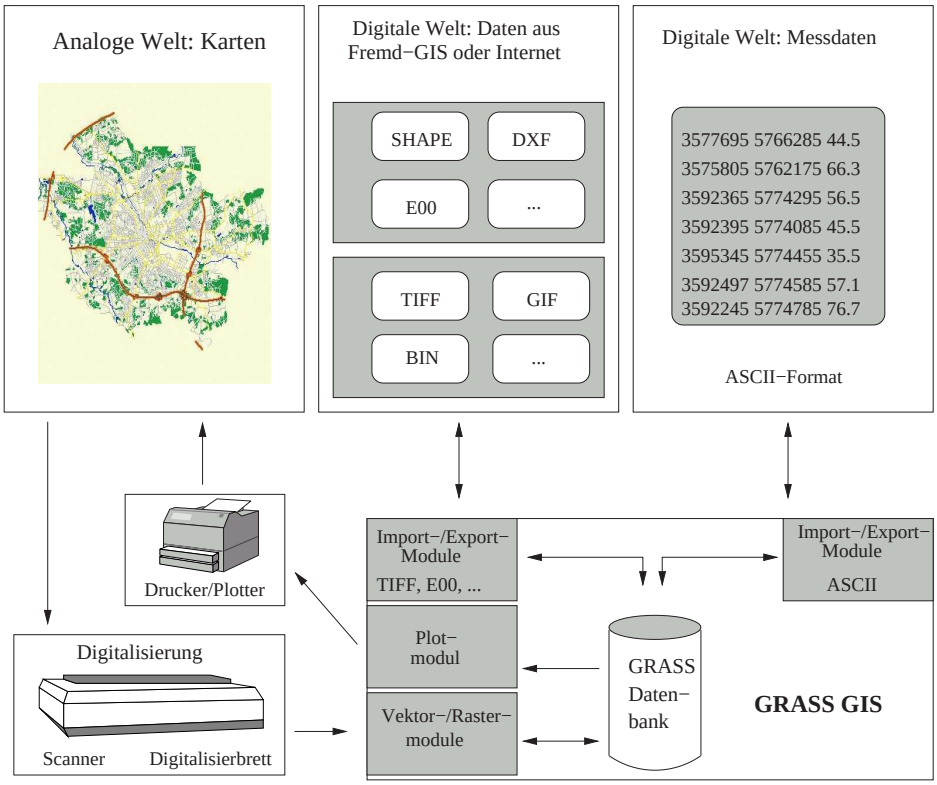

Abbildung 2.2: Datenaustausch bei GRASS

Zur Verwaltung der Daten in einem GIS werden unterschiedliche Datenformate verwendet. Georeferenzierte Daten bestehen aus einer räumlichen, geometrischen oder grafischen, Komponente, die den Ort oder die räumliche Verteilung des Objektes beschreibt und einer zusätzlichen Komponente, die die Eigenschaften des Objekts beschreibt. Die räumliche Komponente kann durch einen der beiden folgenden grundlegenden Ansätze beschrieben werden (siehe Abbildung 2.3):

- die flächige Repräsentation, wobei jeder Punkt (jedes Pixel) im Raum einen zugeordneten Wert (eine Zahl oder Nullwert) hat, dies führt zum Modell der **Rasterdaten**;
- die Repräsentation durch geometrische Objekte, wobei geografische Objekte als Flächen, Linien oder Punkte dargestellt werden, die durch ihre Koordinaten definiert werden, dies führt zum Modell der **Vektordaten**.

Abhängig von der Auflösung der Region eines Projekts, kann sich die Art der Darstellung eines geografischen Objekts ändern. So wird bei einer geringen Auflösung ein Fluß als Linie behandelt und bei einer hohen Auflösung kann er als 3D Objekt dargestellt werden.

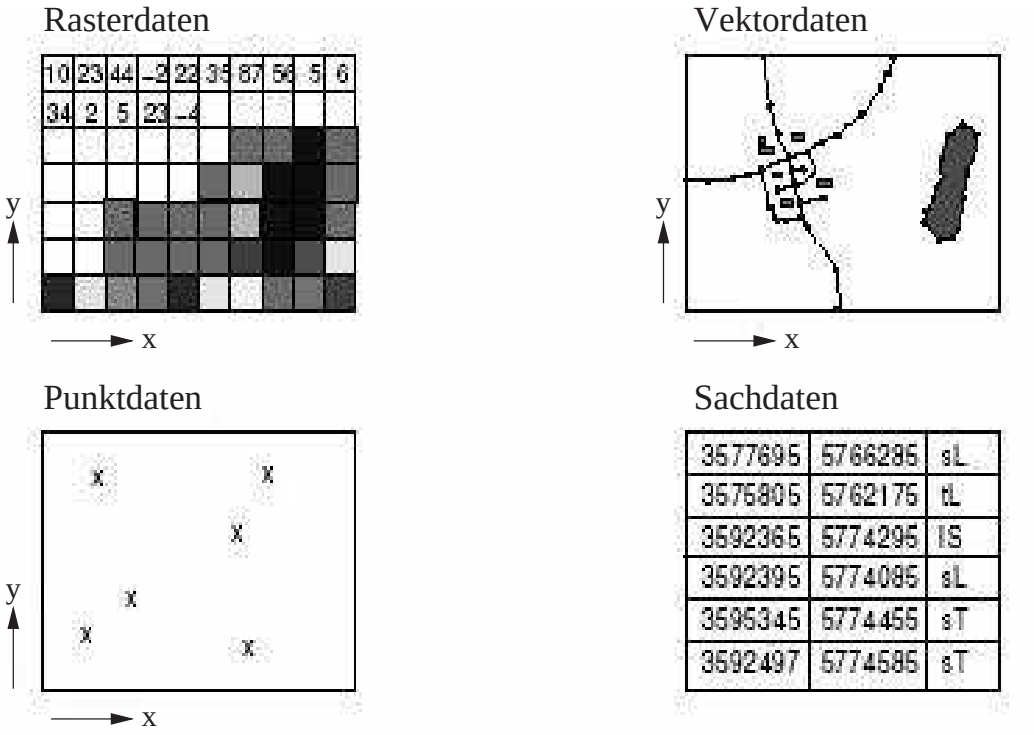

Abbildung 2.3: Datenformate in einem GIS

2.1.1 Rasterdaten

Rasterdaten entstehen durch Scannen von Plänen, Luftbildern u.ä. oder auch direkt bei der Aufnahme durch digitale Kameras, etwa bei Satellitenbildern. Es handelt sich um eine Matrix von Werten (siehe Abbildung 2.3).
Wenn diese Werte einzelnen Gitterpunkten zugeordnet sind, dann repräsentieren sie normalerweise eine gleichmäßige Iso-Fläche (z.B. Höhenlage, Temperatur, Niederschlagsmenge). Sind die Werte aber Gitterzellen (Flächeneinheiten) zugeordnet, so wird ein Bild (Satellitenbild, gescannte Karte) dargestellt. Wenn die Werte Kategorienummern sind, dann können mit Hilfe einer Datenbank ein oder mehrere Attribute diesen Kategorien zugeordnet werden. Die Gitterzellen sind in Reihen und Spalten geordnet. Auf diese Weise oder

mit Hilfe der geografischen Koordinaten kann auf die Daten zugegriffen werden.
Die Größe der Fläche, die eine Gitterzelle darstellt, wird durch die Auflösung bestimmt, d.h. durch die Seitenlänge der Zelle. Sie legt die Genauigkeit der Karte fest.
Die Gitterzellen einer 2D-Matrix werden Pixel genannt, aber auch 3D Daten können in manchen GIS durch räumliche Einheiten, die sogenannten Voxel, gespeichert werden.
Eingescannte Pläne, Luftbilder u.ä. müssen georeferenziert werden. Georeferenzierung bedeutet, dass einem Geodatenobjekt mitgeteilt wird, wo es sich auf der Erdoberfläche befindet. Erst dann ist es möglich geometrisch korrekte Informationen über Koordinaten, Strecken und Flächen zu erhalten. Dies geschieht durch Zuweisung von X,Y-Koordinaten zu einem Ort. Durch Georeferenzierung können die verschiedensten Daten in einem GIS zusammengeführt und verarbeitet werden.
Rasterdaten werden hauptsächlich für kontinuierlich im Raum verteilte Daten eingesetzt, wie z.B. Temperatur, Niederschlagsmenge oder Vegetation. Rasterdaten wurden entworfen mit Blick auf die Analyse von räumlichen Daten, der Modellierung solcher Daten und der Bildverarbeitung. Der Hauptvorteil ist die Einfachheit des Datenmanagement und der Algorithmen für die Analyse und Modellierung der Daten und für die Rechenoperationen, die mit diesen Daten durchgeführt werden können. Wie dies speziell bei GRASS ist, wird in Teil II dieser Arbeit beschrieben.

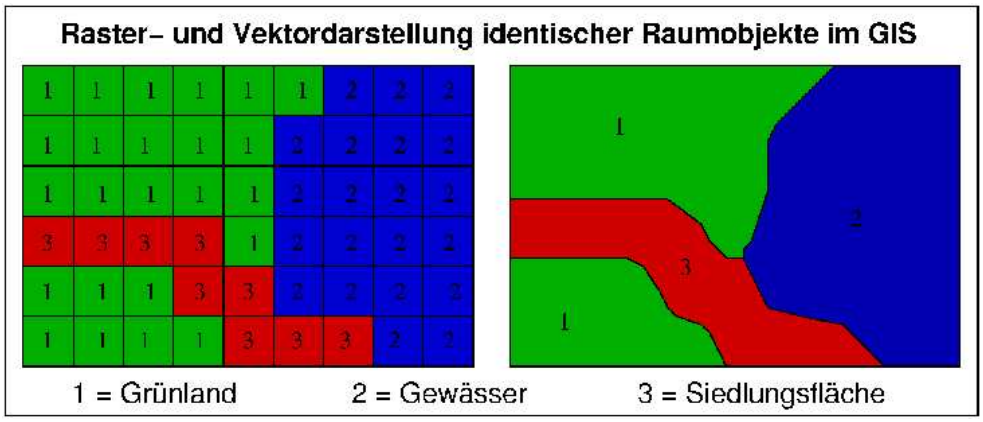

Abbildung 2.4: Vergleich von Raster- und Vektordatentypen auf identischer Fläche

2.1.2 Vektordaten

Die Vektordaten-Struktur ist notwendig zur objektbezogenen Bearbeitung von Daten. Nur mit Vektordaten können topologische Strukturen[5] und komplexe Datenmodelle aufgebaut werden. Zu jedem Objekt können dabei Attribute gespeichert werden, so dass man „intelligente“ Daten erhält. So weiß beispielsweise eine Linie, dass sie eine Wasserleitung ist, mit dem Durchmesser 150 mm aus Grauguss, verlegt im Jahr 1962 und sie kann mit anderen Leitungen, Schiebern, Ventilen, oder Ähnlichem topologisch verknüpft sein. Mit Vektordaten werden Linieninformationen und Flächeninformationen (geschlossene Linienzüge) gespeichert:

- Durch **Flächeninformationen** können Flächenobjekte gebildet werden, zu denen Sachdaten und Topologien gespeichert werden können. Im Unterschied zu graphischen Systemen bedeutet Topologie hier, dass eine Grenze nur ein Mal digitalisiert werden muss. Sie wird sowohl zur Objektbildung der linken wie auch der rechten Fläche verwendet. Der Grenze wiederum ist bekannt, welche Flächen sie begrenzt. Diese Informationen und Verknüpfungen werden bei dem Topologieaufbau von den meisten Systemen automatisch generiert.

- **Linieninformationen** werden auf der Basis der Graphen-Theorie mit dem Knoten-Kanten-Modell gespeichert. An allen Abzweigungen werden Knoten gesetzt. An den Knoten treffen die Kanten aufeinander. Über die Knoten sind die Kanten miteinander topologisch verbunden. Nicht an allen Kreuzungen müssen Knoten gesezt sein, d.h. es muss nicht an jedem Kreuzungspunkt eine Verbindung bestehen. Es kann z.B. eine Landstraße eine Autobahn ohne Verbindung überqueren oder Versorgungsleitungen verlaufen in unterschiedlicher Höhe und kreuzen sich. Zusätzlich ist die Richtung des Graphen von Bedeutung um die Fließrichtung von Wasser oder Verkehr etc. zu modellieren. Um die Anzahl der Punkte, die für komplexe Kurven benötigt werden, zu reduzieren, gibt es in manchen GIS mathematisch definierte Kurventeile oder Splines. Diese werden genutzt, um alle nötigen Punkte zum Zeitpunkt der Darstellung zu berechnen.

- **Punktinformationen** sind ein spezieller Fall des Vektordatenformats. In manchen GIS, wie z.B. GRASS, können sie auch als eigenes Format („site-Format“) eingesetzt werden. Punktdaten stellen sich als Menge von unabhängigen Punkten dar, die durch ihre Koordinaten definiert

[5] Nachbarschaftsbeziehungen

sind und zur Standortdarstellung bzw. als Verknüpfungspunkte von Vektorlinien eingesetzt werden.

Der Vorteil von Vektordaten liegt in der guten, maßstabsunabhängigen Darstellung, der verlustfreien Zoombarkeit und in der im Vergleich zu Rasterdaten sehr geringen Datenmenge. Dies liegt darin begründet, dass nur Knoten, Linien und Attribute gespeichert werden müssen. Außerdem können bei flächenhaften Informationen Isolinien zur Darstellung genutzt werden. Die Objekte im Vektorformat sind eindeutig, so dass objektbezogene Abfragen im GIS möglich sind.
Für kontinuierlich veränderliche Flächendaten, wie z.B. Bilddaten, ist das Vektorformat ungeeignet, da es aus homogenen Flächen besteht. Aus diesem Grund ist es auch für die räumliche Verteilungsanalyse nicht geeignet. Rasterdaten bieten dafür, wie im vorherigen Kapitel beschrieben, bessere Möglichkeiten.

2.1.3 Sachdaten

Sachdaten sind „thematische“ Daten, die sogenannten Attribute, und werden an die Koordinateninformationen geknüpft. Diese Daten werden in einer Datenbank abgelegt, die entweder GIS-intern sein kann oder extern über eine Schnittstelle an das GIS gekoppelt ist.
Deshalb beginnen die großen Datenbank-Hersteller (z.B. Oracle[6]) neue, räumliche Datentypen und Abfragemöglichkeiten (GeoSQL[7]) für ihre relationalen Systeme zu schaffen.
GRASS hat eine interne Datenbank, die aber nur ein Attribut pro Vektorobjekt oder Rasterdatenzelle verwalten kann [2].

2.1.4 Shapefiles

Shapefiles sind Vektordaten mit denen nicht-topologische geometrische Eigenschaften und zusätzliche Informationen von räumlichen Daten gespeichert werden. Die Geometrie wird durch Vektor-Koordinaten in Form von Punkten, Linien oder Flächen gespeichert [18].
Durch die Trennung von der topologischen Datenstruktur entstehen einige Vorteile:

- Die Darstellung der Daten ist schneller.

[6] http://otn.oracle.com/products/oracle9i/datasheets/spatial/spatial.html
[7] http://www.geosql.com

- Die Daten können bearbeitet werden.
- In der Regel wird weniger Speicherplatz verbraucht.

Shapefiles erhält man durch Export aus geeigneten Programmen oder durch Digitalisierung. Nach dem Import in ein GIS wird dort die Topologie aufgebaut.

2.2 Geographical Resources Analysis Support System (GRASS)

2.2.1 Entstehung

Das Geographical Resources Analysis Support System (GRASS) ist ein GIS mit einer weitreichenden Entstehungsgeschichte. Es wurde 1982 von der U.S. Army *(Corps of Engineers)/ CERL (Construction Engineering Research Lab)* für militärische Planungszwecke wie Landverwaltung und Umgebungsplanung entwickelt [1].
Ende der 80er Jahre wurde das Softwarepaket inklusive Quellcode veröffentlicht. Durch das Internet hat sich GRASS weltweit etabliert.
1995 zog CERL sich aus dem Projekt zurück und 1997 wurde das *GRASS Development Team* gegründet, das die Weiterentwicklung übernahm. Es besteht aus der Universität Hannover, der Baylor University in Texas, USA, dem Instituto Trentino di Cultura in Italien und weltweit weiteren Personen.
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.

2.2.2 Inhalt

GRASS wird kostenlos über das Internet verteilt, es ist ein „Open Source GIS“. Gemessen am Umfang des Codes ist GRASS das größte Free Software GIS und es zählt zu den zehn größten Open Source Projekten weltweit. GRASS unterliegt der GNU General Public License (GPL).
Neben den passenden Funktionen war der Open-Source-Charakter von GRASS ein Grund, es in dieser Arbeit einzusetzen.
GRASS läuft auf einer Vielzahl von UNIX-Plattformen, wie 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]. Allerdings ist die Version für Windows noch in der Experimentierphase, einige Funktionalitäten fehlen oder funktionieren nicht. Der

Quell-Code ist identisch zu der UNIX-Version, da der GRASS-Code plattformunabhängig ist.
Die Bedienung von GRASS erfolgt über die Kommandozeile. Genau genommen gibt es kein eigenes GRASS-Programm, sondern bei Aufruf wird ein Script aktiv, das eine neue Shell mit einigen zusätzlichen, GRASS-eigenen Umgebungsvariablen startet. Jeder GRASS-Befehl ist ein eigenes, vollständiges Linux-Programm, das die GRASS-API-Bibliotheken verwendet.
Für die gängigsten Operationen wurde eine grafische Benutzeroberfläche namens `tcltkgrass` erstellt (siehe Abbildung 2.5).

Abbildung 2.5: tcltkgrass

Die Programmierung von Erweiterungen von GRASS ist zum Einen durch Verwendung von Shell-Scripten möglich. Dies sind normale Shell-Scripte, die unter GRASS ausgeführt werden. Das größte Beispiel dafür ist `tcltkgrass`. Zum Anderen kann eine Erweiterung in C über die GRASS C-API programmiert werden. Dies geschieht mit Hilfe eines C-Programms, das einigen Restriktionen unterliegt. So muss neben der Einbindung der Header-Datei `gis.h` beachtet werden, dass die Grass-API einige Funktionen aus der C-Standardbibliothek ersetzt [3].
GRASS unterscheidet Raster- und Vektordaten. Die Analysefunktionen für Rasterdaten sind ausgereifter als die für Vektordaten, doch es ist eine Umwandlung zwischen Raster- und Vektordaten möglich. Die Umwandlung von Vektor- in Rasterdaten ist unproblematisch. Bei der Umwandlung von Raster- in Vektordaten können Fehler auftreten, die manuell korrigiert werden müssen. So kann es passieren das Linienzüge die eine Fläche darstellen nicht geschlossen sind, an Linienkreuzungen Knotenpunkte fehlen, etc..
Zur Darstellung der Layer, einzeln oder übereinander, werden sogenannte Monitore genutzt. Das sind spezielle Fenster, in denen GRASS sämtliche grafischen Ausgaben anzeigt.

Die Befehle in GRASS sind sehr klar strukturiert. Die Namen der GRASS-Programme beginnen alle mit einem speziellen Buchstaben. Dadurch wird gekennzeichnet um welche Art von Befehl es sich handelt (siehe Tabelle 2.1). Die Programme haben alle beschreibende Namen, so heißt das Modul um eine ASCII-Datei ins GRASS Rasterformat zu importieren `r.in.ascii`. Neben den in GRASS vorhandenen Modulen, stehen bei der Arbeit auch sämtliche

Präfix	Funktionsklasse	Bedeutung der Befehle
d.*	display	Für Grafikausgabe und visuelle Abfragen am Monitor
s.*	sites	Für Punktdatenverarbeitung
r.*	raster	Für Rasterdatenverarbeitung
i.*	imagery	Für Bildverarbeitung
v.*	vector	Für Vektordatenverarbeitung
g.*	general	Allgemeine Dateioperationsbefehle
m.*	misc	Verschiedene Befehle
p.*	paint	Kartenerstellungsbefehle
ps.*	postscript	Kartenerstellungsbefehle für Postscriptformat
db.*	database	Datenbankmanagement Module

Tabelle 2.1: Struktur der GRASS-Modulnamen

Unix/Linux Programme zur Verfügung, die über die Shell aufgerufen werden können. Dies ist besonders dann nützlich, wenn man an der Programmierung, Einbindung oder Modifikation eigener bzw. vorhandener GRASS Module interessiert ist.

2.2.3 Begriffe

In GRASS werden einige spezielle Begriffe verwendet:

Mapset bedeutet übersetzt ein „Satz Karten“. Es ist eine Sammlung von Karten. Jede GRASS-Sitzung läuft unter dem Namen eines Mapsets.

Location ist das betrachtete Gebiet (z.B. Osnabrück, Deutschland oder der Mars), der übergeordnete Arbeitsbereich, in dem ein oder mehrere Mapsets liegen.

Region ist im default-Fall die Location, kann während der Sitzung geändert werden.

Database ist die Datenbasis, die den gesamten geografischen Datenbestand enthält, also die Region, die Location und darin die Mapsets.

Die `Location` und das zu bearbeitende `Mapset` müssen zu Beginn jeder Sitzung angegeben werden. GRASS gruppiert seine Dateien zur Ordnung und Übersicht in Locations und Mapsets. Diese sind als normale Verzeichnisse angelegt. In dem Verzeichnis der `Database` liegt das Verzeichnis für die `Location`. Darin wiederum ist das `Mapset`-Verzeichnis gespeichert, in dem die Karten (bzw. Layer) in mehreren Unterverzeichnissen abgelegt sind.

Bei der Erstnutzung von GRASS muss einmalig vom Nutzer im Home-Verzeichnis ein Ordner eingerichtet werden, der üblicherweise `grassdata` genannt wird. Der Pfad dieses Ordners muss beim ersten Programmstart dem GIS mitgeteilt werden. Für jedes in GRASS definierte Projektgebiet wird automatisch eine `Location` in `grassdata` erstellt, in dem sämtliche Projektdaten gespeichert werden. Innerhalb der `Location` findet eine weitere Unterteilung in Mapsets statt.
Alle Dateioperationen (Kopieren, Löschen, Umbenennen) sollten mit den entsprechenden GRASS-Befehlen (`g.copy, g.remove, g.rename`) durchgeführt werden, da die unterschiedlichen Bestandteile (Geometrie-, Sach- und Grafikdaten) der einzelnen Layer in verschiedenen Unterverzeichnissen abgelegt werden (siehe Abbildung 2.6).

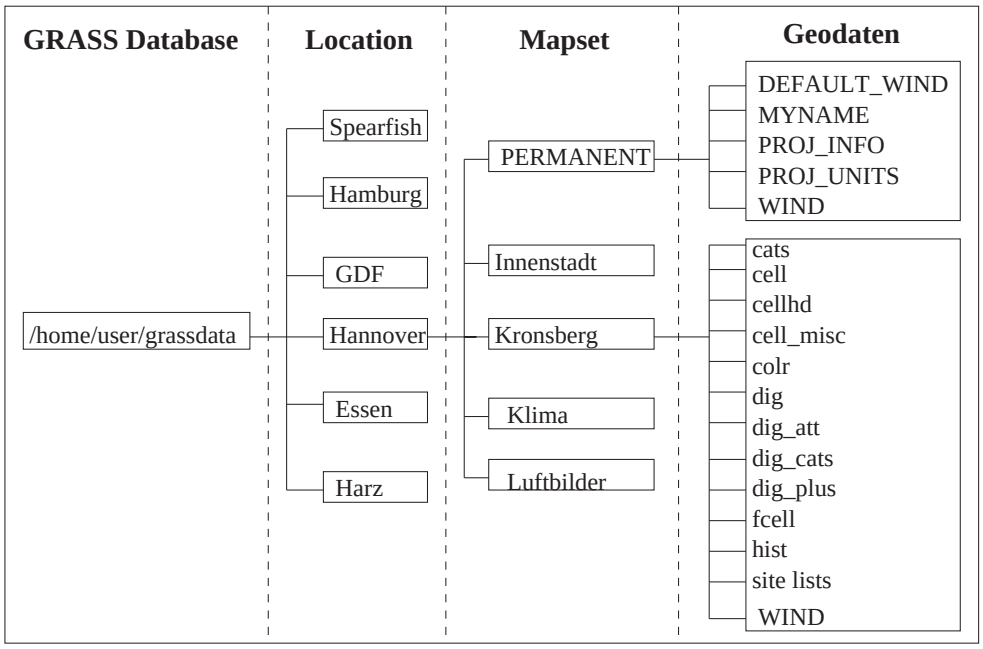

Abbildung 2.6: Verzeichnisstruktur in GRASS

2.2.4 Anlegen der Location

Beim ersten Arbeiten mit einer Location muss diese angelegt werden, bevor die relevanten Daten importiert werden. Das heißt, das oben angesprochene einheitliche Bezugssystem muss geschaffen werden. Dies geschieht durch die Angabe geografischer Informationen.
So wird das Koordinatensystem angegeben und die Randkoordinaten des Pro-

jektgebiets festgelegt. Das Koordinatensystem ist in Deutschland üblicherweise das Gauß-Krüger-Koordinatensystem (siehe Abbildung 2.7). Die Randkoordinaten werden aus topografischen Karten mit Gauß-Krüger-Koordinaten abgelesen [2].

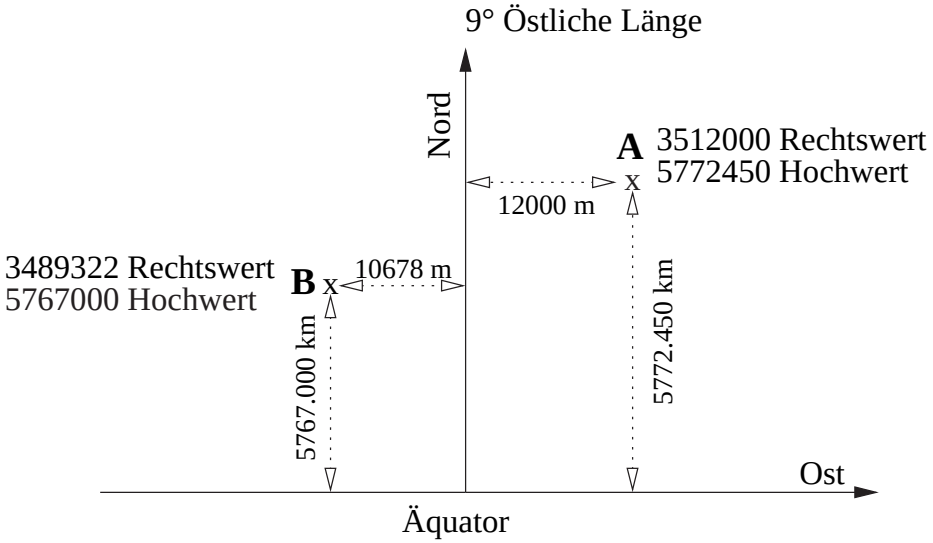

Abbildung 2.7: Das Gauß-Krüger-Koordinatensystem mit zwei Beispielpunkten A und B

Das Gauß-Krüger-System beruht auf der Forderung nach Winkeltreue. Als Bezugskörper wird hier nicht eine Kugel, sondern ein Rotationsellipsoid benutzt. Zur Kartendarstellung wird dieser Zylinder in die Ebene abgerollt. Das System entsteht, indem solch ein Zylinder, der senkrecht, *transversal*, auf der Erdachse steht, über den Erdkörper, hier: *Bessel-Ellipsoid* „gestülpt“ wird. Dadurch berührt der Ellipsoid den Zylinder in genau einem Meridian, den *Berührmedian*. Für die Kartendarstellung werden zwei schmale Streifen westlich und östlich dieses Meridians auf den Zylinder projiziert (siehe Abbildung 2.8). Sie erstrecken sich um je 2° auf der Erdoberfläche mit einer Verzerrung von maximal 12 cm auf 1 km am Streifenrand. Dies ist die Transversale Mercatorprojektion. Die Gauß-Krüger-Blattschnitträndер werden nicht parallel zum Papierrand abgebildet.

Als erste Abfrage beim Anlegen der Location muss aus einem Menü das richtige Projektionssystem ausgewählt werden. Da das Gauß-Krüger System nicht direkt in der Auswahlliste vorkommt, muss es unter `other` selbst definiert werden. Die durchzuführenden Schritte sind demnach:

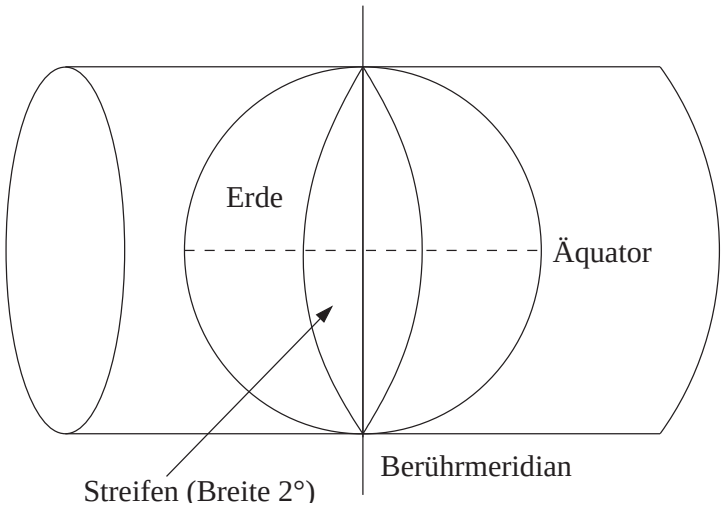

Abbildung 2.8: Die Transversale Mercatorprojektion

Für das Gauß-Krüger System wird angegeben:
`coordinate system for location: other (D)`

Danach folgt eine kurze Beschreibung des Projektgebiets:
`one line description for location: Hannover`

Es folgen nun weitere Angaben zur Projektion:

`specify projection name: tmerc` (Transverse Mercator)
`specify ellipsoid name: bessel` (Bessel Ellipsoid)
`Do you want to specify a map datum for this location? potsdam`
`Enter Central Parallel [lat_0] (23N): 0N` (Bezugsbreitenkreis)
`Enter Central Meridian [lon] (96W): 9E` (Bezugsmeridian)
`Enter Scale Factor at the Central Meridian: 1`
`Enter False Easting: 3500000` (3 wegen 9E als Central Meridian)
`Enter plural form of units: meters` (Einheit ist hier Meter)

Als *Map Datum* wird `potsdam` angegeben, da das *Potsdam Datum* im Bereich zwischen 6° und 15° West gilt [20]. Nachdem die Basisparameter der verwendeten Projektion angegeben wurden, werden die Randkoordinaten des Projektgebiets festgelegt. An die Gauß-Krüger Werte aus topographischen Karten werden dafür 3 Nullen angehängt (metergenaue Bestim-

mung). In derselben Eingabemaske folgt die Angabe zur Auflösung (Rasterzelle in Metern) in Ost-West und Nord-Süd Richtung. Dies ist die Standard-Rasterdatenauflösung (default) der erstellten Location. Auf die koordinatenscharf abgebildeten Punkt- und Vektordaten hat sie keinen Einfluss und sie kann während der Arbeit mit GRASS durch `g.region res=[Zahl]` jederzeit einfach verändert werden. Die verwendete Auflösung bei Rasterdaten hat großen Einfluss auf die benötigte Rechen- und Speicherkapazität, da eine größere Anzahl von Rasterzellen auch zusätzlichen Platz und erhöhten Rechenaufwand benötigen.

Kapitel 3

Fuzzy Logic

3.1 Entstehung

Fuzzy Logic bedeutet übersetzt „unscharfe Logik“ und ist die Theorie der unscharfen Mengen. In Japan wurden erstmals viele Produkte entwickelt die das Wort „Fuzzy“ in ihrem Namen tragen. Besonders bei Haushaltsgeräten ist diese Technik sehr beliebt. „Fuzzy“ steht hier im Grunde für dem Menschen angepasste Handhabung mit erweiterter Funktionalität.
Es wurde zum Beispiel die Waschmaschine „Aisaigo“ („Meine geliebte Frau“) entwickelt, die selbstständig Wassermenge und Waschprogramm an Hand der Wäschemenge und Verschmutzung dieser ermittelt. Fuzzy Logic kommt aber auch noch in vielen weiteren Gebieten zum Einsatz.
Eigentlich ist Fuzzy aber keine japanische Erfindung, sondern es wurde schon in den 60er Jahren von dem Elektrotechnikprofessor Lotfi Zadeh in Berkley, Kalifornien, entwickelt [17]. Da dort aber „Unschärfe“ mit „Ungenauigkeit“ gleich gesetzt wurde, dachte man diese Theorie sei mit Computern unvereinbar und sie stieß nicht auf großes öffentliches Interesse.
In den darauf folgenden Jahren wurde die Theorie hauptsächlich in Europa weiter entwickelt und als „flexible Logik“ oder „mehrwertige Logik“ bezeichnet.
Die Japaner erkannten das Potential, das darin steckte und unter Federführung des japanischen Ministeriums für Handel und Industrie (METI)[1] wurde das LIFE-Institut (Laboratory for International Fuzzy Engineering Research)[2] gegründet.
Der Erfolg von Fuzzy in Japan beruht wahrscheinlich darauf, dass die Betrachtung von Zwischentönen der Wahrheit mehr der asiatischen Philosophie

[1] http://www.meti.go.jp/english/
[2] http://www-2.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq-doc-18.html

entspricht als der hiesigen.

3.2 Inhalt

Fuzzy Logic wird bei Prozessen angewendet, für die eine eventuell unpräzise linguistische Modellierung geeigneter erscheint als klassische mathematische Modelle. Das Fuzzy-Konzept resultiert aus dem unterschiedlichen Symbolvorrat von Mensch und Computer [17].
Die dem menschlichen Denken zu Grunde liegenden Begriffe sind unscharf. Das heißt, es gib oft keine exakte Aussage über Eigenschaften. So sagt der Mensch z.B. „Dieser Turm ist sehr hoch.“ , dabei ist „hoch“ keine genaue Meterangabe.
Viele menschliche Entscheidungsprozesse basieren auf diesen unscharfen Angaben. Deshalb gibt es bei Fuzzy Logic keine scharf definierten Schwellen, sondern es wird als Entscheidungsgrundlage eine Funktion definiert, die den Zugehörigkeitsgrad zu einer Eigenschaft auf das Intervall [0,1] abbildet.
So ist ein Turm bei einer scharfen Relation (im Beispiel Abbildung 3.1: μ_N) sehr hoch (dargestellt als $\mu(m)$) wenn er größer ist als 99 m. Eine Höhe von 100 m ist also sehr hoch, wogegen aber 99 m nicht sehr hoch sind. Sinnvoller wäre hier ein stetiger Verlauf der Zugehörigkeitsfunktion, also eine unscharfe Relation (im Beispiel: μ_M).

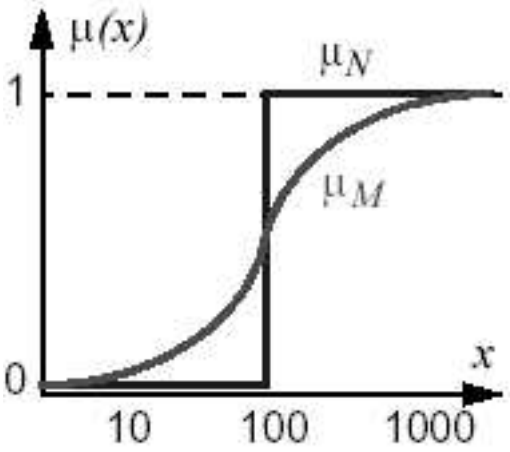

Abbildung 3.1: Kurvenverlauf bei scharfer und unscharfer Funktion

In diesem Beispiel ist $M = \{m | m \in \textit{sehr hoch}\}$ eine unscharfe Menge,

eine **Fuzzy Menge**. Eine Fuzzy Menge ist definiert durch eine Zugehörigkeitsfunktion die auf das geschlossenes Intervall [0,1] abbildet. Ist

$$M = \{(x; \mu_M(x)) | x \in X, \mu_M(x) \in \mathbf{R}\}$$

eine Fuzzy Menge, dann ist $\mu_M(x)$ ihre Zugehörigkeitsfunktion. Für diese Funktion gilt für alle $x \in X : \mu_M(x) \geq 0$ und $\mu_M(x)$ ist umso grösser, je besser x die unscharfe Aussage erfüllt.
Eine Fuzzy Menge hat nicht alle algebraischen Eigenschaften klassischer (scharfer) Mengen. Die Komplementarität muss nicht gelten:

Ist A eine unscharfe Menge, so können folgende Beziehungen eintreten:

$$A \cap A^c \neq \emptyset \; und \; A \cup A^c \neq \Omega$$

Wäre A die Menge *hoch* (schwarze Linie) und A^c die Menge *NICHT hoch* (rote Linen) in Abbildung 3.2, so ist der Schnitt der beiden Mengen der blau markierte Bereich und nicht die leere Menge. Die Vereinigung der Mengen ist nicht Ω, da der Teil, der grün markiert ist, fehlt.
Die Fuzzy Universalmenge Ω hat immer den Zugehörigkeitswert von 1 und die leere Fuzzy Menge $\emptyset$ hat immer einen Zugehörigkeitsgrad von 0 [15].

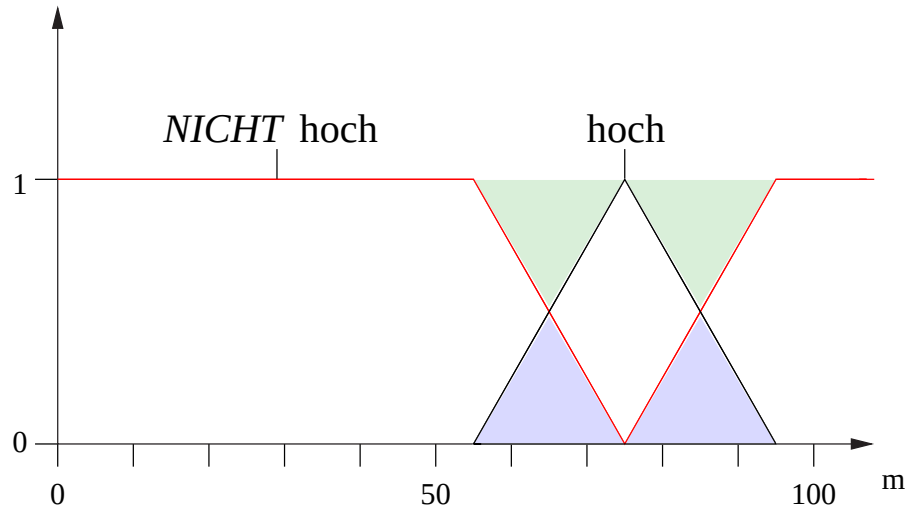

Abbildung 3.2: Die Komplementarität gilt nicht

Eine weit verbreitete alternative Darstellung von Fuzzy Mengen ist durch die sogenannten alpha-Schnitte gegeben. **Alpha-Schnitte** sind Schnitte

durch Fuzzy Mengen, die klassische (nicht-fuzzy) Mengen hervorbringen. In diesem Schnitt liegen alle Elemente der Menge, die größer gleich einer Grenze α für $0 < \alpha \leq 1$ sind.
Ein Alpha-Schnitt für festes α ist eine klassische Menge und kann dementsprechend mit Standardmethoden weiterverarbeitet oder dargestellt werden. In dieser Bachelorarbeit werden Alpha-Schnitte für ein geeignet gewähltes α als adäquate und dem Menschen eingängige Repräsentation der in der Gesamtfuzzymenge enthaltenen relevanten Information verwendet. Ist der Alpha-Schnitt für jedes α aus dem Intervall [0,1] gespeichert, dann bildet diese Menge von Mengen eine alternative äquivalente Darstellung der original Fuzzymenge.
Eine **Fuzzy Funktion** ist eine Abbildung, die Fuzzy Mengen auf Fuzzy Mengen abbildet. Solch eine Fuzzy Funktion kann z.B. über das sogenannte Extensionsprinzip aus klassischen Funktionen erhalten werden. Eine Funktion

$$h : X \to Y$$

für klassische Mengen X und Y induziert dann eine Funktion

$$H : F(X) \to F(Y)$$

von den Fuzzy Mengen über X (F(X)) in die Fuzzy Mengen über Y (F(Y)).
Eine weitere Art, mit Fuzzy Mengen zu rechnen ist durch die Fortsetzung klassischer Mengenoperationen und Erweiterungen gegeben. So kann man die Vereinigung von Fuzzy Mengen durch die Maximumbildung und den Schnitt durch das Minimum definieren.
Allgemeiner kann man Aggregations-Operatoren oder geeignete, problemangepasste Funktionen auf den die Fuzzy Mengen beschreibenden charakteristischen Funktionen definieren. In dieser Arbeit wird eine gewichtete Aggregation von Fuzzy Mengen benutzt (siehe Kapitel 8.1).

Kapitel 4

UMN Mapserver

Der University of Minnesota (UMN) Mapserver[1] ist eine Common Gateway Interface (CGI) basierte Anwendung, mit der Karten aus einem GIS in Form von *Shapefiles* dynamisch im WWW dargestellt werden können. Der Mapserver ist ein Open Source Projekt. Die Darstellung kann frei gestaltet werden, da die Mapserver Funktionen in ein HTML-Formular eingebunden werden. Für komplexere Applikationen ist es möglich, den Mapserver mit Hilfe von Java, JavaScript, PHP und anderen Internettechnologien zu erweitern [1]. Mögliche Betriebssysteme für den Mapserver sind GNU/Linux und andere Unix-Systeme sowie MS Windows und MacOS X.
Der Mapserver besteht aus:

- einem optionalen **Initialization File**, das eine Einführungsoberfläche anzeigt,
- einem **Mapfile**, das kontrolliert wie der Mapserver die Daten verarbeitet,
- einem **Template File**, das das Nutzer-Interface der Applikation im Browserfenster kontrolliert
- und einem **GIS Datensatz**.

Mit Hilfe der *gd-Library* von PHP werden die *Shapefiles* aus dem GIS Datensatz in png-Grafiken umgewandelt. Diese werden in einer temporären Datei abgespeichert, mit eindeutigen Namen versehen und von dort angezeigt.
Durch einen Link der im *Initialization File* oder einer anderen HTML-Seite

[1] http://mapserver.gis.umn.edu

gespeichert ist, wird der UMN-Mapserver gestartet. In diesem Verweis werden die benötigten Angaben zum *Mapfile*, das verwendet werden soll, zu den Layern, die angezeigt werden sollen, und zu der tmp-Datei, in der die Grafiken gespeichert werden sollen, gemacht (Beispiel: siehe Kapitel 9.1).
Im *Mapfile* wird unter Anderem die Struktur der Layer beschrieben und das *Template File*, das genutzt werden soll, wird angegeben.
Das *Template File* ist eine HTML-Datei, die den Rahmen für die angezeigten Layer bereit stellt.
Der Mapserver stellt die Möglichkeit zur Verfügung, eine Legende zu gestalten. In dieser kann der Nutzer angeben, welche Layer angezeigt werden sollen. Die angezeigten Layer werden wie in einem GIS übereinander gelegt und sie können einzeln an- und ausgeschaltet werden.
Weiterhin wird eine Zoomfunktion für die dargestellte Karte zur Verfügung gestellt. Nach jedem Zoomen und Ändern der Zusammenstellung der Layer wird eine neue png-Grafik erzeugt. Dabei wird immer der aktuelle Maßstab unter der Karte angezeigt.
Die Objekte in einem Layer können mit Attributen versehen werden, z.B. können den Straßen Straßennamen zugeordnet werden. Diese werden je nach Zoomtiefe eingeblendet. Vektorpunkte können mit Grafiken versehen werden, z.B. kann für Punkte, die Kirchen darstellen, ein Kirchensymbol in die Karte eingefügt werden. Beispiele für den Mapserver dieser Arbeit können im Anhang betrachtet werden.
Der UMN Mapserver ist allerdings nicht für einen Mehrbenutzerbetrieb angelegt, wenn Layer angezeigt werden sollen, die sich individuell verändern. Der Mapserver ist dafür konzipiert, Karten darzustellen, die für jeden darauf zugreifenden Nutzer dieselben Informationen enthalten, also gleich aussehen, wie z.B. Stadtpläne.
Für diese Arbeit ist der Mapserver in der Form nicht geeignet und wurde daher entsprechend angepasst (siehe Kapitel 9.1.1).

Teil II

Realisierung

Kapitel 5

Daten

5.1 Beschaffung von Geodaten

Die Beschaffung geografischer Vektordaten ist keine leichte Aufgabe, da, außer in den USA, behördlich erhobene Daten nicht frei verfügbar sind, sondern weitreichenden Restriktionen unterliegen, wie hohe Kosten für Vektordaten oder Datenschutz. Die Daten, die außerhalb von Behörden erhoben wurden sind kaum vorhanden oder werden teuer verkauft, da die Herstellung sehr aufwändig ist. So wurde als Grundlage für diese Arbeit anfangs versucht Vektordaten von der Stadt Berlin zu beschaffen, da Berlin auf Grund seiner Größe und seiner Struktur für dieses Projekt gut geeignet gewesen wäre. Allerdings wurde von der Senatsverwaltung für Stadtentwicklung mitgeteilt, dass für die benötigten Daten (ALK) für die gesamte Berliner Landesfläche ein Bereitstellungsentgelt in Höhe von ca. 190.000 Euro und ein Nutzungsentgelt in Höhe von ca. 290.000 Euro erhoben werden. Bei wissenschaftlicher Nutzung der ALK würde das Nutzungsentgelt nicht erhoben werden.
Die nun dieser Arbeit zu Grunde liegenden Daten stammen von einem Projekt namens *Frida*[1], das von der Intevation GmbH, ansässig in Osnabrück, ins Leben gerufen wurde. Bei diesem Projekt wurden im Rahmen eines Praktikums freie Vektor-Geodaten von der Stadt Osnabrück erstellt. Sie unterliegen der GNU/GPL Lizenz, das heißt, die Daten können wie freie Software behandelt werden.
Als Grundlage haben Orthofotos gedient, die von der Stadt Osnabrück zur Verfügung gestellt wurden. Orthofotos weisen einen einheitlichen Maßstab auf. Distanzen können wie auf Karten behandelt werden, da die Einflüsse von Kameraneigung und Gelände mittels einer Orthoentzerrung entfernt werden.

[1] http://frida.intevation.org

Anhand dieser Fotos wurden die Daten digitalisert und attributiert. Besonderen Wert wurde dabei auf die vollständige Digitalisierung der Straßen der Stadt Osnabrück gelegt. Des Weiteren wurden öffentliche Gebäude und andere interessante Objekte erfasst, sowie Gewässerlinien, Gewässer- und Grünflächen. Die Daten liegen nun in Form von *Shapefiles* vor (siehe Kapitel 2.1.4). Sie wurden ausschließlich mit Open-Source-Software erstellt. Zur Digitalisierung der Vektordaten wurde GRASS verwendet. Zur Verwendung der Daten in GRASS müssen diese durch ein geeignetes GRASS-Modul importiert werden. GRASS unterstützt in seinen Import-Modulen verschiedene Raster- und Vektorformate (siehe Tabelle 5.1).

GRASS-Modulkommando	Import Vektorformat
v.in.arc	ARC/INFO ungenerate
v.in.atlas	ATLAS GIS Vektor
v.in.gshhs	Global Self-consistent Hierarchical High-resolution Shoreline (GSHHS)
v.in.tig.basic	U.S. Census Bureau TIGER files
v.in.tig.lndmk	U.S. Census TIGER Landmark features
v.in.dxf	AUTOCAD/DXF
v.in.dxf3d	AUTOCAD/DXF3D
v.in.dlg	ASCII USGS DLG-3
v.in.shape	ESRI/SHAPE
v.in.sdts	SDTS DEM (USGS)
v.in.mif	MapInfo/MIF
v.in.e00	ESRI/e00
v.in.ascii	ASCII

Tabelle 5.1: GRASS Module für den Import von Vektordaten

Das hier vorliegende Shape-Format ist das am häufigsten genutzte Format. Es ist kein topologisches Format, da es keine Lage- bzw. Nachbarschaftsbeziehungen zwischen den einzelnen Objekten gibt. Grenzlinien zwischen zwei Flächen (Polygonen) werden beispielsweise doppelt gespeichert [21].
Daher muss nach dem Import in GRASS mit Hilfe des Programms `v.support` für jede Karte die Topologie aufgebaut werden.
Damit diese richtig aufgebaut werden kann muss eine Location mit den richtigen Angaben (siehe Tabelle 5.2) angelegt werden (wie in Kapitel 2.2.4).
Das Modul, um Shape-Daten nach GRASS zu importieren heisst `v.in.shape`. Da in GRASS nur ein Attribut und ein Label pro Vektor unterstützt wird, muss vor dem Import bei Vektorkarten mit mehreren Attributspalten die ge-

name	Transverse Mercator
datum	potsdam
dx	606.000000
dy	23.000000
dz	413.000000
proj	tmerc
ellps	bessel
a	6377397.1550000003
es	0.0066743722
f	299.1528128000
lat_0	0.0000000000
lon_0	9.0000000000
k_0	1.0000000000
x_0	3500000.0000000000
y_0	0.0000000000
unit	meter

Tabelle 5.2: Projektionsdaten (gemäß GRASS-Kommando g.projinfo)

wünschte Spalte aus der Datenbanktabelle ausgewählt werden. Dazu ist es notwendig einen Blick in die Datentabelle der Shape-Datei zu werfen, um zu bestimmen, welche Attributspalte aus der DBF Tabelle als Attribut und welche als Label der Vektorkarte in GRASS gespeichert werden soll. Dazu kann das Flag `-d` genutzt werden:

```
v.in.shape -d input=gruenflaechen-joined.shp

Attribute fields available in gruenflaechen-joined.shp:
1: gfShapeID [int4:4]
2: gfTypID [int4:2]
3: gfName [text:50]
4: gfTypName [text:50]
```

Anhand der Ausgabe ist zu erkennen, dass der gewählten Shape-Datei eine Attributtabelle mit vier Spalten angehängt ist. Zwei der Spalten beinhalten ganzzahlige Werte (`gfShapeID` und `gfTypID`), die anderen bestehen aus einem Textfeld (`gfName` und `gfTypName`). Es ist möglich sich den Inhalt der Attributtabelle mit einem entsprechenden Editor anzuschauen. Unter GNU/Linux kann dies Staroffice, Openoffice, Gnumeric oder Koffice sein.
Nun wird z.B. `gfShapeID`, als Label und die Spalte `gfTypName` gewählt

und der Befehl ausgeführt, um die Shape-Datei `gruenflaechen-joined.shp` nach GRASS zu importieren:

```
v.in.shape input=gruenflaechen-joined.shp out=gruenflaechen
attribute=gfShapeID label=gfTypName
```

Nach dem Import werden die Daten im *GRASS Binary Vektorformat* gespeichert.

Kapitel 6

Digitalisierung

Im Zuge dieser Arbeit wurden durch Digitalisierung zusätzliche Daten erfasst, um zu zeigen, dass die Möglichkeit besteht, zusätzliche Kriterien zur Wohnortsuche einzufügen. Verwendet wurde dafür das GRASS-Programm *Digitize* (gestartet mit `v.digit`).
Wenn kein Digitalisierbrett zur Verfügung steht, kann mit Hilfe dieses Programmes eine eingescannte georeferenzierte (Raster-) Karte als „Schablone“ genutzt werden. Dabei wird als Ersatz für das Digitalisierbrett die Maus eingesetzt und die topografische Karte in den Hintergrund gelegt, um sie als Digitalisiergrundlage zu verwenden. Je nach Digitalisierthema werden nun manuell an der gewünschten Stelle Punkte, Linien und Flächen digitalisiert (siehe Abbildung 6.1). Die Farben der Objekte können frei gewählt werden. Ebenso kann ein erklärender Attributtext hinzugefügt werden, allerdings nicht direkt in `v.digit`, sondern nach Beendigung der Digitalisierung, mit Hilfe des Moduls `v.support`.
Durch die Zoomfunktion ist sogar eine genauere Digitalisierung als mit einem Digitalisierbrett möglich, da bei der Arbeit mit einem Digitalisierbrett nicht in die Vorlage hinein gezommt werden kann. Unabhängig jedoch von der Art der Kartenvektorisierung gibt es ein paar grundlegende Regeln, auf die wegen der in Karten angewendeten maßstabsabhängigen Generalisierung geachtet werden muss. Vektorlinien müssen entlang der Mitte der linienhaften Information digitalisiert werden (z.B.: Straßenmitte). Bei der Digitalisierung von Flächen müssen die Vektorlinien entlang der Flächengrenzlinie gezogen werden. Eine Fläche ist erst dann korrekt digitalisiert, wenn der Linienzug geschlossen ist und in die Mitte der Fläche ein Labelpunkt gesetzt wurde. Bei Gebäuden ist abgesehen von Katasterkarten nur der Mittelpunkt des Gebäudes lagetreu. Um später die topologischen Funktionen der Vektorkarte nutzen zu können, sollten

- Linien sich im Regelfall nicht ohne Knotenpunkt kreuzen.
- Linien, die einen gemeinsamen Knoten benutzen, diesen exakt treffen.
- Flächenpolygone geschlossen sein. Nicht korrekt digitalisierte Polygone lassen sich anschließend nicht labeln, da keine eindeutige Topologie aufgebaut werden kann.
- zwischen benachbarten Flächen als Grenzlinie immer einfache Linien digitalisiert werden. GRASS ordnet diese Grenzlinien gleichzeitig beiden Flächen zu.

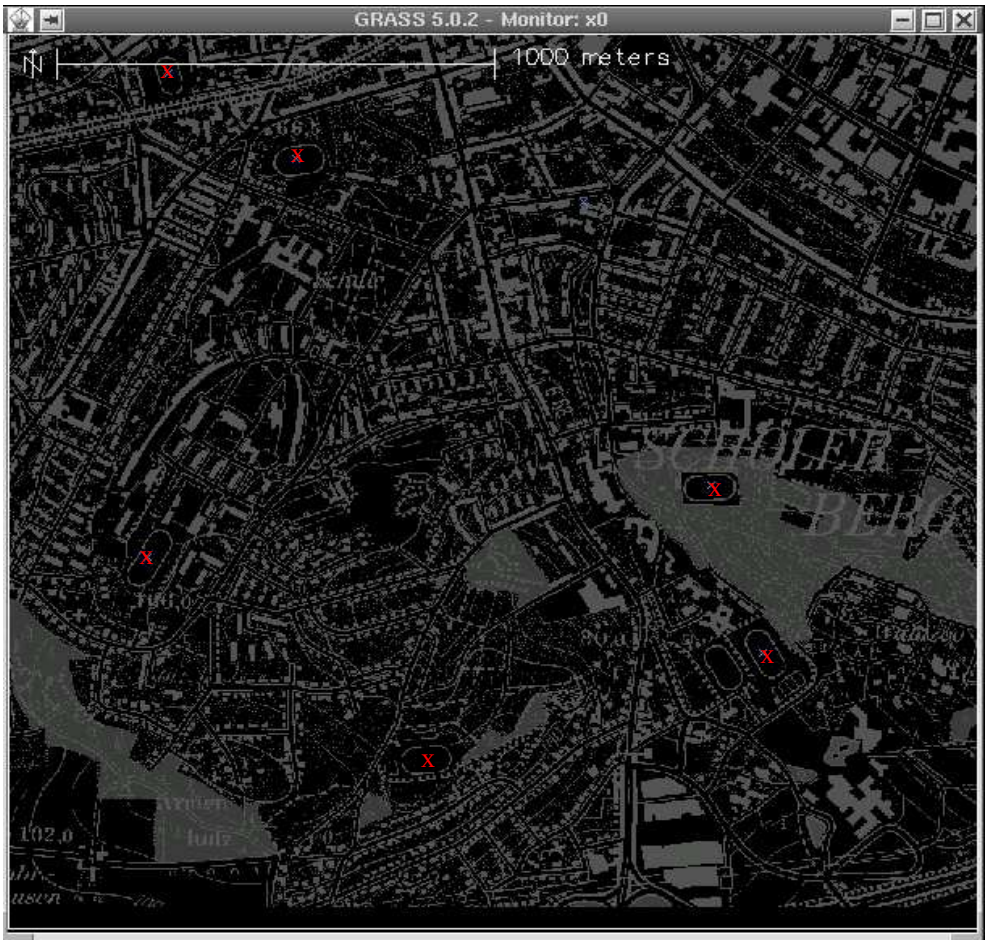

Abbildung 6.1: Das Digitalisieren von Punkten in GRASS mit Hilfe von topografischer Karte als Grundlage

Nachdem ein GRASS-Monitor und das Modul `v.digit` gestartet wurde, muss ein Digitalisierbrett ausgewählt werden. Wenn keines zur Verfügung

steht und mit der Maus gearbeitet werden soll, wird `none` ausgewählt. Nach Auswahl einer neuen oder bereits bestehenden Vektordatei erscheint ein Eingabeformular, in dem optional Angaben zu Person, Datum usw. zu machen sind. Verpfichtend sind Angaben zum Kartenmaßstab.
Nach Bestätigung dieser Angaben im Eingabeformular gelangt man zu einem Menü (siehe Abbildung 6.2). Um die topografische Karte, die als Digitalisiergrundlage verwendet werden soll, in den Hintergrund zu legen, wird unter dem Menüpunkt `Customize` der Punkt `Backdrop cell map` ausgewählt. Danach kann das eigentliche Digitalisiermenü unter `Digitize` aufgerufen werden. Dort kann der zu digitalisierende Typ mit der Taste `t` für `Line, Area, Site` gewählt werden. Punkte lassen sich sowohl als Vektorpunkte als auch im eigenen Punktformat in GRASS speichern. Das Modul `v.digit` ist im Folgenden selbsterklärend.

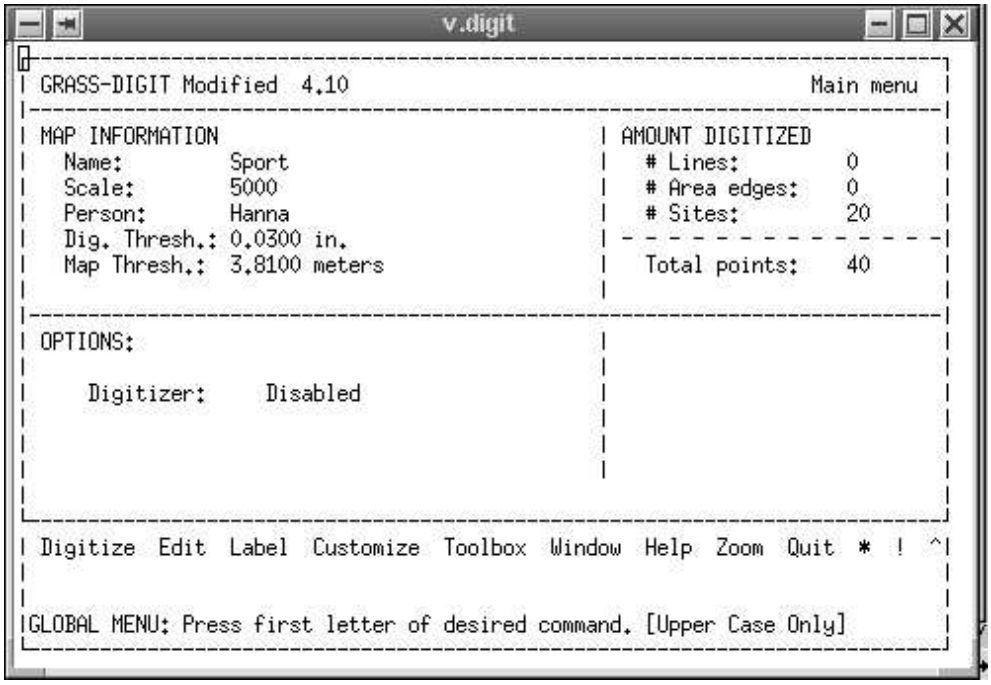

Abbildung 6.2: Das Digitalisiermenü von GRASS

Kapitel 7

Distanzen

Zur Bestimmung eines optimalen Wohnorts müssen die Distanzen zu den nächstgelegenen relevanten Objekten bestimmt und abgespeichert werden. Dazu wurde einmalig zu jedem Wohn-Kriterium eine Distanzkarte berechnet (siehe Abbildung 7.1). In dieser Karte wird in Rasterzellen der Abstand zum nächstgelegenen Objekt gespeichert. Es wird also ein Layer geschaffen der aus Zellen besteht, denen Werte zugeordnet sind. Diese Werte sind die Distanzen angegeben als Anzahl der Zellen bis zum nächsten Objekt.
So werden in der Distanzkarte `gruenflaecheDistanz` die Abstände zur nächstgelegenen Grünfläche, von jedem Punkt (jeder Zelle) auf der Karte aus, gespeichert.
In der jetzigen GRASS-Version sind noch keine Werkzeuge zur Vektornetzwerk-Analyse realisiert (aus den in Kapitel 2.1 erwähnten Gründen), daher müssen die Kostenberechnungen auf der Basis von Rasterdaten umgesetzt werden.
Die Berechnung von kumulativen Kosten bei der Bewegung von einer Zelle einer Rasterkarte zur nächsten Zelle wird auf Basis einer Kostenfläche realisiert. Eine Kostenfläche ist eine Rasterkarte, die die kumulierten Kosten enthält, die bei der Bewegung zwischen verschiedenen Punkten auf einer bestimmten Rasterkarte entstehen.
Jede Zelle einer solchen Rasterkarte enthält einen Wert, der die Kosten darstellt, die aufgewendet werden müssen, um diese Zelle zu durchqueren. Solche Kosten können bei einer Straßenkarte z.B. „die Reisezeit abhängig von der Straßenqualität“ sein. Das GRASS-Modul `r.cost` produziert dann solch eine Rasterkarte, in der jede Zelle die gringsten Kosten enthält um bestimmte vom Nutzer spezifizierte Punkte zu erreichen. Die errechnete Karte `gruenflaecheDistanz` ist eine Kostenfläche. Um Kostenflächen von jedem vorhanden Layer zu erhalten, müssen die Vektorkarten von Frida in Rasterkarten umgewandelt werden:

```
g.region res=50 -a
v.to.rast gruenflaechen.2 out=gruenflaechen.2.50m
```

Dabei wird die Auflösung mit Hilfe von `g.region` auf 50 Meter pro Rasterzelle gesetzt. Diese Auflösung ist zur Wohnortsuche gut geeignet. Einzelne Gründstücke sind normalerweise nicht kleiner als 50 Meter und die Punkte von denen die Wohnortsuche abhängig ist (wie Grünflächen, Sportstätten etc.) liegen nicht näher als 50 Meter beieinander. Bei einer höheren Auflösung, also einer höheren Genauigkeit würde der Speicherplatzbedarf für die Rasterkarten exponentiell ansteigen und bei einer geringeren Auflösung hätte das Ergebnis der Wohnortsuche eine geringere Qualität.
Das Flag -a sorgt dafür, dass die Region an die Auflösung angepasst wird, also die Randkoordinaten passend der Auflösung gewählt werden.
Um die kürzeste Distanz von allen Rasterzellen zu einer Rasterlinie zu berechnen, kann die Kostenfläche mit einem Kostenwert von 1 genutzt werden. Da Grünflächen als Flächen und nicht als Linien gespeichert sind, muss die Karte vor der Umwandlung in eine Rasterkarte mit Hilfe von `v.area2line` bearbeitet werden (Ergebnis: `gruenflaechen.2`).
Die Berechnung mit r.cost wird wie folgt ausgeführt:

```
g.region rast=gruenflaechen.2.50m
r.mapcalc area.one=1
r.cost input=area.one output=gruenflaechen.2Distanz start_rast=
gruenflaechen.2.50m
```

Die Karte `area.one` wird als Information über die Kosten genutzt. Der Kostenwert ist 1, da Entfernungen berechnet werden sollen, die Entfernungen von jeder Rasterzelle zur nächsten Grünflächen. Die resultierende Distanzkarte `gruenflaechen.2Distanz` beinhaltet die Entfernungen als Anzahl der Zellen zur nächsten Grünfläche. Um eine Distanzkarte zu berechnen, die die Entfernungen in Metern enthält, müssen die Werte in den einzelnen Zellen mit der Auflösung multipliziert werden. Dies ist hier jedoch nicht notwendig, da das Ergebnis grafisch dargestellt wird.
Die Werte in den einzelnen Zellen können in GRASS angezeigt werden, durch `d.rast.num`. Dafür muss allerdings weit genug in die betreffende Karte hinein gezoomt werden.

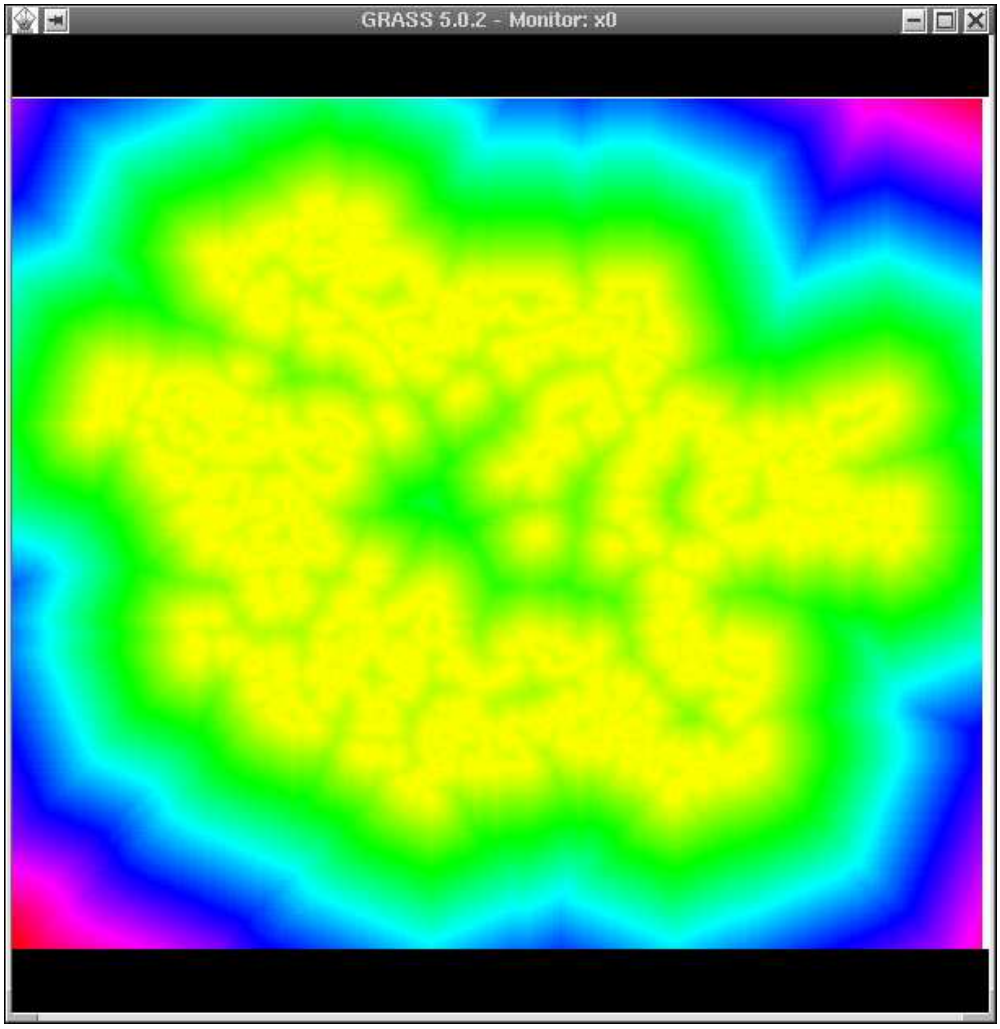

Abbildung 7.1: Die Distanzkarte der Grünflächen. Der gelbe Bereich ist besonders nah an den Grünflächen.

Kapitel 8

Berechnung des Wohngebiets

Die eigentliche Berechnung des geeigneten Wohngebiets, die nach der Anfrage des Nutzers ausgeführt wird, wird durch ein Shell-Script durchgeführt. GRASS wird nicht durch das normale GRASS-Startscript ausgeführt, wie es bei manueller Nutzung üblich ist, sondern die relevanten Umgebungsvariablen werden in dem Script zur Wohngebiet-Berechnung gesetzt. Es werden nur diejenigen Variablen gesetzt, die gebraucht werden um die benötigten GRASS-Programme auszuführen.
Dies ist durch den modularen Charakter von GRASS möglich. Wie bereits in Kapitel 2.2 erläutert läuft kein Programm ab, sondern eine Sammlung von Modulen, die in einer speziellen Umgebung laufen. Diese Struktur erlaubt die Kontrolle von außen durch Scripte. Dabei sind in den Scripten mindestens diese Variablen zu setzten:

```
echo ,,LOCATION_NAME:osnabrueck‘‘              >$HOME/.grassrc5
echo ,,MAPSET:user1‘‘                         > >$HOME/.grassrc5
echo ,,DIGITIZER:none‘‘                       > >$HOME/.grassrc5
echo ,,GISDBASE:/usr/local/share/grassdata‘‘  > >$HOME/.grassrc5
echo ,,GRASS_GUI:text‘‘                       > >$HOME/.grassrc5

export GISBASE=/usr/local/grass5
export GISRC=$HOME/.grassrc5
export PATH=$PATH:$GISBASE/bin:$GISBASE/scripts
```

Mit diesen Variablen wird GRASS der Name der Location und des Mapsets und der Pfad zum GRASS-Verzeichnis mitgeteilt, indem sie in die Datei `.grassrc5` geschrieben werden. So werden die Angaben, die beim Start von GRASS mit dem Startscript in einem Formular gemacht werden müssen, manuell angegeben.
Nach dem Setzen dieser Variablen ist die Umgebung definiert und alle

GRASS-Module können genutzt werden. Mit Hilfe der Distanzkarten (siehe Kapitel 7) wird nun an Hand der vom Nutzer angegebenen Kriterien und Prioritäten die Karte erzeugt, die den geeigneten Wohnort enthält. Dafür muss die Auflösung der Region, in der die Berechnungen durchgeführt werden, der Auflösung der zu bearbeitenden Karten angepasst werden:

```
g.region res=50
```

Dies ist wichtig, da der Arbeitsspeicher überlastet wird, wenn die Auflösungen nicht übereinstimmen. Nun können die eigentlichen Berechnungen folgen:

```
r.mapcalc < < MARK
ergebnis=gruenflachen*($1*1.0)+gewaesser*($2*1.0)
        +fluesse*($3*1.0)+sport*($4*1.0)+poi*($5*1.0)
ergebnis=ergebnis/(($1+$2+$3+$4+$5)*1.0)
ausgabe=if((ergebnis-$7),1,1,0)
end
MARK
```

Die Variablen $1 bis $5 werden beim Aufruf des Scripts durch die PHP-Funktion `shell_exec()` übergeben. Dies sind die Prioritäten die der Nutzer in ein HTML-Formular eingegeben hat und die beschreiben, wie wichtig dem Nutzer die jeweiligen Kriterien zur Wohnortsuche sind.
Die Kartenalgebra, die für die Berechnungen nötig ist, kann mit dem GRASS-Programm `r.mapcalc` durchgeführt werden. Mit `r.mapcalc` können neue Rasterkarten-Layer unter Einbezug von existierenden Rasterkarten, Integer- oder Fließkommazahlen und Funktionen kreiert werden. Prinzipiell wird auf folgende Art gerechnet:
`neuekarte = Wert/karte1 <formel> [karte2, ...]`.
Nach Angabe dieser Rechnung (unter Benutzung der Dateinamen der verwendeten Karten) wird zellenweise die neue Karte nach der angegebenen Formel berechnet (siehe Abbildung 8.1).
Bei `r.mapcalc` gilt (siehe Tabelle 8.2):

$$if(x, a, b, c) : a \; wenn \; x > 0, \; b \; wenn \; x \; gleich \; 0, \; c \; wenn \; x < 0.$$

Möchte man bei `r.mapcalc` eine Ergebniskarte im Fließkommaformat erstellen, so muss ganzzahligen Werten in der Rechnung ein Dezimalpunkt angehängt werden (z.B. 10. statt 10) und soll eine Integerkarte in eine Fließkommakarte umgewandelt werden, muss sie mit 1.0 multipliziert werden. Als Ergebnis der oben angeführten Berechnungen erhält man einen eigenen

Layer, der das Wohngebiet enthält. Dieser liegt in Form von Rasterdaten vor. Die Vorgehensweise innerhalb dieser Berechnungen wird im nächsten Kapitel näher erläutert.

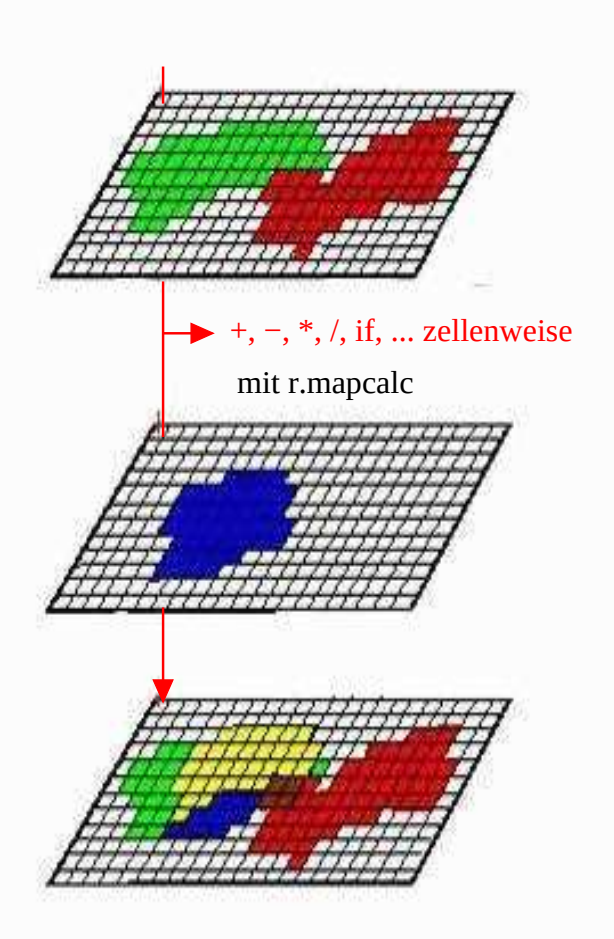

Abbildung 8.1: Arbeitsweise von r.mapcalc

8.1 Fuzzy Logic Funktion

Entsprechend der Fuzzy Logic Theorie werden die Rasterzellen in der Distanzkarte (aus Kapitel 7), in denen die Abstände zum nächsten relevanten Objekt gespeichert sind, als Fuzzy-Menge modelliert. So werden durch die Funktion

$$F_i(x) = 1 - \frac{x}{max.Entfernung}$$

%	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
#	Operator zur Farbtrennung in R, G, B

Tabelle 8.1: Operatoren in r.mapcalc

abs(x)mode()	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
float(x)	wandelt x in Fließkommazahl
if	Entscheidungsoperator
if(x)	1, wenn x nicht 0, 0 sonst
if(x,a)	a, wenn x nicht 0, 0 sonst
if(x,a,b)	a, wenn x nicht 0, b sonst
if(x,a,b,c)	a, wenn x > 0, b wenn x gleich 0, c wenn x < 0
isnull()	1, wenn x gleich NULL (no data)
log(x)	natürlicher log von x
log(x,b)	log von x zur Basis b
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)

Tabelle 8.2: Funktionen in r.mapcalc

die Werte in den Rasterzellen (x) der Distanzkarte (i=1,...,5) zu Werten aus [0,1] umgewandelt. Dies ist nötig, da Fuzzy Mengen durch Zugehörigkeitsfunktionen definiert werden, die auf [0,1] abbilden, wobei 1 das Optimum (in diesem Fall der Entfernung) und 0 die schlechteste Wahl ist. Die maximale Entfernung auf einer Karte kann mit Hilfe von g.region ermittelt werden. Im Hauptmenü werden u.a. die Anzahl der Spalten und Zeilen der Karten in der aktuellen Region angegeben. Die größere der beiden Zahlen ist der maximale Abstand innerhalb dieser Karte.
Mit dem GRASS-Modul `r.mapcalc` werden die Distanzen der Kriterien mit den ausgewählten Prioritäten gewichtet. Die Prioritäten (p_i) können Werte zwischen 0 und 5 annehmen. Ist das Kriterium für den Nutzer äußerst wichtig, so wählt er eine Priorität von 5 und ist das Kriterium nicht von Bedeutung, so wählt er 0.
Die Gesamtfuzzymenge ergibt sich durch Aggregation und anschließender Normierung der gewichteten Kriterien. Die Normierung gewährleistet, dass die Funktion zur Bildung der Gesamtmenge wieder auf [0,1] abbildet. Die Berechnung:

```
ergebnis=gruenflachen*($1*1.0)+gewaesser*($2*1.0)
+fluesse*($3*1.0)+sport*($4*1.0)+poi*($5*1.0)
ergebnis=ergebnis/(($1+$2+$3+$4+$5)*1.0)
```

entspricht also der Fuzzy-Funktion:

$$H(F_i(x)) = \frac{\sum_{i=1}^{n} p_i F_i(x)}{\sum_{i=1}^{n} p_i},$$

wobei n der Anzahl der auszuwählenden Kriterien entspricht.
Durch einen alpha-Schnitt erhält man eine klassische Menge, die die wesentliche Information der Fuzzy Menge darstellt, nämlich die Menge aller mindestens zum Grad α zugehörigen Werte:

```
ausgabe=if((ergebnis-$7),1,1,0).
```

Der α-Schnitt kann je nach der gewünschten Genauigkeit gewählt werden. Bei α=0,9 z.B. werden alle Rasterzellen in der erstellten binären Karte, die größer gleich 0,9 sind mit 1 gekennzeichnet. Diese Karte beschreibt nun an den mit 1 gekennzeichneten Stellen das optimale Wohngebiet.
Bei obiger gewichteter Aggregation ist auf Grund der Normierung die absolute Größe der Gewichtungsfaktoren p_i irrelevant und nur die relative Beziehung untereinander relevant.
Dieses ist adäquat, wenn man die Fuzzy Mengen untereinander verrechnen

möchte, für eine einheitliche Darstellung des Ergebnisses allerdings eventuell ungeeignet. Wenn z.B. Flüsse mit Priorität 5 ausgewählt werden (der Rest der Kriterien sei Null) ergibt sich das Selbe, wie bei Priorität 1 für Flüsse. Intuitiv würde man allerdings im letzteren Fall ein größeres mögliches Wohngebiet erwarten als im ersten Fall (siehe Abbildungen A.2 und A.3).
Dieses Verhalten kann leicht durch geeignete Wahl der alpha-Schnitte ermöglicht werden: im letzteren Fall werden auch Punkte zurückgegeben, die einen geringeren Zugehörigkeitsgrad zur Gesamtfuzzymenge haben, als im ersteren Fall. Das heißt das α, welches den Mindestzugehörigkeitsgrad angibt, wird entsprechend der höchsten Priorität gewählt.
Als geeignet hat sich folgende Zuordnung erwiesen:

Priorität	Schranke
1	0.975
2	0.98
3	0.985
4	0.99
5	0.995

Die höchste Priorität wird in der Datei *anzeige.php*, die nach dem Ausfüllen des Abfrage-Formulars aufgerufen wird, durch die php-Funktion `max()` ermittelt. Die entsprechende Schranke wird in der Variablen `$alphacut` an das Script übergeben und dort als Schranke für den α-Schnitt eingesetzt.

```
$maximum=max($_POST[gew1], $_POST[gew2], $_POST[gew3],
$_POST[gew4], $_POST[gew5]);
$alphacut=0.97+$maximum*0.005;
```

So wird erreicht, dass die vom Nutzer angegebenen Prioritäten Einfluss auf die Größe des berechneten Wohngebiets haben.

Kapitel 9

Darstellung

Die Ausgabe des Wohngebietes ist in eine Webapplikation eingebunden, die mit HTML und PHP erstellt wurde. Nachdem die Kriterien mit den zugehörigen Prioritäten ausgewählt wurden und der „Absenden“-Button aktiviert worden ist, wird das Script zur Berechnung des Wohngebiets mit der PHP-Funktion `shell_exec()` gestartet, die die Parameter aus dem vom Nutzer ausgefüllten Formular, eine Variable, die die Schranke für den α-Schnitt (siehe Kapitel 8.1) und einen eindeutigen Dateinamen (siehe Kapitel 9.1.1) übergibt.

```
shell_exec(,,sh wohngebiet $_POST[gew1] $_POST[gew2] $_POST[gew3]
$_POST[gew4] $_POST[gew5] $filename $alphacut“);
```

Nun besteht die Möglichkeit, die Ausgabe mit Hilfe von Scaleable Vector Grafics (SVG) oder durch einen Mapserver umzusetzten. Der Vorteil bei der Darstellung mit SVG besteht darin, dass die Karte mit dem Wohngebiet mit Hilfe von PHP ohne Zwischenspeicherung dargestellt werden könnte. Dies führt dazu, dass die Applikation schneller ist und weniger Speicherplatz gebraucht wird.
Andererseits gibt es aber noch kein ausgereiftes kostenfreies Programm zur Umwandlung von *Shapefiles* nach SVG für freie GIS. Deshalb wird die Ergebniskarte in dieser Arbeit durch einen Mapserver dargestellt.

9.1 UMN Mapserver

Für diese Arbeit wurde der frei verfügbare UMN-Mapserver genutzt [22], der im Projekt *Frida* der Intevation GmbH bereits vorkonfiguriert wurde. Dieser Mapserver wurde für dieses Projekt entsprechend angepasst. Die Layer für

das Wohngebiet und die Sportstätten wurden hinzugefügt und das *Template File* angepasst. Der Layer für das Wohngebiet wird im *Mapfile* folgendermaßen definiert:

```
#1.Layer WOHNGEBIET
LAYER
  NAME ,,Wohngebiet‘‘
  DATA ,,ergebnis‘‘
  TYPE POLYGON
   CLASS
    NAME ,,Ergebnis‘‘
    COLOR 255 30 10
   END
  STATUS ON
END
```

Im *Mapfile* wird angegeben wo, die *Shapefiles* aus dem GIS Datensatz gespeichert sind. Dort ist ebenfalls das *Shapefile* für das Wohngebiet unter dem Namen `ergebnis` gespeichert. Der Inhalt des *Shapefiles* besteht aus Polygonen und die Farbe, in der diese dargestellt werden sollen, ist Rot. Der Status ist `ON`, da der Layer beim ersten Aufruf des Mapservers mit diesem *Mapfile* angezeigt werden soll.
Im *Template File* mussten der Legende das Wohngebiet und die Sportstätten hinzugefügt werden, sowie das Aussehen der Seite an den Rest des Projekts angepasst werden (siehe Abbildungen A.1 und A.2). Der Verweis zum Aufruf des Mapservers, der gestartet wird, nachdem das Script *wohngebiet* ausgeführt wurde, lautet:

```
http://snowball.informatik.uni-osnabrueck.de/cgi-bin/mapserv?
map=../hkraeuse/umn-mapserver/html/osnabrueck.map
&layers=sonststrassen+nebenstrassen+hauptstrassen+bundestrasse
+autobahn+Beschriftung&layer=Wohngebiet&layer=Gr%FCnfl%E4chen
&layer=gewaesser&program=/cgi-bin/mapserv&map_web_imageurl=
http://snowball/tmp/.
```

Dabei ist

- `/cgi-bin/mapserv` der Aufruf des Mapservers,
- `map=.../osnabrueck.map` das hier benötigte *Mapfile*,

- `layers=...&layer=gewaesser` die Layer die angezeigt werden sollen und
- `map_web_imageurl=http://snowball/tmp/` die Datei in der die png-Grafiken gespeichert werden.

Mit diesen Angaben wird der Mapserver aufgerufen und die ausgewählten Layer dargestellt (siehe Abbildungen in A).

9.1.1 Mehrbenutzerproblem

Wie in Kapitel 4 bereits erläutert ist der UMN Mapserver nicht für den Mehrbenutzerbetrieb geeignet. Bei jedem Nutzer, der auf den Mapserver zugreift, wird der selbe, zu dieser Zeit existierende, Layer `Wohngebiet` als Ergebnis angezeit. Das bedeutet, dass unter Umständen nicht das Wohngebiet angezeigt wird, das zu den vom Nutzer gewählten Kriterien passt. Dies ist genau dann der Fall, wenn mehrere Nutzer unterschiedliche Abfragen absenden und das Script *wohngebiet* mit anderen Parametern ausgeführt wird. Dadurch wird der Layer `Wohngebiet` immer wieder mit den aktuellen Parametern überschrieben und jeder Nutzer erhält die zuletzt berechnete Karte.
Damit der Mehrbenutzerbetrieb gewährleistet wird, muss jeder Anfrage der verschiedenen Nutzer eine eigene Datei zugeordnet werden, die das Ergebnis der Anfrage enthält. Dies wird gelöst, indem in der Datei *anzeige.php* mit Hilfe der php-Funktion `tempnam()` der eindeutige Dateiname `$filename` erzeugt wird. Dieser wird an das Script *wohngebiet* übergeben. In dem Script wird das Ergebnis der Berechnungen, das *Shapefile* des Wohngebiets, mit `$filename` benannt.

```
<?php
$tmpfile=tempnam(,,/‘‘ ,,ergebnis‘‘);
$filename=basename($tmpfile);
$file=$filename.,,.map‘‘;
shell_exec(,,sh wohngebiet $_POST[gew1] $_POST[gew2] $_POST[gew3]
$_POST[gew4] $_POST[gew5] $filename $alphacut‘‘);
?>
```

Da aber im *Mapfile* angegeben werden muss, wie der Layer heißt, der das Wohngebiet enthält, muss zu jeder Anfrage auch ein eigenes *Mapfile* erstellt werden, das mit Hilfe der übergebenen Variablen `$filename` eindeutig benannt wird.
Darin wird nun der Layer `Wohngebiet` eindeutig referenziert, indem der durch `tempnam()` erzeugte eindeutige Dateiname `$filename` dort als Name für den

Layer angegeben ist.
Damit das richtige *Mapfile* bei der Darstellung genutzt wird, muss es durch `<?php echo ($file) ?>` im Verweis zum Mapserver angegeben werden.

```
<META HTTP-EQUIV=Refresh CONTENT=,,0; URL=http://
snowball.informatik.uni-osnabrueck.de/cgi-bin/mapserv
?map=../hkraeuse/umn-mapserver/html/<?php echo ($file) ?>
&layers=sonststrassen+nebenstrassen+hauptstrassen+bundestrasse
+autobahn+Beschriftung&layer=Wohngebiet&layer=Gr%FCnfl%E4chen
&layer=gewaesser&program=/cgi-bin/mapserv&map_web_imageurl=
http://snowball.informatik.uni-osnabrueck.de/tmp/‘‘>
```

Dadurch wird erreicht, dass jedem Nutzer ein eigenes *Mapfile* und das richtige *Shapefile* zugeordnet ist und ein Mehrbenutzerbetrieb möglich wird.
Die anfallenden Dateien, die individuellen *Mapfiles* und *Shapefiles*, die bei dieser Lösung entstehen, müssen gelöscht werden. Derzeitig werden sie per Hand in regelmäßigen Abständen gelöscht. Nun stellt sich die Frage, wie dies sinnvoller umgesetzt werden könnte.
Es wäre denkbar, die Dateien mit Hilfe eines *cron jobs*[1] in PHP nach einer Stunde zu löschen. Falls der Nutzer dann noch aktiv wäre, würde ihm eine HTML-Seite präsentiert werden , die ihn darauf hinweist, dass er die Abfrage erneut absetzen muss.

[1] Ein Cron Job ist ein Programm oder ein Skript, das in zeitlichen Intervallen (cron = cronos = zeit) automatisch gestartet wird.

Kapitel 10

Ausblick

In dieser Bachelor-Arbeit wurde eine Anwendung entwickelt, mit deren Hilfe Interessierte an Hand verschiedener Kriterien ihr optimales Wohngebiet in einer Stadt berechnen lassen und auf einer Karte das Ergebnis betrachten können.
Es gibt verschiedene Möglichkeiten, wie dieses Projekt über den Rahmen dieser Bachelor-Arbeit hinaus optimiert und erweitert werden kann:

- Die **Kriterien**, die vom Nutzer ausgewählt werden können und die Bestimmung des Wohngebiets beeinflussen, können beträchtlich erweitert oder ausgetauscht werden. Es ist sehr individuell, auf welche Dinge bei der Wohnortsuche Wert gelegt wird, aber durch eine Umfrage, vielleicht auf der Hompage von PlaceToBe selber, kann erfasst werden, welche die am häufigsten gewünschten Kriterien sind.

- Die Fuzzy Logic Menge der Entfernungen, die in den Rasterzellen der Distanzkarten eingetragen sind, ist linear auf ein Intervall [0,1] abgebildet worden. Es ist aber denkbar, dass die **Bewertung** der Entfernungen durch den Nutzer nicht linear abfällt, sondern vielleicht nur am Anfang und sich ab einer gewissen Entfernung die Güte der Punkte exponentiell verschlechtert. Man könnte z.B. eine Funktion der Art:

  $$f[F_i] = \frac{1}{1 + e^{-F_i}}$$

  für die Bewertung der Entfernungen heranziehen. Die geeignete Funktion ist wiederum durch Umfrage bestimmbar.

- Im jetzigen Zustand der Arbeit werden auch Flächen in die Auswahl mit einbezogen, die **nicht bewohnbar** sind, wie z.B. Parks, Seen, Flüsse, Schienen, etc.. Diese können herausgefiltert werden, wenn man die

entsprechenden Daten vorliegen hat, indem sie in der binären Ergebniskarte (Kapitel 8) mit 0 gekennzeichnet werden.

- Wie in Kapitel 9 angesprochen kann die Ausgabe statt durch einen Mapserver auch mit Hilfe von **SVG** angezeigt werden.

- Die Anwendung könnte mit einer Datenbank verknüpft werden, in der **Wohnungen, Häuser und Grundstücke** gespeichert sind, die zum Verkauf oder zur Vermietung zur Verfügung stehen. Nach einer Abfrage durch den Nutzer würden die Objekte, die innerhalb des Ergebnisgebiets liegen, aufgelistet werden.

- Es könnte ein Formular angelegt werden, in dem **Adressen** eingegeben werden können. Für diese Adressen wird dann geprüft, ob sie im relevanten Wohngebiet liegen oder nicht. So kann der Nutzer z.B. Wohnungsanzeigen aus Zeitungen auf ihre Lage prüfen.

Anhang A

Screenshots PlaceToBe

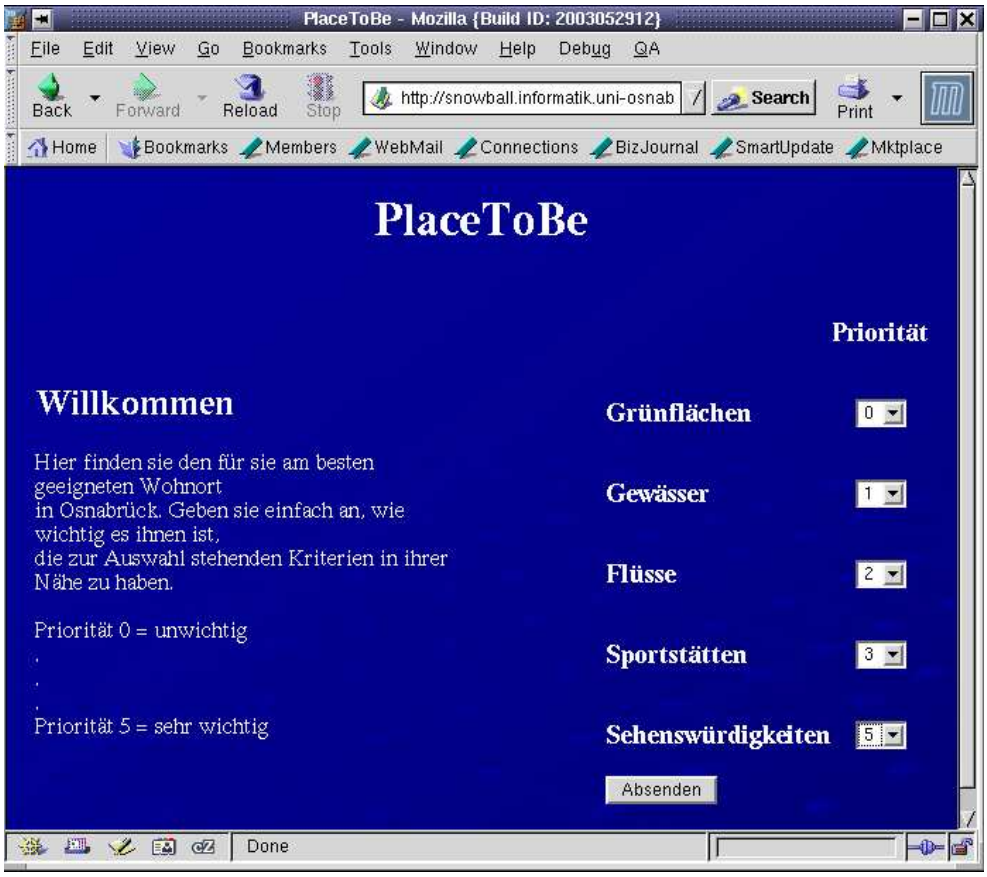

Abbildung A.1: Abfrageformular

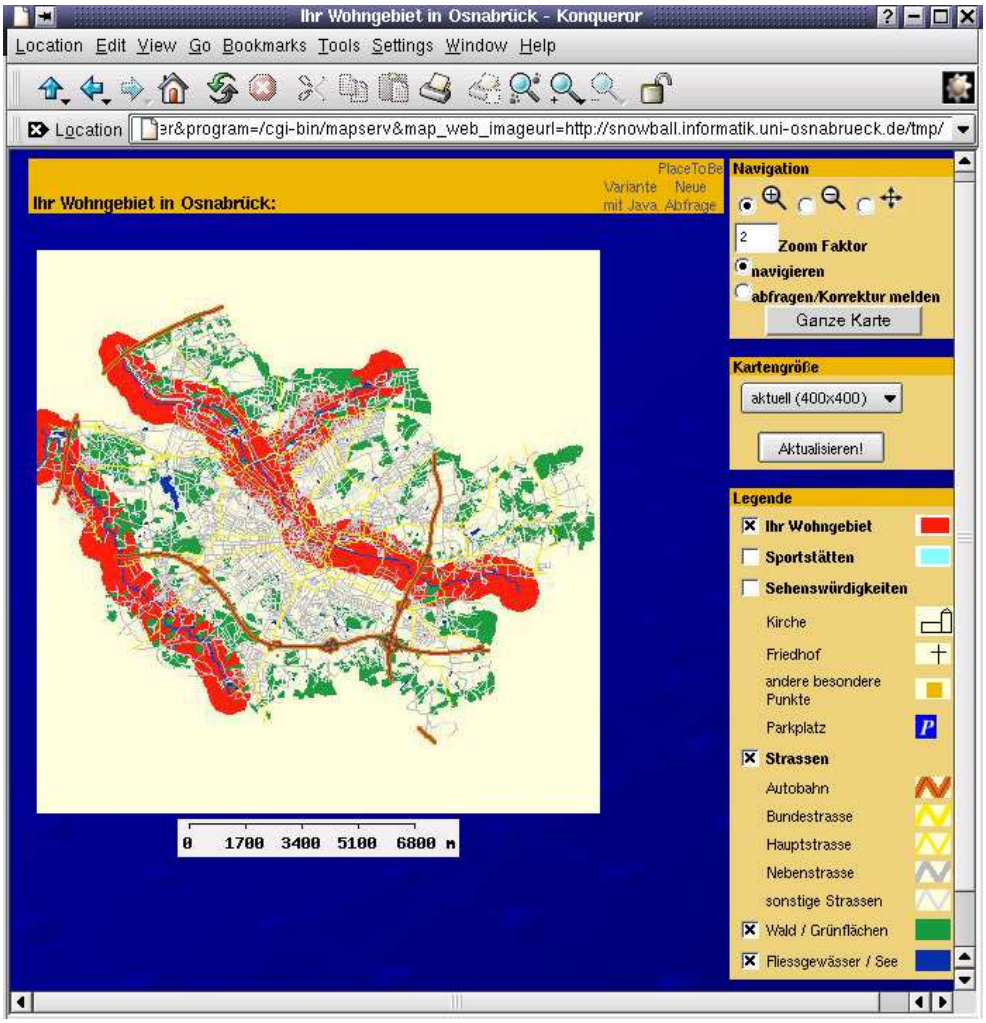

Abbildung A.2: Wohngebiet für das Kriterium Flüsse mit Priorität 1

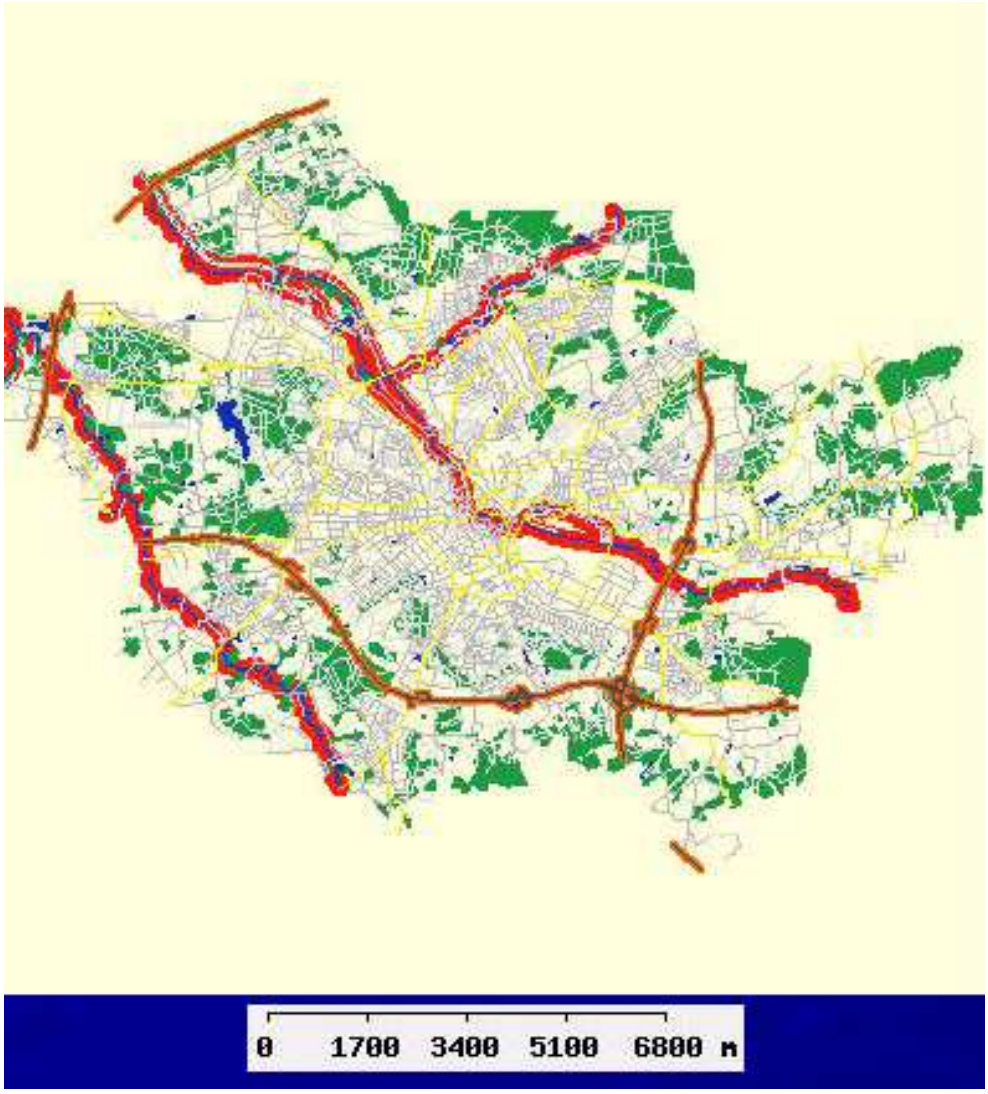

Abbildung A.3: Wohngebiet für das Kriterium Flüsse mit Priorität 5

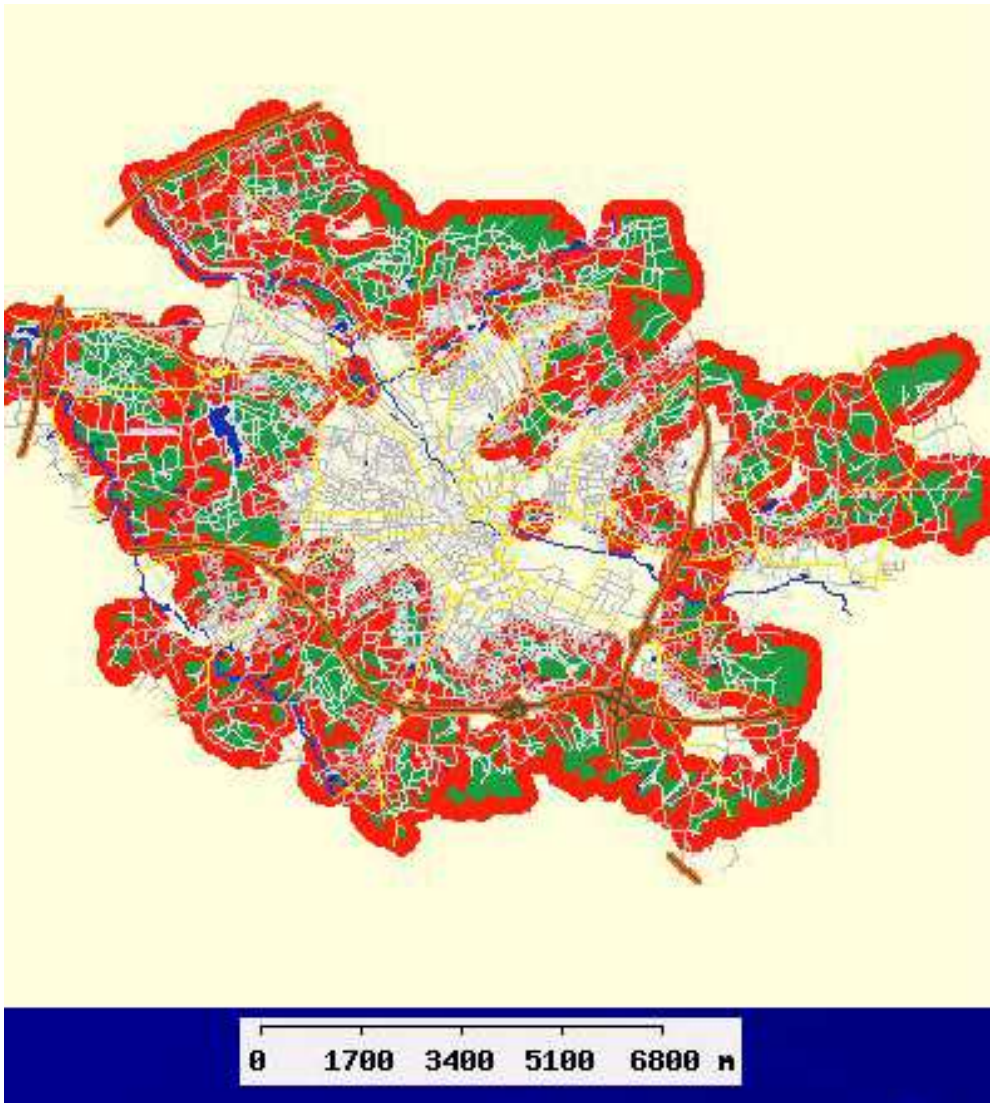

Abbildung A.4: Wohngebiet für das Kriterium Grünflächen mit Priorität 3

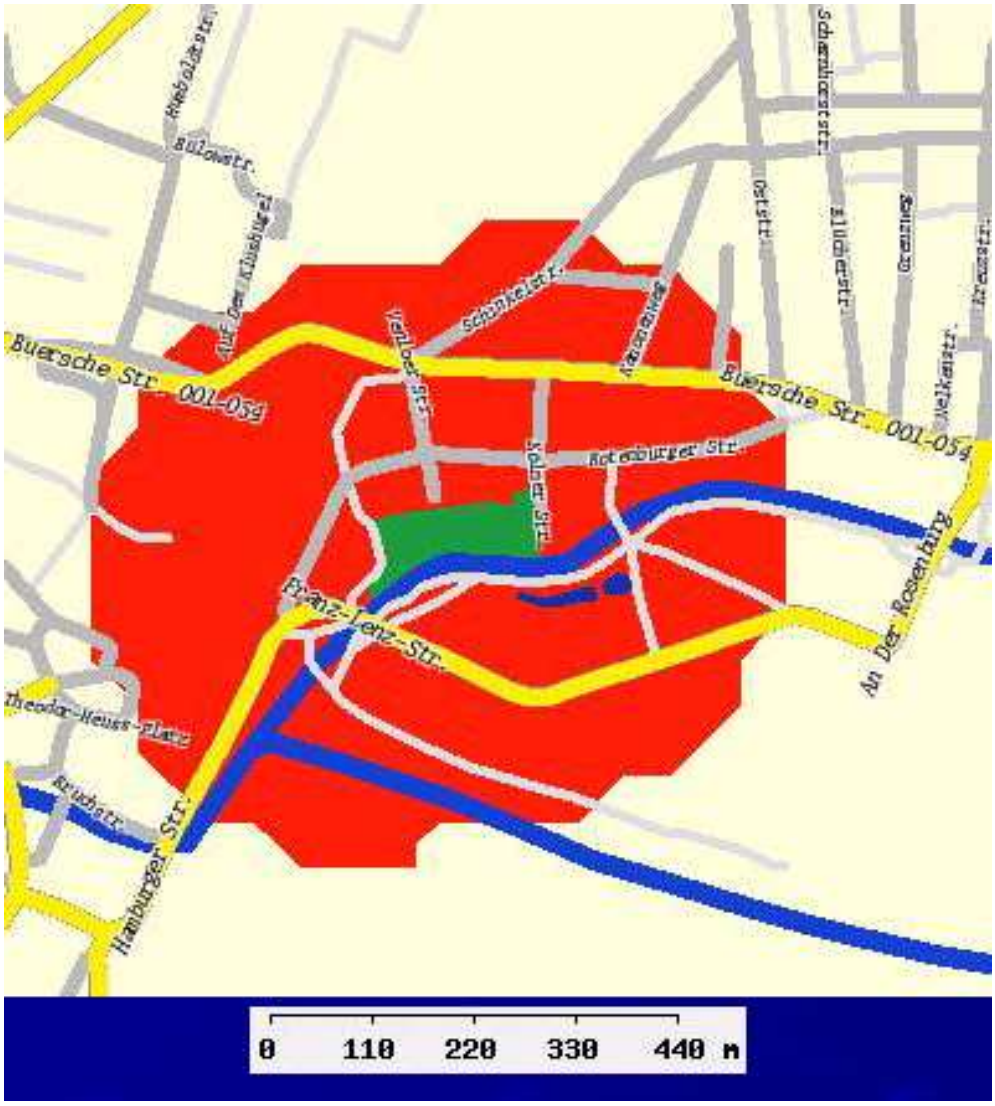

Abbildung A.5: Wohngebiet für das Kriterium Grünflächen mit Priorität 3 in einem anderem Maßstab

Anhang B

Inhalt der CD-Rom

arbeit	**Bachelorarbeit**
arbeit/latex	Latex-Quellen und Grafiken
arbeit/pdf	PDF-Version
arbeit/ps	Postscript-Version
realisierung	**der praktische Teil**
realisierung/applikation	Dateien zur Berechnung und Darstellung des Wohngebiets
realisierung/umn-mapserver	konfigurierte Dateien für den UMN-Mapserver
realisierung/README	Installationsanleitung
software	**benötigte Software**
software/grass	das GIS GRASS
software/umn-mapserver	der UMN-Mapserver

Literaturverzeichnis

[1] Markus Neteler und Helena Mitasova. Open Source GIS: A GRASS GIS Approach. KAP, 2002

[2] Markus Neteler. GRASS Handbuch, Leitfaden zum Geographischen Informationssystem GRASS. Version 1.1 (2000, 2003)

[3] Linux-Magazin - GRASS: http://www.linux-magazin.de/Artikel/ausgabe/2001/05/grass/grass.html

[4] GAV: GRASS-Kurs: http://www.grass-verein.de/grasskurs/

[5] GRASS Kurs Sommerstemester 2002: http://www.geologie.uni-freiburg.de/projekte/grass/sommer2002/infos.html

[6] Official GRASS GIS Homepage: http://grass.itc.it

[7] GRASS 5.0 Kursskript: http://www.grass-gis.de/grass/html/einfuehrung/index.html

[8] GRASS Tutorial: http://tutorial.grass.it-zone.org

[9] Das GIS-Tutorial: http://www.gis-tutor.de/start.htm

[10] GIS-Tutorial: GIS Historie: http://www.gis-tutor.de/einleitg/history.htm

[11] GIS-Tutorial: Geo Basisdaten, ALK, ATKIS: http://www.gis-tutor.de/theorie/daten/basisdat/basisdat.htm

[12] G.O. Das Internet-Magazin für Geo- und Naturwissenschaften: http://www.g-o.de

[13] GIS Einführung: http://www.kulturgeo.uni-freiburg.de/mitarb/fuest/giskurs/gisallg.html

[14] Frida: Freie Vektor-Geodaten Osnabrück:
http://www.frida.intevation.org

[15] James J. Buckley, Thomas Feuring. Fuzzy and Neural: Interactions and Applications. Physica-Verlag, 1999

[16] Unscharfe Logik (Fuzzy Logic): http://wwwags.informatik.uni-kl.de/lehre/ws03-04/ES/Folien.dir/Fuzzy.pdf

[17] Constantin von Altrock. Über den Daumen gepeilt, Fuzzy Logic: scharfe Theorie der unscharfen Mengen. ct, 1991, Heft 3

[18] Scalable Vector Graphics (SVG) 1.2:
http://www.w3.org/TR/SVG12/#howto

[19] SVG - Skalierbare Vektorgraphiken - Tutorial:
http://www.karto.ethz.ch/td/tutorial/svgtutorial

[20] GPS: Das Geheimnis des Map Datums:
http://home.t-online.de/home/kontext/mapdatum.htm

[21] ESRI White Paper - July 1998. ESRI Shapefile Technical Description:
http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[22] FreeGis MapServer: http://ftp.intevation.de/freegis/gnu-linux-i586/
freegis-1.2.3/doc/mapserver.de.html

Erklärung

Hiermit erkläre ich, dass ich die Bachelorarbeit selbstständig angefertigt und keine Hilfsmittel außer denen in der Arbeit angegebenen benutzt habe.

Osnabrück, den ....

(Unterschrift)