



Diplomarbeit

INTEGRATION VON
GENERALISIERUNGSFUNKTIONALITÄT
FÜR DIE AUTOMATISCHE ABLEITUNG
VERSCHIEDENER LEVELS OF DETAIL
VON OPENSTREETMAP WEBKARTEN

zum Erlangen des Hochschulgrades

Diplom-Ingenieur
(Dipl.-Ing.)

Bearbeiter: Ralf Klammer

Matrikelnummer: 3018143

Betreuer: Prof. Dr.-Ing. Dirk Burghardt, TU-Dresden
Dipl.-Ing. (FH) Stefan Hahmann, TU-Dresden

Tag der Einreichung: 28.04.2011



Aufgabenstellung für die Diplomarbeit

Studiengang: Kartographie

Name des Diplomanden: Ralf Klammer

Thema: Integration von Generalisierungsfunktionalität für die automatischen Ableitung verschiedener Levels of Detail von OpenstreetMap Webkarten

Zielsetzung:

Durch das rasante Wachstum des OpenStreetMap (OSM) Projektes haben die darüber verfügbaren Daten in einigen Gebieten bereits einen beachtlichen Grad an Vollständigkeit und Genauigkeit erreicht. Für viele Anwender stellen diese Daten deshalb einen gut nutzbaren umfangreichen Geodatenbestand dar. Die kartographische Visualisierung der OSM Daten geschieht unter anderem über die Webseite des Projektes openstreetmap.org mit den dafür entwickelten Renderern Mapnik und Osmarender. Aufgrund der teilweise sehr hohen Datendichte ergeben sich umfangreiche Anforderungen an die kartographische Generalisierung, die in der aktuellen Funktionalität der Renderer nur ansatzweise umgesetzt ist.

Der Diplomand wird im Rahmen der Arbeit untersuchen, inwieweit mit vorhandenen Generalisierungsoperatoren ein Beitrag geleistet werden kann, um die Komplexität von OSM Webkarten zu reduzieren und damit die Lesbarkeit zu erhöhen. Es soll ausgehend von einer Ist-Analyse der Software untersucht werden, ob die Basisrenderer Mapnik oder Osmarender geeignet sind, um Generalisierungsfunktionalität zu integrieren. Sollten sich diese als ungeeignet erweisen kann auch WMS Software, wie z.B. GeoServer, betrachtet werden. Weiterhin soll untersucht werden, inwiefern die Generalisierungsservices des Projektes WebGen / WebGen WPS genutzt werden können.

Der Diplomand wählt ein Testgebiet aus, das einer topographischen Karte entspricht, die Straßen-, ÖPNV- und Gewässernetz, administrative Grenzen, Siedlungen, Vegetation und einige Points of Interests (POI) enthält. Im Testdatensatz sollen ausgewählte Generalisierungsoperatoren für ausgewählte Objektklassen in verschiedenen Levels of Detail (LoD) getestet und implementiert werden.

Einzureichen sind zwei gedruckte Exemplare und die digitale Fassung der Arbeit in Form einer CD sowie ein Farbposter, auf dem die wichtigsten Ergebnisse der Arbeit zusammengefasst werden. Die Implementierung soll an den Betreuer der Hochschule übergeben werden und gegebenenfalls in den Foren der OpenStreetMap Entwicklercommunity bekannt gegeben werden. Weiterhin ist eine Publikation des Textteils der Arbeit auf dem Publikationsserver Qucosa der SLUB anzustreben.

Betreuer: Prof. Dr.-Ing. Dirk Burghardt
Dipl.-Ing. (FH) Stefan Hahmann

Ausgehändigt am: 01. 11. 2010

Einzureichen am: 30. 04. 2011

Prof. Dr. Manfred F. Buchroithner
Prüfungsausschuss

Prof. Dr.-Ing. Dirk Burghardt
Betreuender Hochschullehrer

Zusammenfassung

OpenStreetMap (OSM) konnte sich seit der Gründung 2004 sehr schnell etablieren und stellt mittlerweile eine konkrete Alternative gegenüber vergleichbaren kommerziellen Anwendungen dar. Dieser Erfolg ist eindeutig auf das revolutionäre Grundkonzept des Projektes zurückzuführen. Weltweit werden räumliche Daten durch Mitglieder erhoben und dem Projekt OSM zur Verfügung gestellt. Über die zugrunde liegenden Lizenzbestimmungen wird sichergestellt, dass OSM-Daten frei verfügbar und kostenfrei weiter verwendbar sind. Vor allem die Vorstellung der Unabhängigkeit von proprietären Daten hat zu starker, weiterhin zunehmender globaler Beteiligung geführt. Resultierend daraus erreichen die verfügbaren Daten inzwischen hohe Dichte sowie Genauigkeit.

Visualisierungen in Form von interaktiven, frei skalierbaren Weltkarten, welche über die vollständig automatisierten Softwarelösungen Mapnik und Osmarender erstellt werden, sind am weitesten verbreitet. Infolgedessen müssen kartographische Grundsätze und Regeln formalisiert und implementiert werden. Insbesondere in Bezug auf kartographische Generalisierung treten teils erhebliche Mängel in den entsprechenden Umsetzungen auf. Dies bildet den Ausgangspunkt der Untersuchung. Ausgehend von einer Ist-Analyse werden vorhandene Defizite identifiziert und anschließend Möglichkeiten zur Integration von Generalisierungsfunktionalitäten untersucht.

Aktuelle Entwicklungen streben die Anwendung interoperabler Systeme im Kontext kartographischer Generalisierung an, mit dem Ziel Generalisierungsfunktionalitäten über das Internet bereitzustellen. Grundlage hierfür bilden die vom Open Geospatial Consortium (OGC) spezifizierten Web Processing Services (WPS). Sie ermöglichen die Analyse und Verarbeitung räumlicher Daten. In diesem Zusammenhang werden Web Generalization Services (WebGen-WPS) auf mögliche Integration in die Softwarelösungen untersucht und bilden somit einen zentralen Untersuchungsgegenstand der vorliegenden Arbeit.

Mapnik stellt, nicht zuletzt durch dessen offengelegten Quelltext („Open Source“), optimale Voraussetzungen für jene Implementierungen zur Verfügung. Zur Verarbeitung von OSM-Daten verwendet Mapnik die freie Geodatenbank PostGIS, welche ebenfalls Funktionalitäten zur Analyse und Verarbeitung räumlicher Daten liefert. In diesem Kontext wird zusätzlich untersucht, inwiefern PostGIS-Funktionen Potential zur Anwendung kartographischer Generalisierung aufweisen.

Abstract

OpenStreetMap (OSM) has established very quickly since its founding in 2004 and has become a suitable alternative to similar commercial applications. This success is clearly due to the revolutionary concept of the project. Spatial data is collected by members worldwide and is provided to the project OSM. The underlying license agreement ensures that OSM-Data is freely available and can be used free of charge. Primarily, the idea of independence from proprietary data has led to strong, still growing, global participation. Resulting from that, the available data is now achieving high density and accuracy.

Visualizations in form of interactive, freely scalable maps of the world, which are constructed by the fully automated software solutions Mapnik and Osmarender are most common. In consequence cartographic principles and rules must be formalized and implemented. Particularly with respect to cartographic generalization, some serious faults appear in the corresponding implementations. This is the starting point of this diploma thesis. Based on an analysis of the current state, actual existing deficiencies are identified and then examined for possibilities to integrate generalization functionalities.

Recent developments aim at the deployment of interoperable systems in the context of cartographic generalization, with the intention of providing generalization functionalities over the Internet. This is based on Web Processing Services (WPS) that were developed by the Open Geospatial Consortium (OGC). They enable the analysis and processing of spatial data. In this context, Web Generalization Services (Webgen-WPS) are examined for possible integration into the software solutions and represent therefore a central object of investigation within that examination.

Mapnik provides, not least through its “open source” code, ideal conditions for those implementations. Mapnik uses the “open source” spatial database PostGIS for the processing of OSM-Data, which also provides capabilities to analyze and process spatial data. In this context is examined in addition, to what extent the features have potential for implementation of cartographic generalization.

Inhaltsverzeichnis

Aufgabenstellung.....	ii
Zusammenfassung.....	iii
Abstract.....	iv
Abbildungsverzeichnis.....	viii
Tabellenverzeichnis.....	ix
Abkürzungsverzeichnis.....	x
1 Einleitung.....	1
1.1 Motivation.....	1
1.2 Aufbau der Arbeit.....	4
2 Grundlagen.....	5
2.1 OpenStreetMap.....	5
2.1.1 Ablauf der Erstellung von OSM-Karten	6
2.1.2 Mapnik	9
2.2 Web Services.....	12
2.2.1 OGC Web Processing Services.....	12
2.2.2 Web Generalization Services.....	14
2.2.3 Verkettung von OGC Web Services.....	16
2.3 Kartographische Generalisierung.....	17
2.3.1 Konzeptionelle Modellvorstellungen.....	18
2.3.2 Generalisierungsoperatoren.....	22
3 OpenStreetMap & Generalisierung – aktueller Stand.....	24
3.1 Allgemeine Analyse und Kritik.....	25
3.2 OSM & konzeptionelle Modelle.....	28

4 Theoretische Überlegungen.....	31
4.1 Einbindung des WebGen-WPS.....	32
4.1.1 Direkteinbindung des WebGen-WPS.....	32
4.1.2 Einbindung von WebGen-WPS für „MRDB-OSM“.....	34
4.2 PostGIS-Funktionen.....	36
4.3 OpenStreetMap - Generalisierungscommunity.....	38
5 Implementierungen & Ergebnisse.....	40
5.1 Technische Voraussetzungen.....	41
5.1.1 Systemvoraussetzungen.....	41
5.1.2 Testgebiet.....	41
5.2 Einbindung des WebGen-WPS in Mapnik.....	42
5.2.1 Einbindung in den automatischen Prozess.....	42
5.2.1.1 Allgemeiner Programmablauf.....	43
5.2.1.2 Implementierungsansätze.....	44
5.2.2 Praktische Umsetzung einer „MRDB-OSM“.....	47
5.2.2.1 Verfahrensablauf.....	48
5.2.2.2 Polygonvereinfachung.....	51
5.2.2.3 Linienvereinfachung.....	57
5.3 Implementierung von PostGIS-Funktionen.....	59
5.3.1 Auswahl.....	59
5.3.2 Betonung.....	60
5.3.3 Linienvereinfachung.....	61
5.3.4 Polygonvereinfachung.....	61
6 Schlussfolgerungen und Ausblicke.....	65
6.1 Diskussion der Ergebnisse.....	65
6.2 Fazit.....	71

7	Quellennachweise.....	72
7.1	Literaturverzeichnis.....	72
7.2	Internetquellennachweis (ohne eindeutige Autoren).....	77
8	Anhang.....	79
	Anhang A - Beispiel eines Pythonskripts zum Ausführen von Mapnik.....	79
	Anhang B - Beispiel einer OGC-konformen XML-Datei zum Senden von Geodaten.....	82
	Anhang C - Codebeispiel in C++ zum WPS-Aufruf mittels cURL (libcurl).....	84
	Anhang D - OpenStreetMap-Karten im Maßstabsbereich von etwa 1:25.000.....	86
	Anhang E - Gebäudegeometrien im Maßstabsbereich von etwa 1:25.000.....	90
	Anhang F - OpenStreetMap-Karten im Maßstabsbereich von etwa 1:10.000.....	94
	Anhang G - Liniengeneralisierung mittels WebGen-WPS-Service „LineSmoothing“.....	96
	Anhang H - Auswahl nach Mindestgrößen mittels PostGIS-Funktion „ST_Length“.....	98
	Anhang I - Liniengeneralisierung mittels PostGIS-Funktion „ST_Simplify“.....	100
	Anhang J - Vergleich „ST_Simplify“ vs. „ST_SimplifyPreserveTopology“.....	102
	Anhang K - Gebäudegeneralisierung mittels PostGIS.....	105
	Anhang L - CD mit Quellcodes und Abbildungen.....	107

Abbildungsverzeichnis

Abbildung 2.1: Ablauf der Berechnung einer Karte mit Osmarender.....	8
Abbildung 2.2: Ablauf der Berechnung einer Karte mit Mapnik	9
Abbildung 2.3: Beispiel für das Strukturelement "Map" des Mapnik-Stylefile.....	10
Abbildung 2.4: Beispiel für das Strukturelement "Style" des Mapnik-Stylefile.....	10
Abbildung 2.5: Beispiel für das Strukturelement "Layer" des Mapnik-Stylefile	11
Abbildung 2.6: Beispiel für einen "GetCapabilities" - Request.....	13
Abbildung 2.7: Beispiel für einen "DescribeProcess" - Request.....	13
Abbildung 2.8: Zyklus der WPS-Operationen (Brauner 2008, Abb. 1).....	14
Abbildung 2.9: Plug-in für WebGen-WPS in OpenJUMP.....	15
Abbildung 2.10: Modell der digitalen Generalisierung	19
Abbildung 2.11: Modellvorstellung der kartographischen Generalisierung.....	20
Abbildung 2.12: Rahmenkonzept für Web Generalization Services.....	22
Abbildung 3.1: Weltansicht im Vergleich	24
Abbildung 3.2: Gebäudepolygone werden nicht vereinfacht	25
Abbildung 3.3: Fehlende Linienglättung	26
Abbildung 3.4: Fehlen von Zusammenfassung und Verdrängung	27
Abbildung 3.5: Anpassung der Modellvorstellung von Grünreich.....	29
Abbildung 4.1: Ablaufschema für „MRDB-OSM“	35
Abbildung 4.2: Einflussfaktoren auf optimale Darstellung von OpenStreetMap-Daten.....	38
Abbildung 5.1: Karte des Testgebietes.....	41
Abbildung 5.2: Schema des Pythonskript für Mapnik.....	42
Abbildung 5.3: Fehlermeldung des WebGen-WPS, die nicht nachvollzogen werden kann	45
Abbildung 5.4: Schematische Darstellung der wichtigsten Klassen des Mapnik-Quellcodes.....	46
Abbildung 5.5: Datenbankabfrage mit Bedingung.....	48
Abbildung 5.6: Plug-in zur Ansprache des WebGen-WPS.....	48
Abbildung 5.7: Übertragung von Attributen.....	50
Abbildung 5.8: Speichern der generalisierten Daten als neue Tabelle.....	50
Abbildung 5.9: Beispiel einer modifizierten Datenbankabfrage für „MRDB-OSM“	51
Abbildung 5.10: Kombierter Kartenausschnitt generalisierter Gebäude und Differenzbild	53

Abbildung 5.11: Reine Gebäudedarstellung als kombinierter Kartenausschnitt	54
Abbildung 5.12: Kombiniertes Kartenausschnitt	55
Abbildung 5.13: Reine Gebäudedarstellung	55
Abbildung 5.14: Kombinierte Darstellung eines Resultates mit <code>minlength = 5</code>	56
Abbildung 5.15: Kombinierte Darstellung un- & generalisierter Fließgewässer mit Toleranzwert = 100	57
Abbildung 5.16: Zu starke topologische Veränderungen	57
Abbildung 5.17: Beispiel für die Einbindung der PostGIS-Funktion " <code>ST_Length()</code> "	59
Abbildung 5.18: Beispiel für Betonung nach Mindestlängen.....	60
Abbildung 5.19: Generalisierte lineare Geometrien (PostGIS).....	60
Abbildung 5.20: Beispiel für eine Implementierung der PostGIS-Funktion " <code>ST_Simplify()</code> "	61
Abbildung 5.21: Differenzbild der Funktionen „ <code>ST_Simplify()</code> “ & „ <code>ST_SimplifyPreserveTopology()</code> “	62
Abbildung 5.22: Beispiel für Gebäudegeneralisierung über „ <code>ST_SimplifyPreserveTopology()</code> “	62
Abbildung 5.23: Kartenausschnitt generalisierter Gebäude in reiner Gebäudedarstellung	63

Tabellenverzeichnis

Tabelle 1: Aktuell verfügbare Generalisierungsfunktionen auf WebGen-WPS	16
Tabelle 2: Elementare Vorgänge kartographischer Generalisierung	23
Tabelle 3: Verwendete Systemvoraussetzungen bei Implementierungen.....	41
Tabelle 4: Merkmale der Formgeneralisierung von Gebäuden	52

Abkürzungsverzeichnis

AGG.....	Anti-Grain Geometry
API.....	Application Programming Interface
ATKIS.....	Amtliches Topografisch-Kartographisches Informationssystem
CC BY-SA 2.0...	Creative Commons Attribution-Share Alike 2.0
DLM	Digitales Landschaftsmodell
DKM.....	Digitales Kartographisches Modell
GIS.....	Geoinformationssystem
GDI.....	Geodateninfrastruktur
GML.....	Geography Markup Language
GPS.....	Global Positioning System
GUI.....	Graphical User Interface
HTML.....	Hypertext Markup Language
HTTP.....	Hypertext Transfer Protocol
JPEG.....	Joint Photographic Experts Group
MRDB.....	Multiple Representation Databases
OdbL.....	Open Database License
OGC.....	Open Geospatial Consortium
OpenJUMP.....	Open source Java Unified Mapping Platform
OSM.....	OpenStreetMap
PNG.....	Portable Network Graphics
PDF.....	Portable Document Format
SLD.....	Styled Layer Descriptor
SQL.....	Structured Query Language
SRID.....	Spatial Reference System Identifier
SVG.....	Scaleable Vektor Grafik
URL.....	Uniform Resource Locator
W3C.....	World Wide Web Consortium
WFS.....	Web Feature Services
WMS.....	Web Mapping Services
WPS.....	Web Processing Services
XML.....	Extensible Markup Language
XSLT.....	Extensible Stylesheet Language Transformation

1 Einleitung

1.1 Motivation

Der Charakter von Karten hat sich im Laufe der menschlichen Geschichte häufig stark gewandelt. Waren es anfangs lediglich triviale Zeichnungen zur Orientierung in der unmittelbaren Umgebung, wurden Karten über lange Zeiten hinweg als Machtinstrument verwendet, welches Privilegien und Einfluss indizierte. Eine Sichtweise, die sich in der westlich industrialisierten Welt, gerade im Zusammenhang des kalten Krieges, bis ans Ende des 20. Jahrhunderts gehalten hat (Monmonier 1991).

Im Zuge der rasanten Entwicklungen des World Wide Web seit den 1990er Jahren wandelte sich diese „Allmächtigkeit“ zu einer „Allgegenwärtigkeit“ von Karten (Meng 2008). Vor allem interaktive Anwendungen von MapQuest, Yahoo und Google haben dazu beigetragen, dass Karten heute ein ständig frei verfügbares Medium sind. Peterson (2005) betrachtete die Entwicklung von Papierkarten zu Internetkarten in drei signifikanten Phasen. In einer ersten Stufe wurden Papierkarten zunächst simpel digitalisiert und als statische Bilddatei zur Verfügung gestellt. Im weiteren Verlauf, Ende der 1990er Jahre, wurden interaktive Karten entwickelt, welche Anfang des 21. Jahrhunderts, in der dritten Stufe, immer stärker an der Lösung spezifischer Problemstellungen orientiert sind.

Im Jahr 2004 wurde mit OpenStreetMap ein völlig neuartiges Konzept vorgestellt, welches eine weitere, vierte Phase einleitete. Damit konnte erstmals jeder Nichtfachmann, sowohl individuelle Karten entwickeln, als auch an der Entstehung von Datengrundlagen mitwirken. Zuvor waren Kartenanwender reine Informationsempfänger vorgegebener Visualisierungen, die rein auf proprietären geographischen Daten basieren. Durch OpenStreetMap können nun auch Laien in Bereiche der Kartographie vordringen, die zuvor rein von Fachkräften besetzt wurden. Laut Meng (2008) besteht in der Zusammenarbeit von Amateuren und Experten das Potential einer kollektiven Intelligenz, sodass notwendiges Fachwissen durch Experten bereitgestellt wird, wodurch Anwender in der Lage sind spezifische, auf individuelle Bedürfnisse angepasste, Karten anzufertigen.

Gerade für das Projekt OpenStreetMap, müssen in diesem Zusammenhang die wissenschaftlichen Aspekte der Kartographie formalisiert und in Verfahrensabläufe umgewandelt werden. In Betrachtung der interaktiven Weltkarten von OpenStreetMap (z.B.: www.openstreetmap.org) wird diese Notwendigkeit besonders deutlich. OpenStreetMap-Daten

werden zu jeder Zeit aktualisiert, da Nutzer ständig neue Informationen in die Datenbank einpflegen. Dementsprechend werden die abgeleiteten Weltkarten regelmäßig aktualisiert, was nur in einem vollständig automatisiertem Verfahren realisiert werden kann.

In diesem Kontext muss jedoch auch einer der wichtigsten Teilbereiche der Kartographie, die kartographische Generalisierung, umgesetzt werden. Vor allem durch das hohe Maß an philosophischen Betrachtungsweisen (McMaster & Shea 1992) kartographischer Generalisierung drängt sich die Frage auf, wie dies formalisiert und implementiert werden kann. Patentlösungen für optimale Ableitungen von Kartendarstellungen aus räumlichen Informationen existieren schließlich nicht.

Zu diesem Zweck findet in der vorliegenden Arbeit erstmals eine Auseinandersetzung mit der Anwendung kartographischer Generalisierung im Projekt OpenStreetMap statt. Bisher wurden lediglich Untersuchungen von OpenStreetMap durchgeführt, welche sich mit der Qualität der Daten (Neis et al. 2010; Haklay 2009, 2010; Haklay et al. 2010; Ludwig 2010; Kounadi 2009) oder der Qualität resultierender Karten (Zollinger 2008) auseinandersetzten. Jedoch wurde kartographische Generalisierung in diesem Kontext zumeist lediglich als ein Teilbereich der Analysen umgesetzt oder blieb vollständig ungeklärt. Für kartographische Generalisierung in Bezug auf OpenStreetMap ergaben sich im Vorfeld dieser Arbeit unterschiedlichste Fragestellungen, welche dieser Untersuchung zugrunde liegen:

Besteht die Notwendigkeit zur Einbindung kartographischer Generalisierung?

Es existieren zahlreiche Kartendarstellungen der freien OpenStreetMap-Daten, weshalb in einer ersten Phase geklärt werden soll, in welcher Qualität und Quantität kartographische Generalisierung derzeit im Bereich von OpenStreetMap angewendet wird.

Ist es möglich, Generalisierungsfunktionalitäten in den Berechnungsvorgang von OpenStreetMap-Karten zu integrieren?

Der Berechnungsvorgang von OpenStreetMap-Karten wird vollständig automatisiert umgesetzt, weshalb im weiteren Verlauf untersucht werden soll, ob es pauschal möglich ist Funktionalitäten darin einzubinden.

Welche Funktionalitäten sind geeignet und wie können diese implementiert werden?

Es muss zunächst geklärt werden, welche der existierenden Softwarelösungen für Einbindungen geeignet sind. Wenn damit Implementierungen möglich sind, sollte untersucht werden, welche Generalisierungsfunktionalitäten zur Implementierung in den Berechnungsvorgang von OpenStreetMap geeignet sind und wie mögliche Implementierungen umgesetzt werden könnten.

Welche Funktionalitäten erzeugen ansprechende Resultate?

Vorgestellte Implementierungsansätze sollen praktisch umgesetzt, angewandt und anschließend daraus resultierende Ergebnisse analysiert werden. Aus diesen Analysen werden abschließend Empfehlungen aufgezeigt, ob und in welcher Form Implementierungen weiterhin angewandt werden sollten.

Wie kann die Anwendung kartographischer Generalisierung innerhalb des Projektes zusätzlich gesteigert werden?

Im Zuge der Vorbereitungsarbeiten dieser Arbeit sind bereits zahlreiche Mängel aufgefallen, die offensichtlich nicht durch die Implementierung zusätzlicher Generalisierungsfunktionalitäten behoben werden können. Deshalb soll geklärt werden, ob zusätzliche Möglichkeiten zur Steigerung des Grades angewandter kartographischer Generalisierung vorhanden sind.

Aktuelle wissenschaftliche Entwicklungen beschäftigen sich mit der Anwendung interoperabler Systeme bezüglich kartographischer Generalisierung in Form von Web Services. In diesem Kontext können über Web Processing Services (WPS) Funktionalitäten per Internet zur Verfügung gestellt werden, womit die Verarbeitung von Daten an externen Servern möglich ist. In diesem Kontext wurde der Dienst WebGen-WPS entwickelt, mit dessen Hilfe auf verschiedene Generalisierungsfunktionalitäten zugegriffen werden kann. Die Verwendung des WebGen-WPS bildet einen zentralen Bestandteil der Aufgabenstellung dieser Arbeit, weshalb insbesondere dieser Service auf mögliche Implementierungen in OpenStreetMap untersucht werden soll. Darüber hinaus werden jedoch auch alternative Lösungsansätze angestrebt.

1.2 Aufbau der Arbeit

Im folgenden Kapitel werden grundlegende Erläuterungen zu den Themen OpenStreetMap, Web Services und kartographischer Generalisierung den tatsächlichen Untersuchungen vorangestellt. In Kapitel 3 wird anschließend die Frage nach der Notwendigkeit dieser Arbeit geklärt, indem der aktuelle Stand kartographischer Generalisierung in Bezug auf OpenStreetMap analysiert wird. Darauf folgend werden in Kapitel 4 theoretische Überlegungen zur Implementierung von Generalisierungsfunktionalitäten vorgestellt. Auf dieser Basis werden in Kapitel 5 Implementierungen praktisch umgesetzt und erhaltene Resultate ausgewertet. Abschließend werden in Kapitel 6 Schlussfolgerungen gezogen, die sich aus den vorangegangenen Untersuchungen ergeben und entsprechender Forschungsbedarf parallel aufgezeigt.

2 Grundlagen

In diese Arbeit fließen unterschiedlichste Anforderungsbereiche ein, weshalb jene im folgenden Grundlagenkapitel zunächst kurz erläutert werden. Einführend wird das Projekt OpenStreetMap vorgestellt, indem wesentliche Elemente und Abläufe aufgezeigt werden. Daraufhin werden im zweiten Unterkapitel Web Services sowie diesbezügliche Umsetzungen, die zur Anwendung kartographischer Generalisierung geeignet sind, vorgestellt. Das Grundlagenkapitel schließt mit Ausführungen zur kartographischen Generalisierung, indem konzeptionelle Modelle sowie Generalisierungsoperatoren beschrieben werden.

2.1 OpenStreetMap

Das im Jahr 2004 von Steve Coast gegründete Projekt hatte von Anfang an die Maßgabe, Geodaten jedem frei zugänglich zu machen, um damit der Abhängigkeit von proprietären Daten entgegenzuwirken und eine vollständig frei erhältliche Weltkarte zu erstellen. Im Sinne einer „Wikipedia der Kartografie“ (Ramm & Topf 2010) wird eine Internetcommunity damit beauftragt, geographische Daten zu sammeln. Diese Informationen werden der Öffentlichkeit in Rohform oder als vorberechnete Karten zur Verfügung gestellt. Da die Daten unter die Lizenz „Creative Commons Attribution-Share Alike 2.0“ (CC BY-SA 2.0) gestellt wurden, ist jede Nutzungsart unter Angabe von Ursprung und Lizenz kostenlos möglich. Gleichzeitig stehen sämtliche daraus resultierenden Produkte unter dieser Lizenz. Allerdings sind die Daten in vielen Ländern nicht vollständig geschützt, weshalb derzeit auf die Open Database License (OdbL) umgestellt wird (www.openstreetmap.de/lizenzaenderung.html). Damit bietet dieses Projekt, im Rahmen der allgemeinen Nutzbarkeit, einen klaren Vorteil im Vergleich zu anderen frei erhältlichen Kartenprodukten wie „Google maps“. Denn „frei“ bedeutet nicht automatisch „kostenlos“ (Ramm & Topf 2010, S.4). Schließlich können, bei kommerzieller Nutzung von Daten anderer Anbieter, hohe Kosten oder unübersichtliche Rechtslagen entstehen. Durch die von OpenStreetMap (OSM) angewendeten Lizenzbestimmungen werden solche Faktoren vermieden.

Dieses revolutionäre Prinzip wurde von Anfang an, gerade von Fachleuten, sehr kritisch betrachtet. Es gibt weder vorgeschriebene feste Strukturen, nach denen sich Nutzer richten müssen. Noch scheint es sich auf den ersten Blick, an bestehenden wissenschaftli-

chen Konzepten zu orientieren. Zwar wurden bereits einige große Datenimporte, zum Beispiel aus TIGER- oder AND-Daten (<http://wiki.openstreetmap.org/wiki/Import>), durchgeführt. Dennoch besteht die grundlegende Datenbasis aus, von Nutzern, importierten GPS-Tracks und den daraus abgeleiteten Informationen. Dementsprechend wird häufig kritisiert, dass die Qualität so entstandener Daten nicht befriedigend sein kann, was bereits Inhalt einiger Untersuchungen war (Neis et al. 2010; Haklay 2009, 2010; Haklay et al. 2010; Zollinger 2008; Ludwig 2010; Kounadi 2009). Ein Argument, das anfänglich sicherlich noch Berechtigung hatte. Hinter dem Konzept steht jedoch die Idee, dass fehlerhafte Daten zwar aufgenommen werden, diese hingegen durch andere Mitglieder korrigiert oder gelöscht werden können. Auf diese Weise kontrolliert und korrigiert sich das gesamte Projekt quasi selbstständig. In diesem Sinne und gerade in Betrachtung der wachsenden Beteiligung wird deutlich, dass die Qualität der Daten im Verlauf der Zeit stetig zunimmt und zunehmen wird. So stieg die Zahl angemeldeter Mitglieder von ca. 100.000 (Anfang 2009) auf fast 350.000 (Anfang 2011) deutlich an. Parallel stieg die Datendichte von ca. 75.000 (Anfang 2009) auf ca. 210.000 (Anfang 2011) Uploads (<http://wiki.openstreetmap.org/wiki/Stats>).

Der Anspruch dieser Arbeit liegt auf der Erforschung verschiedener Möglichkeiten zur Integration von Generalisierungsfunktionalitäten während der Berechnung (zumeist als Rendering bezeichnet) von Kartenprodukten auf Basis von OSM-Daten. Aus diesem Grund werden in Folge relevante Aspekte dieser Prozesse dargelegt, die zum Verständnis notwendig sind.

2.1.1 Ablauf der Erstellung von OSM-Karten

Der vollständige Umfang aller Funktionen und Möglichkeiten, welche das Projekt OpenStreetMap bietet, ist so enorm, dass es vollkommen unmöglich ist, auf alle Aspekte einzugehen. Deswegen wird für umfassende Erläuterungen auf „OpenStreetMap – Die freie Weltkarte nutzen und mitgestalten“ (Ramm & Topf 2010) verwiesen, da dies als Pflichtlektüre angesehen werden kann. Ebenso ist die Verwendung des speziell angelegten OSM-Wiki (http://wiki.openstreetmap.org/wiki/Main_Page) anzuraten, da spezifische Fragestellungen dort nachvollziehbar sind. Zusätzlich werden von einigen Mitgliedern Blogs geführt, die verschiedenste Arbeitsprozesse im Detail erläutern (z.B.: <http://weait.com/>).

Zum Einstieg in die Thematik ist es zunächst notwendig Daten zu beschaffen. Dies ist durch unterschiedliche Möglichkeiten realisierbar. Sie können in vollständiger Ausführung vom zentralen Datenbankserver als sogenanntes „*planet-file*“ heruntergeladen, respektive

über die damit verbundene OSM-API¹ als separate Datenauszüge exportiert werden. Der Nachteil des „*planet-file*“ ist die hohe Speicherkapazität², die bereit gehalten werden muss. Bei Auszügen ist hingegen der Export auf einen relativ kleinen geographischen Raum beschränkt. Somit können Daten einer maximalen Fläche von 0,5°x0,5° (Pönisch 2010, 3. Kapitel) direkt exportiert werden. Darüber hinaus können gleichfalls zahlreiche vorberechnete Datenauszüge von Drittanbietern heruntergeladen werden. Diese sind jedoch so aktuell wie Daten vom zentralen Server, da jener wöchentlich aktualisiert wird.

Die Speicherung der Daten erfolgt daraufhin in einem speziellen OSM-Datenformat, welches auf dem Standard der Extensible Markup Language (XML) basiert. Es beinhaltet die grundlegenden geometrischen Attribute Punkte („nodes“) und Linien („ways“), sowie die semantischen Attribute Relationen („relation“), Kennzeichen („tag“) und Veränderungsnachweise („changeset“). Über die Relationen werden einerseits geometrische Attribute gruppiert, sodass weitere Geometrien, wie Polygone³, indirekt entstehen können. Andererseits werden Relationen zur Ableitung von räumlichen Beziehungen geometrischer Attribute verwendet. Dementsprechend müssten Relationen vielmehr als Mischform zwischen semantischen und geometrischen Attributen bezeichnet werden. Über Kennzeichen sind Semantik (z.B.: Gebäude) sowie Metainformationen mit geometrischen Objekten verknüpft. Veränderungsnachweise sind Attribute, welche an die gemeinschaftliche Arbeitsweise der Community angepasst sind.

Eine Bewertung des Dateiformates wird an dieser Stelle nicht angestellt, da es zu weitreichend und irrelevant für die vorliegende Arbeit ist.

Nun stellt sich die Frage, wie aus OSM-Daten Kartenprodukte erstellt werden? Am weitesten verbreitet sind Visualisierungen in Form von frei skalierbaren Web-Karten der gesamten Erde, sogenannte „*Slippy-Maps*.“ Sie sind analog zur Darstellung von „Google maps“ konzipiert und übers Internet jederzeit verfügbar. Gängige Praxis solcher Visualisierungen ist die Zerlegung der Darstellung in separate Bitmap-Grafiken. Diese sogenannten „*Tiles*“ (Kacheln) werden für exakt definierte Zoomstufen vorberechnet und anschließend auf Servern vorgehalten. In Zoomstufe 0 wird die gesamte Erde in nur einer Kachel abgebildet. Für die nächsthöhere Zoomstufe wird jede Kachel in vier weitere zerlegt. Folglich wächst die Anzahl der Kacheln pro Zoomstufe exponentiell, sodass für die höchstmögliche Zoomstufe 18 rund 69 Milliarden Kacheln berechnet werden müssen (vgl. Ramm & Topf 2010). Für Berechnung und Vorhaltung wurden verschiedene Lösungen implementiert. Am

1 API - Application Programming Interface

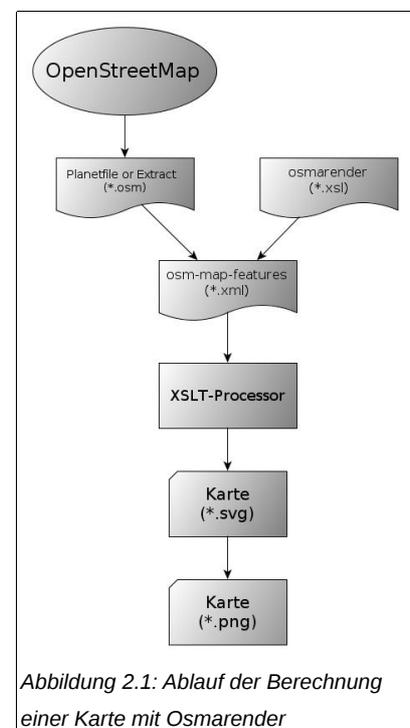
2 Größe eines „*planet-file*“ ca. 15 GB - <http://planet.openstreetmap.org/planet-latest.osm.bz2> – 31.03.2011

3 geschlossene Linien können ebenso als Polygone interpretiert werden

stärksten hat sich die Umsetzung in Form des Mapnik-Tile-Servers profiliert, wo Kacheln zentral berechnet werden (z.B.: <http://tile.openstreetmap.org/z/x/y.png>). Bereits aus dem Namen ist erkennbar, dass dafür Mapnik verwendet wird. Ebenso anerkannt ist die Serverlösung des Projektes Tiles@Home (z.B.: <http://tah.openstreetmap.org/Tiles/tile/z/x/y.png>). Dort wird die Berechnung der Kacheln auf einzelne Community-Mitglieder aufgeteilt und die Renderinglösung Osmarender verwendet. Des Weiteren ist es gleichfalls möglich, einen eigenen Kachelserver zu eröffnen (vgl. <http://weait.com/content/build-your-own-openstreetmap-server>). In allen Fällen wird eine vollständig automatisierte Verarbeitung angewendet. Durch eine Renderingsoftware werden „Tiles“ automatisch berechnet, auf einem Server vorgehalten und in der finalen Repräsentation des Browsers über eine API, welche die nötigen Kacheln automatisch lädt, angezeigt.

Alternativ können ebenso Einzelkarten eines bestimmten Gebietes berechnet werden. Es ist möglich, diese als digitale Bildschirmkarten zu verwenden oder in Form einer gedruckten Version auszugeben.

Zur Berechnung von Karten kommen vorrangig Mapnik und Osmarender zum Einsatz. Im engeren Sinne ist Osmarender kein Programm zur Berechnung von Karten. Es ist vielmehr eine regelbasierte Auswertung des OSM-Dateiformates mit Hilfe eines beliebigen XSLT⁴-Prozessors. Folglich werden die Daten in Abhängigkeit von vorab generierten Regeln, als skalierbare Vektorgrafik (SVG) erstellt (<http://wiki.openstreetmap.org/wiki/Osmarender/Howto>). Abgelegt werden diese Regeln in einer spezifischen XML-Datei (Style-Datei). Ein vereinfachter Ablauf der Berechnung einer Kartendarstellung mit Osmarender wird in Abbildung 2.1 skizziert.



Mapnik kann ebenso als „regelbasiert“ bezeichnet werden, ist allerdings ein eigenständiges Programm. Die Daten werden hierbei zunächst in eine PostGIS-Datenbank eingelesen und in Bezug auf, innerhalb einer vordefinierten XML-Datei, angegebene Spezifikationen als Bilddatei gerendert.

Als dritte Alternative ist das Programm Maperitive zu nennen, zuvor als Kosmos bekannt. Maperitive rendert in Echtzeit, dementsprechend werden OSM-Daten direkt über eine grafische Benutzeroberfläche (GUI) visualisiert. Insbesondere die spärlich verfügbare

4 XSLT - Extensible Stylesheet Language Transformation

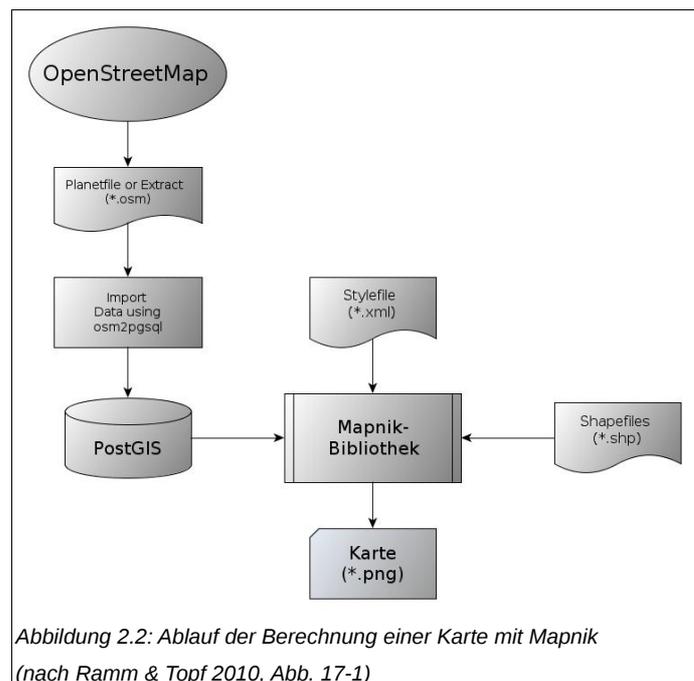
Dokumentation lässt sich darauf zurückführen, dass die Software noch in der Entwicklungsphase ist. Darüber hinaus wird Maperitive nicht zur Erstellung von „Slippy-Maps“ verwendet, weshalb es irrelevant für die vorliegende Arbeit ist.

In Kapitel 5 werden entwickelte Implementierungsmöglichkeiten anhand der Renderingsoftware Mapnik exemplarisch erprobt. Diesbezüglich werden nachfolgend wichtige Elemente der Software eingehend erläutert.

2.1.2 Mapnik

Für den Begriff „Mapnik“ existieren in der OSM-Wikipedia zwei Verwendungen. Zum Einen wird damit die Software zur Erzeugung von Karten aus Datenbanken bezeichnet. Zum Anderen verwenden Nutzer „Mapnik“ auch in Bezug auf die OSM-Weltkarte der Hauptseite (www.openstreetmap.org), da diese durch Mapnik erstellt wird. Im weiteren Verlauf bezieht sich der Begriff „Mapnik“ stets auf die Softwarelösung.

Mapnik ist eine, von Artem Pavlenko im Jahr 2006⁵ vorgestellte freie Werkzeugsammlung, die zum Rendern von Karten genutzt werden kann. Sie ist in C++ geschrieben, wobei aber auch Python-Anbindungen implementiert wurden, um den Aufruf wesentlich zu erleichtern. In Abbildung 2.2 kann der allgemeine Ablauf der Berechnung einer Karte mit Mapnik nachvollzogen werden. Da Mapnik unabhängig von OSM entwickelt wurde, basiert es auf einem divergenten Datenmodell. OSM-



Daten müssen aus diesem Grund zunächst in eine PostGIS-Datenbank importiert werden. PostGIS stellt eine Erweiterung der objekt-relationalen Datenbank PostgreSQL dar, welche die Verarbeitung von räumlichen Daten ermöglicht. OSM-Daten werden in diesem Zusammenhang über das spezielle Programm „Osm2pgsql“ (<http://wiki.openstreetmap.org/wiki/DE:Osm2pgsql>) in die PostGIS-Datenbank importiert. Räumliche Attribute werden hierbei in einer spezifischen „Geometriespalte“ (www.giswiki.org/wiki/PostGIS_Tutorial#Tabelle_mit_Geome-

⁵ Jahreszahl ist bezogen auf Angaben des Copyright im Quellcode von Mapnik

triespalte_erstellen) und die rein semantischen Attribute, Kennzeichen („tag“), als zusätzliche separate Spalten abgespeichert. Folglich werden einzelne Punkt-, Polygon-, Linien- und Straßentabellen angelegt. Überdies nutzt Mapnik zum Rendern von kleinmaßstäbigen Karten vorgefertigte „*Shapefiles*“ der weltweiten Küstenlinien, wodurch eine wesentliche Reduzierung des Berechnungsaufwandes in diesen Maßstabsbereichen erzielt wird.

Das entscheidende Element beim Rendering mit Mapnik ist das „*Mapnik-Stylefile*“. Im weiteren Verlauf dieser Arbeit wird das „*Mapnik-Stylefile*“ schlicht als Stylefile bezeichnet. Diese XML-basierte Datei enthält alle Darstellungsregeln und Ebeneninformationen, die zur Berechnung der Karte notwendig sind. Grundlegende Teile der Formatierung sind die Strukturelemente „*Map*“, „*Style*“ und „*Layer*“. „*Map*“ (Abbildung 2.3) enthält allgemeine Informationen wie Hintergrundfarbe, Referenzsystem und Mindestversion von Mapnik.

```
<Map bgcolor="#a7a7a7" srs="&srs900913;" minimum_version="0.7.1">
```

Abbildung 2.3: Beispiel für das Strukturelement "Map" des Mapnik-Stylefile

Innerhalb von „*Style*“ (Abbildung 2.4) werden alle Fakten angegeben, welche die Darstellung der geometrischen Attribute einer Ebene beeinflussen. Darin sind verschiedene Regeln („*Rule*“) enthalten, welche es ermöglichen, die Darstellung geometrischer Attribute in der Karte exakt zu definieren. Dieses Strukturelement wird als „*Symbolizer*“ bezeichnet. In Abhängigkeit beschriebener Geometrien werden jeweils spezifische Bezeichnungen verwendet (z.B.: *LineSymbolizer* & *PolygonSymbolizer*). Die Darstellung geometrischer Attribute erfolgt dabei in Bezug auf korrespondierende semantische Attribute, spezifiziert innerhalb des Elementes „*Filter*.“ Ein „*Style*“ ist über den Namen immer direkt mit einer bestimmten Darstellungsebene („*Layer*“) verknüpft.

```
<Style name="tunnels-casing">
  <Rule>
    <Filter>[highway] = 'motorway' or [highway]='motorway_link'</Filter>
    &maxscale_zoom12;
    &minscale_zoom12;
    <LineSymbolizer>
      <CssParameter name="stroke">#506077</CssParameter>
      <CssParameter name="stroke-width">3</CssParameter>
      <CssParameter name="stroke-dasharray">4,2</CssParameter>
    </LineSymbolizer>
  </Rule>
  ...
</Style>
```

Abbildung 2.4: Beispiel für das Strukturelement "Style" des Mapnik-Stylefile

Mit dieser Syntax orientierte Pavlenko sich offensichtlich an der ersten Version des, vom „Open Geospatial Consortium“ (OGC) vorgestellten, Formats „Styled Layer Descriptor“ (SLD) (OGC 2002). Vor allem die Spezifikation der Darstellungsweisen über vordefinierte Styles, Regeln, Filter und Symbolizer wird dort vergleichbar umgesetzt. Darin sind zusätzliche Informationen der angesprochenen Ebenen definiert, welche im Stylefile ebenfalls über „Layer“ (Abbildung 2.5) abgefragt werden. Alle Ebenen der resultierenden Karte werden dadurch näher beschrieben. Inhalt sind einerseits die Namen der damit direkt verbundenen Styles und andererseits die Datenquelle. Im Kontext des Rendering von OSM-Daten ist die Datenquelle natürlich eine PostGIS-Datenbank. Zur Abfrage der gewünschten Objekte wird als Parameter eine SQL-Abfrage spezifiziert, wobei die geometrischen Attribute in Bezug auf deren semantischen Attribute abgefragt werden. Insbesondere dadurch existiert ein großer Unterschied zu SLD (OGC 2002), da eine Datenbankabfrage dort nicht vorgesehen ist.

```
<Layer name="buildings" srs="+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0
+lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +no_defs
+over">
<StyleName>buildings</StyleName>
<Datasource>
  <Parameter name="dbname">meingis</Parameter>
  <Parameter name="estimate_extent">>false</Parameter>
  <Parameter name="extent">-20037508,-
19929239,20037508,19929239</Parameter>
  <Parameter name="host"></Parameter>
  <Parameter name="password"></Parameter>
  <Parameter name="port"></Parameter>
  <Parameter name="table">
(select way from planet_osm_polygon where (building = 'yes')
) as buildings
</Parameter>
  <Parameter name="type">postgis</Parameter>
  <Parameter name="user">ralf</Parameter>
</Datasource>
</Layer>
```

Abbildung 2.5: Beispiel für das Strukturelement "Layer" des Mapnik-Stylefile für eine simple Abfrage von Gebäudegeometrien

Über ein Pythonskript (Anhang A) wird der Renderingprozess begonnen. Alle notwendigen Routinen werden dabei angesprochen und im Ergebnis ein Kartenprodukt ausgegeben. Standardmäßig wurde zunächst nur die Ausgabe als PNG, PNG256 oder JPEG implemen-

tiert. Dies wird über die Grafikbibliothek „Anti-Grain Geometry“ (AGG) realisiert. Durch Einbindung der Cairographics-Bibliothek ist allerdings seit Mapnikversion 0.7.0 auch die Ausgabe im SVG- und PDF-Format möglich.

2.2 Web Services

Einen Hauptteil der vorliegenden Arbeit macht die Untersuchung aus, inwiefern es möglich ist Generalisierungsfunktionalitäten über Web Services zu integrieren. Dementsprechend werden in diesem Kapitel notwendige Grundlagen beschrieben, wobei für detaillierte Erläuterungen auf die verwendeten Quellen verwiesen wird.

Klassische Web Services sind laut Definition: „...a software system designed to support interoperable machine-to-machine interaction over a network“ (W3C 2004). Eine Anwendung dieser Technik im geoinformationstechnischen Bereich wurde spätestens durch die Entwicklung eines entsprechenden Standards (OGC 2005) möglich.

Foerster (2010) unterteilt die vom OGC spezifizierten Web Services in 3 grundlegende Kategorien. *Datenservices* stellen Daten zur Verfügung, die weiter verarbeitet werden können, beispielsweise „Web Feature Service“ (WFS) für Vektordaten. Über *Darstellungsservices* werden vorgefertigte Karten abgerufen. Bezeichnet wird ein solcher Service als „Web Map Service“ (WMS). An *Prozessierungsservices* können räumliche Daten gesendet, dort mit Hilfe verschiedener Operatoren editiert und abschließend ein verarbeitetes Resultat zurückerhalten werden. Diese Art der Umsetzung ermöglicht folglich die Analyse sowie Verarbeitung räumlicher Daten und nennt sich „Web Processing Service“ (WPS). Für die vorliegende Arbeit ist eine Implementierung von WPS vorgesehen, weshalb sich eine detaillierte Beschreibung derselben anschließt.

2.2.1 OGC Web Processing Services

Web Processing Services ermöglichen es, Daten im Sinne einer GDI an einen Server zu senden, diese dort auszuwerten und zu bearbeiten. Verschiedenste Datenverarbeitungsmöglichkeiten können somit auf einem zentralen Server vorgehalten und zugänglich gemacht werden. Die Möglichkeit des Austausches von Wissen über Algorithmen ist ein weiterer Vorteil dieser Technik (vgl. Kapitel 2.2.2). Im Jahr 2007 wurde über den Standard

des „OpenGIS Web Processing Service“ (OGC 2007) eine einheitliche Spezifikation bezüglich der Umsetzung solcher Services im Bereich der OGC Web Services vorgeben.

Der Aufruf von Operationen eines WPS-Servers kann über HTTP/GET gesendet und Informationen direkt in den Uniform Resource Locator (URL) eingebunden werden. Sie sind also Teil des Headers (z.B.: Abbildung 2.6). Des Weiteren können HTTP/POST-Anfragen im XML-Format verarbeitet werden. Dementsprechend erfolgt der Serveraufruf in Form einer separaten XML-Datei (z.B.: Anhang B). Bei OGC-WPS-Implementierungen ist eine der beiden Formen des Aufrufs stets obligatorisch und kann zusätzlich in alternativer Form gesendet werden (OGC 2007). In Folge wird ausschließlich die jeweils verbindliche Form genannt.

OGC-WPS bestehen immer aus 3 grundlegenden Operationen. Mittels „*GetCapabilities*“ (Abbildung 2.6) werden auf Anfrage alle verfügbaren Prozesse in XML zurückgeliefert. Für diese Operation ist der Aufruf über HTTP/GET vorgeschrieben.

```
http://kartographie.geo.tu-dresden.de/webgen_wps/wps?  
service=WPS&Request=GetCapabilities&AcceptVersions=1.0.0&language=en-GB
```

Abbildung 2.6: Beispiel für einen "GetCapabilities" - Request

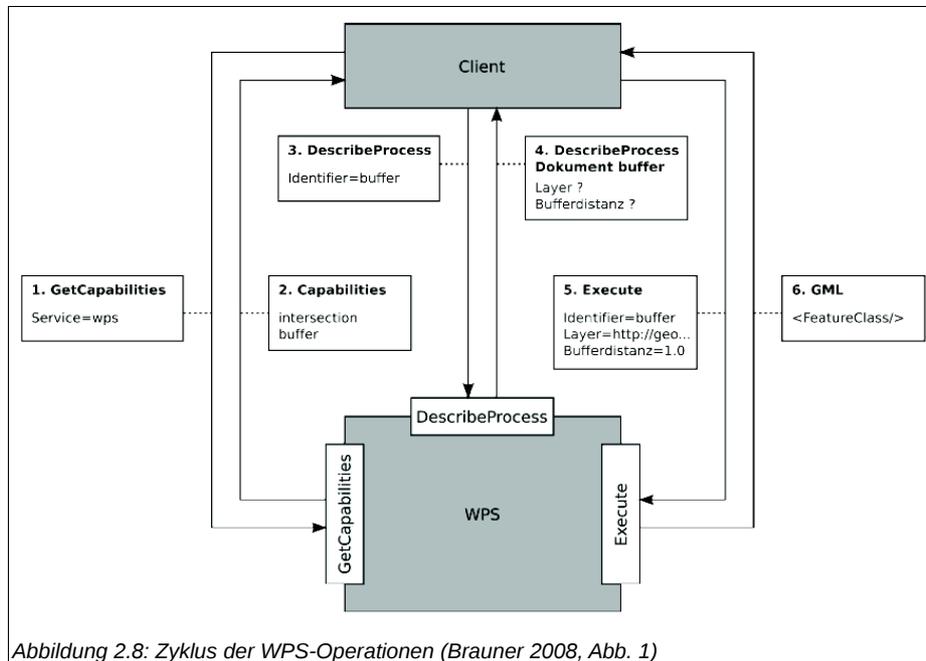
Über „*DescribeProcess*“ (Abbildung 2.7) erfolgt die Rückgabe aller Informationen, die einen bestimmten Prozess betreffen. Vor allem die Ein- und Ausgabeparameter zählen darunter, da sie für die Ausführung von Diensten notwendig sind. Dieser Befehl muss ebenfalls als HTTP/GET übermittelt werden und liefert ein XML-Antwortdokument zurück.

```
http://kartographie.geo.tu-dresden.de/webgen_wps/wps?  
service=WPS&Request=DescribeProcess&version=1.0.0&language=en-  
GB&Identifier=ch.unizh.geo.webgen.service.BuildingSimplification
```

Abbildung 2.7: Beispiel für einen "DescribeProcess" - Request

Abschließend ist die Datenübermittlung mittels „*Execute*“ zu nennen. Auch hierbei werden Daten an den Server gesendet, dort verarbeitet und das Resultat zurückgeliefert. Die Anfrage muss über HTTP-POST als XML-Datei (siehe Anhang B) gesendet werden. So wird die Übertragung großer Datenmengen gewährleistet. In diesem Zusammenhang wird zwischen einfachen Daten („*LiteralData*“), wie Datentyp Integer, und komplexen Daten („*ComplexData*“), beispielsweise im Format der „*Geography Markup Language*“ (GML), unterschieden.

In vereinfachter Form kann der Zyklus der WPS-Operationen in Abbildung 2.8 sehr gut nachvollzogen werden.



2.2.2 Web Generalization Services

Um einen allgemeinen Überblick zu geben, wird im Folgenden die Chronologie der „Web Generalization Services“ kompakt dargestellt. Eine ausführliche Beschreibung der Entwicklungsstufen dieser Technik ist in der Arbeit von Foerster (2010) ersichtlich, weshalb die Erläuterungen in dieser Arbeit bewusst sehr knapp gehalten sind. Gleichwohl wird zunächst nicht auf das zugrunde liegende konzeptionelle Modell der „Web Generalization Services“ näher eingegangen, sondern in Kapitel 2.3.1 ausführlich erläutert.

Generalisierung über Web Services wurde zunächst nur als Plattform zum Austausch von Generalisierungswissen erdacht (Edwards et al. 2003), wodurch es erstmals möglich werden sollte, innerhalb einer wissenschaftlichen Community, Algorithmen zu entwickeln und über Web Services auszutauschen. Aus diesem Konzept entstand die Idee, Generalisierungsalgorithmen über eine Nutzeroberfläche auch Communityfremden Anwendern zur Verfügung zu stellen (Edwards et al. 2005).

Neun & Burghardt (2005) entwickelten mit „WebGen“ eine auf Web Services basierende Umsetzung für das zuvor entworfene Grundgerüst (Edwards et al. 2003). Diese XML-basierte Umsetzung kann per Webbrowser oder javabasiertes OpenJUMP⁶-Plug-in angesprochen werden. Infolge der Standardisierung des OGC-WPS kam der Wunsch nach einer Anpassung von WebGen an diesen Standard auf (Foerster 2008), sodass es zum WebGen-WPS weiterentwickelt wurde. Dieses ist inzwischen frei verfügbar und kann über http://kartographie.geo.tu-dresden.de/webgen_wps/ getestet werden. Für einfache und benutzerfreundliche Anwendung wurde wiederum ein Plug-in für das frei verfügbare Geoinformationssystem OpenJUMP entwickelt. Dieses ist bereits auf den Standard der OGC Web Services angepasst, was zuvor lediglich durch Einbindung von WMS-Ebenen deutlich wurde. Da der Aufruf des WebGen-WPS nun möglich ist, wird der Charakter eines web-basierten GIS deutlich verstärkt.

In Abbildung 2.9 lässt sich die grafische Benutzeroberfläche (GUI) des OpenJUMP-Plug-in erkennen. Die Serverkommunikation entspricht dem OGC-Standard und wird über die beschriebenen grundlegenden Operationen (vgl. Kapitel 2.2.1) realisiert. Foerster (2010) kritisiert, dass eine semantische Interoperabilität bisher nicht erreicht werden

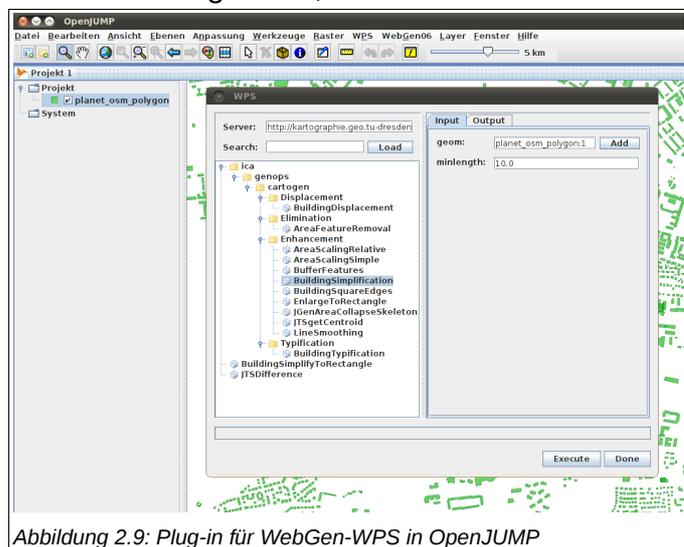


Abbildung 2.9: Plug-in für WebGen-WPS in OpenJUMP

kann. Zwar wurde eine Verkettung von Diensten bereits implementiert (Burghardt & Neun 2006), allerdings kann dies nicht vollständig automatisiert umgesetzt werden, da die Konfiguration nach wie vor manuell erstellt werden muss (Foerster 2010).

Alle aktuell verfügbaren Funktionen, die über „GetCapabilities“ erhalten werden, sind in Tabelle 1 aufgelistet.

Praktische Anwendungen des WebGen-WPS-Plug-in sowie der generische Aufruf des Servers, in Verbindung mit der Verwendbarkeit für OSM-Daten, wird in Kapitel 4.1 ausführlich beschrieben. Hauptsächlich mit Blick auf die Einbindung des WebGen-WPS wird untersucht, inwiefern bisherige Implementierungsmöglichkeiten in den automatischen Renderingvorgang von OSM-Daten vorhanden sind. Diesbezüglich wird es notwendig, einzelne Softwarekomponenten und Web Services zu verknüpfen und automatisiert nachein-

⁶ OpenJUMP (Open source Java Unified Mapping Platform) – freies Geoinformationssystem (GIS)

ander anzusprechen. Einzelne Aspekte der Verkettung von Web Services lassen sich darauf anwenden, weshalb die wesentlichen Aspekte im Folgenden komprimiert aufgezeigt werden.

Generalisierungsfunktion	Beschreibung (übersetzt)
AreaFeatureRemoval	Entfernt Features, die kleiner als der übermittelte „ <i>tosmall</i> “ - Wert sind
AreaScalingRelative	Erweitert alle Gebäudegeometrien einer Ebene in Relation zur Originalgröße
AreaScalingSimple	Erweitert alle Gebäudegeometrien in Relation zur Mindestgröße
BufferFeatures	Pufferfunktion für Punkte, Linien und Polygone
BuildingDisplacement	Versetzen von Features in Relation zu Mindestabstand unter Berücksichtigung des Abstandes zu einschränkenden Features (z.B.: Straßen)
BuildingSimplification	Gebäudevereinfachung
BuildingSimplifyToRectangle	Gebäudevereinfachung zu Rechtecken
BuildingSquareEdges	Erzeugung quadratischer Gebäudeecken
BuildingTypification	Typisierung von Gebäuden oder anderen Features
EnlargeToRectangle	Erweiterung von Polygonen (z.B.: Gebäuden) zu Rechtecken
JGenAreaCollapseSkeleton	Keine Beschreibung vorhanden
JTSDifference	Keine Beschreibung vorhanden
JTSgetCentroid	Keine Beschreibung vorhanden
LineSmoothing	Linienglättung

Tabelle 1: Aktuell verfügbare Generalisierungsfunktionen auf WebGen-WPS

2.2.3 Verkettung von OGC Web Services

Zur Verkettung von Diensten ist es notwendig, dass diese jeweils interoperabel sind und sich einem gemeinsamen Standard unterordnen. Dementsprechend sollte jeder Dienst einer Verkettung an den Standard der OGC Web Services angepasst sein.

Drei grundlegende Fragen sollten bei gegenseitiger Verknüpfung von Diensten in Bezug auf die Einrichtung von Möglichkeiten zur Benutzerinteraktion beachtet werden (Alameh 2003). In welchem Maße der Benutzer einen Einblick in die Verkettung einzelner Dienste bekommen sollte? Inwiefern die Datenquellen eingesehen werden können? Welche Fehler angezeigt werden sollten? Foerster (2010) fasst die Umsetzung dieser Fragen auf drei Interaktionsmöglichkeiten zusammen. Bei „transparenter“ Interaktion hat der Nutzer vollen Zugriff auf einzelne Aktionen, wobei allerdings ein hohes Maß an Kennt-

nissen der einzelnen Services vorausgesetzt werden muss. Als „transluzent“ werden Verkettungen bezeichnet, bei welchen dem Nutzer zwar einzelne Services und Interaktionen bekannt sind, die Reihenfolge dieser jedoch nicht variabel ist. „Opake“ Verkettungen sind für den Nutzer ausschließlich als ein Service sichtbar, wobei keine Möglichkeiten zur Interaktion angeboten werden.

2.3 Kartographische Generalisierung

Innerhalb der kartographischen Generalisierung greifen so viele Bereiche ineinander, dass dies eine der komplexesten Thematiken in der Kartographie darstellt. Das Spektrum reicht von philosophischen Betrachtungsweisen (McMaster & Shea 1992) in abstrahierten Modellvorstellungen (Hake et al. 2002) bis hin zum rein technischen Verständnis als Optimierungsproblem (Lamy et al. 1999). Infolgedessen wird in diesem Kapitel Generalisierung zunächst allgemein definiert, die wichtigsten konzeptionellen Modelle automatisierter Generalisierung aufgezeigt und abschließend bedeutende Generalisierungsoperatoren beschrieben.

Im Allgemeinen ist die kartographische Generalisierung eine notwendige Verallgemeinerung der komplexen Realität zur vereinfachten und optimierten Darstellung in einer Karte. Da das menschliche Auge perzeptiven Grenzen unterliegt, wird ein Großteil der Darstellungsmöglichkeiten innerhalb einer Karte von Mindestgrößen bestimmt (Bollmann & Koch 2001). In Abhängigkeit vom Maßstab müssen zahlreiche Details entfernt oder geometrisch verändert werden, denn mit Verdoppelung der Maßstabszahl nimmt der Darstellungsraum um ein Viertel ab (SGK 2002). Ebenso bestimmt das Ausgabemedium wie stark die darzustellenden Daten verallgemeinert werden müssen. Bildschirme haben grundsätzlich eine geringere Auflösung als ein gedrucktes Resultat. Folglich müssen Objekte auf Bildschirmkarten wesentlich größer dargestellt werden, wodurch der Darstellungsraum wiederum eingegrenzt wird. Der gewählte Generalisierungsgrad hat somit einen hohen Einfluss auf das resultierende Kartenbild. Wird er sehr hoch gewählt, entsteht ein deutliches und prägnantes Bild, während der Informationsgehalt recht niedrig ausfällt. Ein zu geringer Abstraktionsgrad erzeugt ein dichtes Kartenbild mit hoher Informationsdichte, wodurch die „Grenze der graphischen Belastbarkeit“ (Arnberger & Kretschmer 1975, S.202) überschritten werden könnte.

Kartographische Generalisierung kann allgemein in zwei Aspekte unterteilt werden. Zum Einen semantische Auswahl und zum Anderen geometrische Modifikation von Objekten (Wright 1942, Ratajski 1967, Robinson 1978). Von semantischer Generalisierung wird gesprochen, wenn Objekte nicht in die resultierende Karte aufgenommen werden, da sie zweckbestimmt irrelevant sind oder maßstabsbedingt nicht dargestellt werden können. Geometrische Generalisierung wird angewandt, wenn ein Objekt dargestellt werden muss, dieses aber geometrisch nicht exakt abgebildet werden kann. Eine derart simple Klassifizierung wird dem komplexen Prozess kartographischer Generalisierung nicht zureichend gerecht, weshalb sich eine Vorstellung grundlegender Konzepte anschließt.

2.3.1 Konzeptionelle Modellvorstellungen

Der komplexe Werdegang konzeptioneller kartographischer Modelle wurde bereits von zahlreichen Autoren (McMaster & Shea 1992, Sarjakoski 2007, Harrie & Weibel 2007, Regnaud & McMaster 2007, Foerster 2010) umfassend erläutert. Dementsprechend soll an dieser Stelle lediglich eine kurze Zusammenfassung und Erläuterung der, für diese Arbeit, relevanten Modelle gegeben werden.

Konzeptionelle Modellvorstellungen gehen in der kartographischen Generalisierung seit jeher mit dem Wunsch nach Automatisierung des gesamten Generalisierungsprozesses einher. Die allgemeine Formalisierung begann Anfang des 20. Jahrhunderts mit der erstmaligen Beschreibung von Kartographie als Wissenschaft durch Eckert (1921). Erste Ansätze zur Automatisierung folgten daraufhin erst zwischen 1960 und 1980 mit dem Entwurf von Generalisierungsalgorithmen, deren Auslegung sich allerdings nur auf exakt definierte Problemstellungen bezog (z.B.: Douglas & Peucker 1973). In diesem Zeitraum wurden ebenfalls erste formale Modelle vorgestellt. Hierbei dominierte das Verständnis von kartographischer Generalisierung als Prozess, der in verschiedene Teilbereiche untergliedert (Robinson et al. 1978, Morrison 1974, Ratajski 1967) oder als Kommunikationsprozess (Bertin 1967) aufgefasst werden kann.

Die verschiedenen Ansätze fassten McMaster & Shea (1992), vor allem mit Blick auf neue Computertechnologien, in ihrem fundamentalen Grundgerüst für digitale Generalisierung zusammen. Sie zerlegen den gesamten Generalisierungsprozess in die Teilaspekte: philosophische Grundsätze („Warum?“), kartometrische Auswertung („Wann?“) und räumliche & Attributtransformationen („Wie?“).

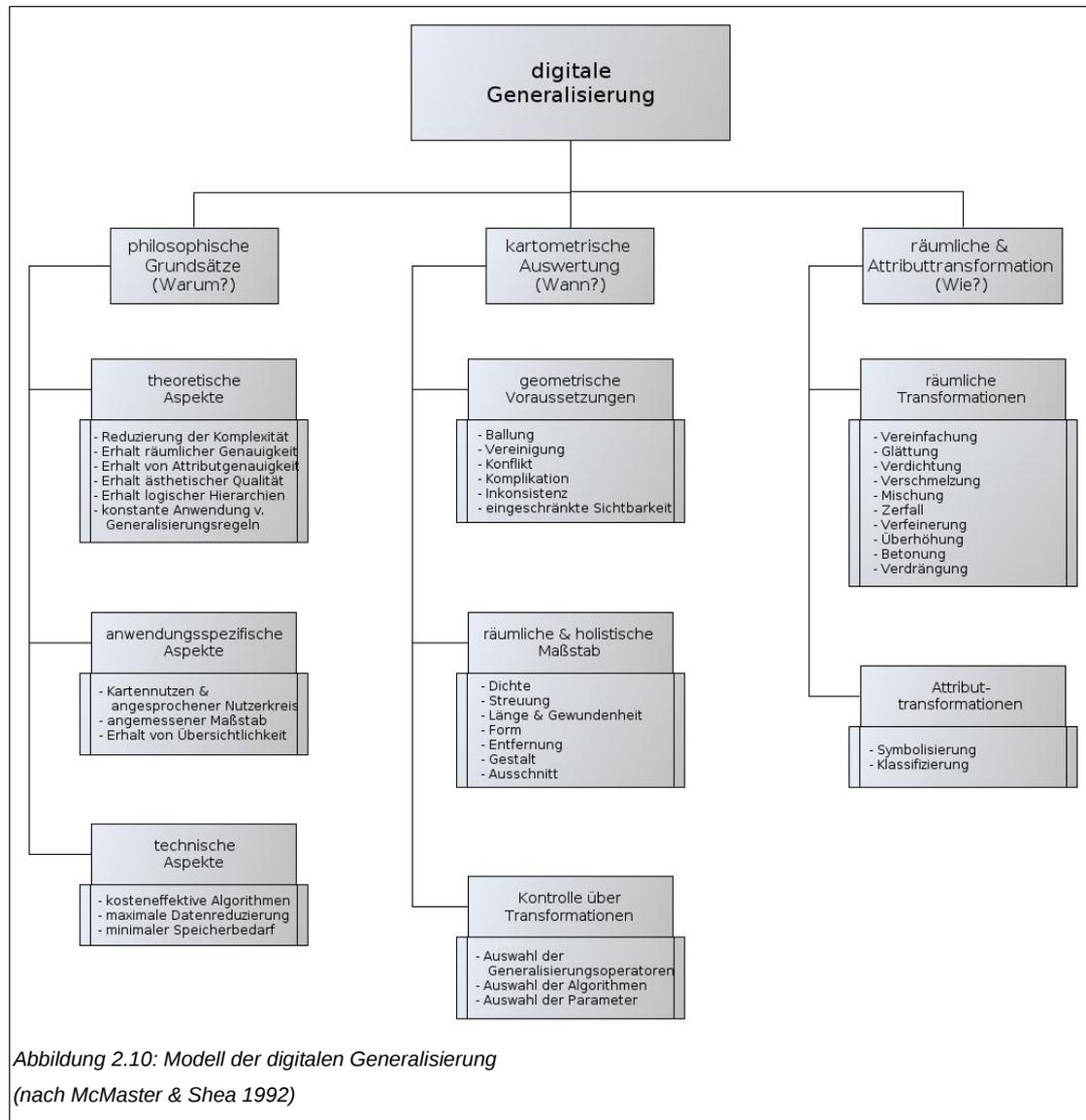
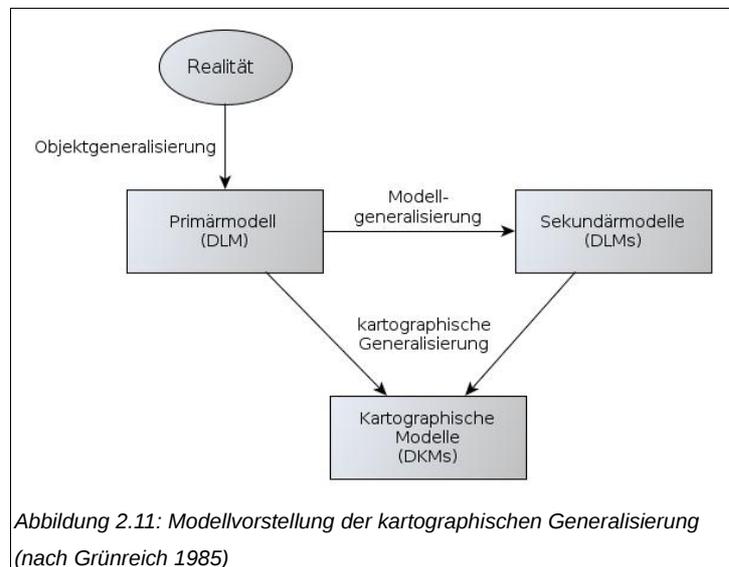


Abbildung 2.10: Modell der digitalen Generalisierung
(nach McMaster & Shea 1992)

Unter *philosophischen Grundsätzen* verstehen sie in diesem Zusammenhang die Beachtung von allgemeinen, intuitiven kartographischen Prinzipien, das Verständnis von Anforderungen an spezifische Generalisierungsproblematiken sowie die Einbeziehung bestehender Berechnungstechniken & -möglichkeiten. Der ideale Punkt für Generalisierung ist dann erreicht, wenn die Karte in ihrer Darstellung versagt (Shea & McMaster 1989). Dementsprechend betrachten sie *kartometrische Auswertungen*, zur Bestimmung dieses Punktes, unter drei Blickwinkeln: Die Notwendigkeit von Generalisierung unter bestimmten geometrischen Voraussetzungen. In welchem Maß diese angesetzt werden. Und wie einzelne Transformationen kontrolliert werden können.

Der Kern des gesamten Konzeptes sind Generalisierungsoperatoren. Diese Operatoren entsprechen den Aspekten *räumlicher & Attributtransformation* und sind in Abbildung 2.10 detailliert aufgeführt. Darüber hinaus werden Generalisierungsoperatoren in Kapitel 2.3.2 näher betrachtet. Dieses „globale konzeptionelle Modell“ (McMaster & Shea 1992) kann als Schnittstelle zwischen manueller und automatisierter digitaler Generalisierung angesehen werden, da es für beide Bereiche anwendbar ist.

Im Zusammenhang mit der Entwicklung von ATKIS wurde eines der ersten konzeptionellen Modelle vorgestellt, welches auf einen automatisierten Lösungsansatz ausgerichtet ist (Grünreich 1985). Hierbei wird, zur Abgrenzung der einzelnen Generalisierungsstufen, zwischen Objekt-, Modell- und kartographischer Generalisierung unterschieden (Abbildung 2.11). *Objektgeneralisierung* beschreibt die semantische Ableitung eines Primärmodells aus der Realität, das sogenannte digitale Landschaftsmodell (DLM). Durch semantische und geometrische Generalisierung lassen sich daraus weitere digitale Landschaftsmodelle mit niedrigerer Detailtiefe ableiten. Dieser Prozess wird als *Modellgeneralisierung* definiert und erzeugt das sogenannte Sekundärmodell. Als *kartographische Generalisierung* wird die Ableitung



einer Symbolisierung aus einem DLM bezeichnet, das digitale kartographische Modell (DKM). An dieser Stelle fließt das gesamte kartographische Wissen zur optimalen Darstellung einer Karte ein.

Die Vorstellung von „Modellgeneralisierung“ hatten ebenso Brassel & Weibel (1988) als sie ihr Konzept für eine nahezu vollständige Automatisierung entwickelten. Allerdings differenzierten sie zunächst noch zwischen den Begriffen „statistische Generalisierung“ und „kartographische Generalisierung“ (Sarjakoski 2007). Sie unterteilen Generalisierung in die fünf Stufen: Strukturerkennung, Prozesserkennung, Prozessmodellierung, Prozessausführung und Datenanzeige.

In den 1990er Jahren stellte sich eine allgemeine Akzeptanz dieser konzeptionellen Modelle ein. Um Automatisierung von Generalisierung weiter voran zu treiben, wurden diese aufgegriffen und modifiziert. Kilpeläinen (1992) nutzte beispielsweise den Fortschritt

von Datenbanktechnologien und entwickelte das Modell der „Multiple Representation Databases“ (MRDB). Objekte werden dabei in verschiedenen Abstraktionsleveln abgespeichert und können, je nach Bedarf, für unterschiedlichste Repräsentationen abgerufen werden.

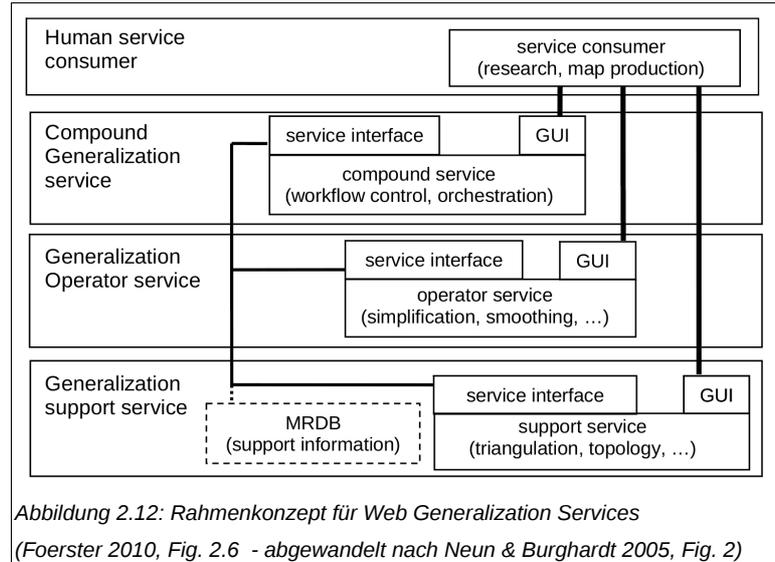
Die rein regelbasierte Umsetzung der fundamentalen Konzepte wurde in dieser Zeit („Condition-Action Modelling“) überwunden und die Interaktion zwischen Mensch und Computer („Human-Interaction Modelling“) in den Vordergrund gestellt (Harrie & Weibel 2007).

Darüber hinaus wurden auch neue, alternative Ansätze verfolgt. So kam Beard (1991) ebenfalls von der Vorstellung eines regelbasierten Ansatzes vollständig ab und entwickelte das Konzept der „Constraint-Based Generalization.“ Hierbei wird nur noch beschrieben, wie das Ergebnis aussehen und nicht wie es erreicht werden soll. Basierend darauf wurde im Zuge des EU-Projektes „AGENT“ (Automated Generalisation New Technology) der Ansatz Agenten-basierter Generalisierung (Lamy et al. 1999) aufgegriffen. Jedes einzelne Objekt wird von einem spezifischen Agenten kontrolliert. Die einzelnen Agenten versuchen die Umsetzung eigener Optimierungsregeln zu maximieren.

Auch wenn die bisherigen Ansätze und Umsetzungen alle sehr vielversprechend sind, wird eine vollständige Automatisierung der Generalisierung in der Praxis bisher nicht angewendet. Laut Foerster (2010) liegt dies größtenteils darin begründet, dass alle europäischen Institutionen eigenständig agieren. Für ihn liegt der Schlüssel im internationalen Austausch von Wissen. Anfang des 21. Jahrhunderts, mit Fortschreiten des Internetzeitalters, begann die Entwicklung von Web Services, wodurch dieser Forderung nachgegangen wurde (vgl. Kapitel 2.2.2). Infolgedessen entstand das Konzept der „Web Generalization Services“ (Edwards et al. 2005, Neun & Burghardt 2005, Foerster et al. 2008, Foerster 2010). Generalisierung wird in diesem Fall nicht mehr als lokaler Vorgang verstanden, der durch bestimmte Software verarbeitet wird. Vielmehr findet Generalisierung innerhalb einer Gemeinschaft statt, die zusammen Lösungen entwickelt und diese zentral zur Verfügung stellt.

„Web Generalization Services“ werden in drei Komplexitätsabstufungen unterteilt (Abbildung 2.12). *Unterstützende Generalisierungsservices* („generalization support services“) stellen grundlegende Funktionalitäten dar, die zur Datenvorverarbeitung oder -bereinigung notwendig sind (z.B.: Pufferung, Delauney-Triangulation). Die Verwaltung eines MRDB-Systems fällt ebenfalls darunter. Auf der nächsten Stufe stehen *Services für Generalisierungsoperatoren* („generalization operator services“). Hierbei werden reine Funk-

tionen (vgl. McMaster & Shea 1992) zur Verfügung gestellt, die zur Generalisierung bestimmter Geometrien angewendet werden können. *Services zur Steuerung des Generalisierungsprozesses* („generalization services“) sind übergeordnete Services zur Kontrolle und Orchestrierung der untergeordneten Services. Zusätzlich können



damit entstandene Ergebnisse automatisch evaluiert werden. Es könnte eine vollständige Automatisierung erzielt werden, indem die Steuerungsservices ohne äußeren Einfluss agieren.

2.3.2 Generalisierungsoperatoren

Generalisierungsoperatoren sind ein immer wiederkehrendes Element zahlreicher konzeptioneller Modelle und beschreiben grundlegende Aktionen, die zu einer grafisch veränderten Darstellung von Objekten führt. Gerade in der frühen Phase der Formalisierung kartographischer Generalisierung lag der Fokus eher auf einer Klassifizierung dieser fundamentalen Elemente. McMaster & Shea (1992) haben, im Kontext digitaler Generalisierung, Generalisierungsoperatoren erstmals sehr detailliert klassifiziert (vgl. Abbildung 2.10). In Folge finden sich zahlreiche Ansätze sinnvoller Unterteilung. In Abhängigkeit von der Herangehensweise können unterschiedliche Klassifikationen angestellt werden, sodass die Festlegung auf eine allgemeingültige Definition schwer fällt. Grundsätzlich sind Generalisierungsoperatoren stets die Beschreibung atomarer Funktionalitäten (Foerster 2010). Zu verstehen im Sinne der Ansprachen eines ganz bestimmten, exakt definierten Features. So funktionieren beispielsweise Algorithmen für Gebäudevereinfachung nur für Polygone, die Gebäude repräsentieren. Werden diese auf Polygone mit anderer Bedeutung angewendet, kann es zu drastischen Fehlern bei der Visualisierung kommen.

Die Klassifikation nach McMaster & Shea (1992) wurde für eine umfassende theoretische Betrachtung der Generalisierungsoperatoren entwickelt. Allerdings ist diese so detailliert, dass sie für allgemeine Beschreibungen schwer anwendbar ist. Aufgrund dessen wird im weiteren Verlauf die Einteilung nach Hake et al. (2002) verwendet, welche einen leichten allgemeinen Überblick ermöglicht. Innerhalb dieser Einteilung wird zwischen geometrischer, semantischer und temporaler Generalisierung unterschieden. Im einleitenden Kapitel 2.3 wurden die Begriffe geometrische und semantische Generalisierung bereits erläutert. Bei temporaler Generalisierung werden Angaben zum zeitlichen Verhalten von Objekten in die Anwendung von Generalisierungsoperatoren, vor allem im Rahmen thematischer Generalisierung (Hake et al. 2002), einbezogen. Da jene Angaben innerhalb von OpenStreetMap-Daten nicht vorhanden sind, wird dieser Aspekt nicht betrachtet. In Tabelle 2 können die Generalisierungsoperatoren nach Hake et al. (2002) nachvollzogen werden.

Geometrische Generalisierung	Semantische Generalisierung mit geometrischer Wirkung
Vereinfachung (Teilbereich: Glätten)	Zusammenfassung
Vergrößerung	Auswahl
Verdrängung (resultiert aus Vergrößern)	Klassifizierung (Teilbereiche: Heraufstufen & Signaturieren)
	Bewertung

Tabelle 2: Elementare Vorgänge kartographischer Generalisierung
(nach Hake et al. 2002, Abb. 3.63)

3 OpenStreetMap & Generalisierung – aktueller Stand

OpenStreetMap hat bereits breite Akzeptanz erfahren und eine Vielzahl von abgeleiteten Projekten hervorgebracht. In diesem Zusammenhang könnte davon ausgegangen werden, dass bereits Möglichkeiten zur Generalisierung von OpenStreetMap-Daten untersucht wurden. Inwiefern das der Fall ist, wird in diesem Kapitel eingehender betrachtet.

Dazu wird im ersten Teil untersucht, in welchem Ausmaß aktuell bereits generalisiert wird. Eine vollständige Analyse aller derzeit existierenden OSM-Visualisierungen wäre sehr aufwendig und würde das Ausmaß dieser Arbeit übersteigen. In Abbildung 3.1 ist zu erkennen, dass selbst zwischen den bekanntesten „Slippy-Maps,“ die über Mapnik und Osmarender erstellt wurden, starke Darstellungsdifferenzen vorhanden sind.

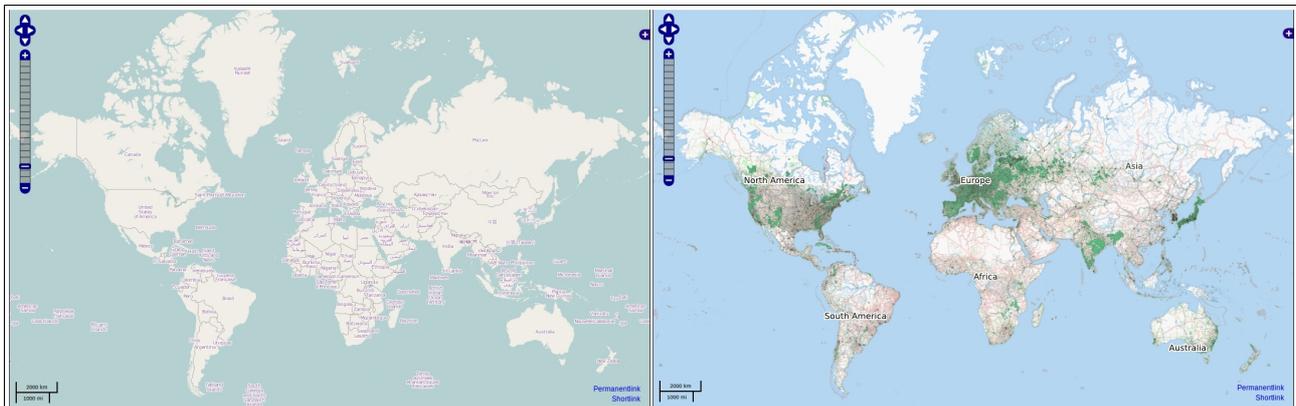


Abbildung 3.1: Weltansicht im Vergleich

Mapnik-Tile-Server (links) und Tiles@Home-Server (rechts) auf Zoomstufe 2 (Quelle: www.openstreetmap.org) – nicht maßstabsgetreu

Das Kartenbild der Osmarender-Version (Abb. 3.1, rechts) wirkt sehr unruhig und überladen, da Straßendaten, administrative Grenzen sowie Vegetationsbedeckung (wenn in der Datenbank vorhanden) in diesem Maßstab (ca. 1:115.000.000)⁷ noch abgebildet werden. Dem gegenüber steht das sehr klare und übersichtliche Ergebnis der Mapnik-Version (Abb. 3.1, links). Darüber hinaus existiert eine Vielzahl zusätzlicher Abwandlungen (z.B.: <http://opencyclemap.org>, <http://openpistemap.org>), wodurch der Aufwand einer umfassenden Analyse erheblich ansteigen würde. Dementsprechend wird lediglich eine Analyse von Ausschnitten innerhalb des gewählten Testgebietes (vgl. Kapitel 4.1.2.), unter Verwendung von Kartenausschnitten des Mapnik-Tile-Servers, betrachtet. Dies sollte repräsentativ genug sein, da die Qualität angewandter Generalisierungsmaßnahmen zwar differiert, deren Quantität jedoch in den meisten Darstellungen vergleichbar ist.

⁷ gilt nicht für Abbildung 3.1, sondern für die Anzeige als „Slippy-Map“ auf www.openstreetmap.org

Im zweiten Teil werden daraufhin einige Verbindungen zwischen bestehenden konzeptionellen Modellen der kartographischen Generalisierung und der Verarbeitung von OSM-Daten aufgezeigt.

3.1 Allgemeine Analyse und Kritik

Eine umfassende Analyse wird bereits durch die Arbeit von Zollinger (2008) geliefert. Er hat dazu einen Kriterienkatalog aufgestellt und auf dieser Basis eine Analyse, der mit Mapnik gerenderten „Slippy-Map,“ durchgeführt. Dabei kommt er zu der Schlussfolgerung, dass die Visualisierungen erwartungsgemäß einige Schwächen aufweisen, was allerdings immer im Kontext der Zielstellung von OSM sowie den Rahmenbedingungen der Kartenerstellung gesehen werden muss. Infolgedessen muss für eine Untersuchung der Anwendung von Generalisierungsfunktionalitäten zunächst betrachtet werden, ob dies überhaupt möglich ist. Wie in Kapitel 2.1.1 beschrieben, basiert die Berechnung von „Slippy-Maps“ auf einer vollständig automatisierten Implementierung ohne jegliche menschliche Interaktion. Daten werden so verarbeitet, wie sie in der Datenbank vorliegen. Lediglich bei der Erstellung von Einzelkarten ist eine Bearbeitung der Daten möglich. Da es, wie in Kapitel 2.3.1 beschrieben, bisher keine vollständig automatisierte Lösung für Generalisierung gibt, kann bereits im Vorfeld die These aufgestellt werden, dass ausschließlich elementare Generalisierungsvorgänge (Hake et al. 2002) bereits implementiert wurden. Zur Überprüfung der These wird, orientiert an Tabelle 2, in diesem Kapitel analysiert, inwiefern elementare Maßnahmen in der Mapnikversion auf www.openstreetmap.org implementiert wurden.



Abbildung 3.2: Gebäudepolygone werden nicht vereinfacht

Vergleich zwischen Zoomstufe 14 (links) und Zoomstufe 17 (rechts) – nicht maßstabsgetreu

Geometrische *Vereinfachung*, im Sinne einer Reduzierung der Komplexität von Objekten, ist nicht vorhanden. Beispielsweise werden in Abbildung 3.2 Gebäude in jedem Maßstab grundrisstreu dargestellt, selbst wenn diese Detailtiefe nicht wahrgenommen werden kann.

Ebenso verhält es sich für Linienglättung. In Abbildung 3.3 wird eine stark gewundene Straße konstant abgebildet, ohne maßstäbliche Anpassung der Geometrie zur besseren Lesbarkeit (Abbildung 3.3, links). Die Biegungen sind gerade noch zu erkennen, haben jedoch keine Relevanz und müssten demnach längst geglättet worden sein.

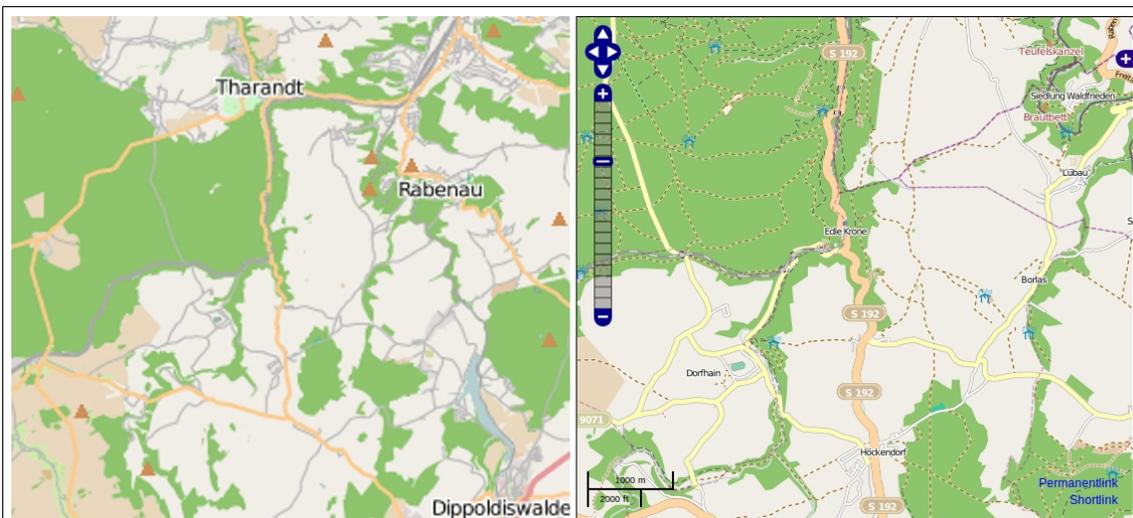


Abbildung 3.3: Fehlende Linienglättung

Vergleich zwischen Zoomstufe 11 (links) (vergrößert um Faktor 2) und Zoomstufe 13 (rechts) – nicht maßstabsgetreu

Vergrößerung, im Sinne von Überhaltung, ist einer der elementarsten Vorgänge der Kartographie und tritt selbstverständlich auf. Dies ist sehr einfach zu realisieren, da Straßen ab einem bestimmten Maßstab nicht mehr proportional dargestellt werden können. In Abbildung 3.4 ist allerdings zu erkennen, dass Überhaltung auch indirekt entstehen kann. Hierbei handelt es sich um Straßen mit einer Abtrennung. In Zoomstufe 18 (Abbildung 3.4, rechts) werden sie korrekt dargestellt, wohingegen die Visualisierung in den folgenden Maßstäben immer mangelhafter wird, bis es so scheint, als wären diese vergrößert dargestellt (Abbildung 3.4, links). Tatsächlich werden die Straßen jedoch noch einzeln abgebildet. Dabei werden grafische Mindestabstände drastisch unterschritten, wodurch das menschliche Auge die Objekte nicht mehr differenziert wahrnehmen kann. Folglich werden *Verdrängung* und *Zusammenfassung* nicht angewendet. Die Abwesenheit des Vorganges

Zusammenfassung wird ebenso in Abbildung 3.2 deutlich, da einzelne Gebäudegeometrien nicht zusammengefasst werden. Dies kann lediglich indirekt realisiert werden, wenn in der Datenbank vorgefertigte Geometrien der Siedlungsflächen vorhanden sind.

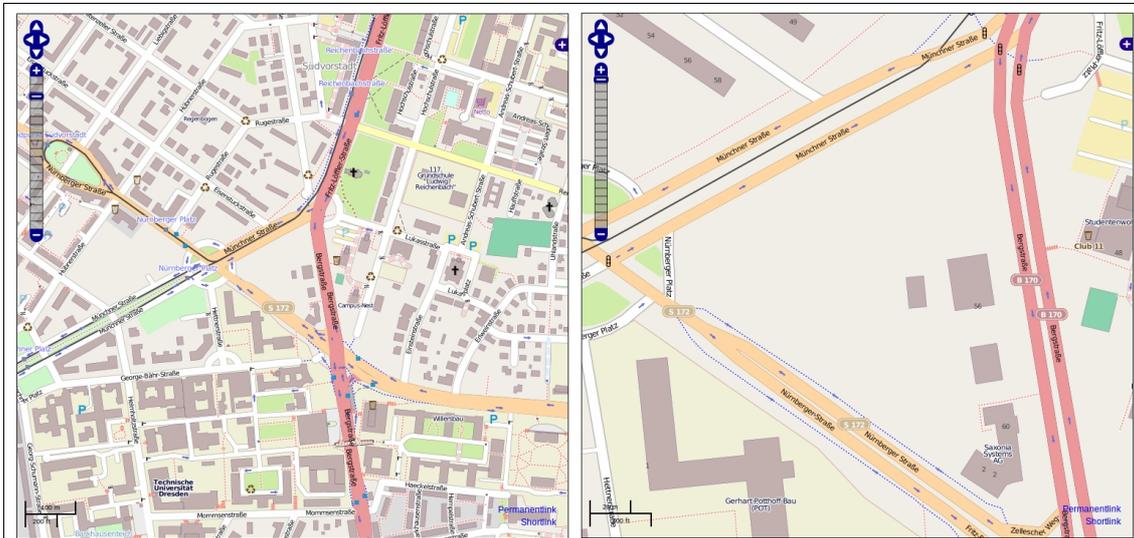


Abbildung 3.4: Fehlen von Zusammenfassung und Verdrängung
Vergleich zwischen Zoomstufe 16 (links) und Zoomstufe 18 (rechts) – nicht maßstabsgetreu

Auswahl ist eine Generalisierungsmaßnahme, ohne die eine Umsetzung der „Slippy-Maps“ nicht vorstellbar ist. Denn es würde im Umkehrschluss bedeuten, dass in jedem Maßstabsbereich alle verfügbaren Daten angezeigt werden. Allerdings weist die Umsetzung noch einiges Optimierungspotential auf. Zollinger (2008) kritisiert den sprunghaften Wechsel dargestellter Informationen von einer Zoomstufe auf die nächste. Ein Kritikpunkt, der kaum korrigiert werden kann, da dieser Umstand auf die allgemeine Umsetzung von „Slippy-Maps“ mittels Kacheln zurückzuführen ist. Verbesserungswürdig ist allerdings die Anwendung von Auswahlparametern. Es kommt häufig vor, dass Objekte in kleinen Maßstabsbereichen angezeigt werden, in denen sie bereits hätten entfernt werden müssen (vgl. Abbildung 3.1 & 3.2). Überdies werden Objekte lediglich in Abhängigkeit vom Darstellungsmaßstab und deren Semantik ausgewählt. Zusätzliche Auswahlnormen (Töpfer 1979), wie in Bezug auf Dichte, Mindestmaße oder Relevanz, sind in keiner Form implementiert.

Klassifizierung setzt sich aus zwei Bereichen zusammen. *Signaturierung* wird zwar teilweise angewandt, die Bereiche, in denen sie fehlt, überwiegen allerdings. In Abbildung 3.4 (links) werden beispielsweise drei Kirchen dargestellt, die bei guter Signaturierung auch in Abbildung 3.2 (links) erkennbar sein müssten. Das ist jedoch nicht möglich, da sie nach

wie vor grundrisstreu dargestellt werden. Der zweite Bereich, *Heraufstufung* von flächenhaften Ausprägungen, konnte innerhalb des Testgebietes nicht gefunden werden. Zollinger (2008) beschreibt dies allerdings in seiner Arbeit. Theoretisch wäre die Umsetzung nicht schwierig, da ähnliche Areale (z.B.: unterschiedliche Waldklassen) ab, einem bestimmten Maßstab, in einer übergeordneten Klasse dargestellt werden könnten. *Bewertung* von Objekten, also Betonung oder Minderung, ist im Testgebiet nicht zu finden.

Im Ergebnis bleibt festzuhalten, dass derzeit lediglich elementare Generalisierungsvorgänge (Hake et al. 2002) implementiert wurden, die singulär angewendet werden können und keine Effekte erzeugen, die eine wechselseitige Abstimmung erfordern. Ob sich Verbindungen zu allgemeinen konzeptionellen Modellen der kartographischen Generalisierung finden lassen, wird im folgenden Kapitel untersucht.

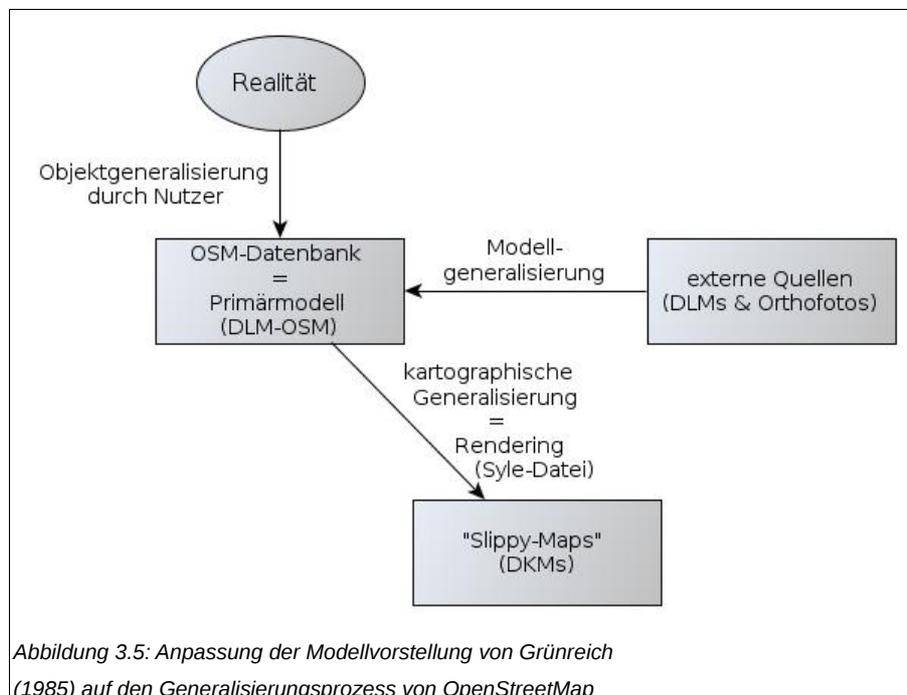
3.2 OSM & konzeptionelle Modelle

Beim Einstieg in die Thematik erweist es sich als vorteilhaft nach Ähnlichkeiten zur Modellvorstellung von Grünreich (vgl. Abbildung 2.11) zu suchen, da es ebenso als Grundlage des kartographischen Kommunikationsnetzes (Hake et al. 2002) gesehen wird.

In diesem Zusammenhang kann die zentrale OSM-Datenbank als Primärmodell verstanden werden, denn alle vorhandenen geographischen Informationen sind darin in höchster Detailtiefe enthalten. Folglich könnte es als „digitales Landschaftsmodell von OpenStreetMap“ (DLM-OSM) bezeichnet werden. In diesem Kontext kann das Einpflegen von nutzergenerierten Daten als Objektgeneralisierung verstanden werden. Für den Import von Daten aus externen Quellen (<http://wiki.openstreetmap.org/wiki/Import>) sowie für die Ableitung von Informationen aus Satellitenbildern tritt allerdings das erste Problem in der eins zu eins Übertragung des Modells auf. Formal lässt es sich als Modellgeneralisierung verstehen, da Daten aus einem vorhandenen DLM abgeleitet werden. Es entsteht daraus jedoch kein Sekundärmodell im Sinne des allgemeinen kartographischen Kommunikationsprozesses. Vielmehr wird das DLM-OSM um Daten eines externen DLM erweitert.

Als kartographische Generalisierung müsste theoretisch die reine Berechnung von Visualisierungen aus den OSM-Daten bezeichnet werden. Eine vollständige Übertragbarkeit kann jedoch nicht gewährleistet werden. In der konzeptionellen Modellvorstellung nach Grünreich (1985) wird zwischen Modellgeneralisierung, der Ableitung von DLMs unterschiedlicher Maßstabbereiche, und kartographischer Generalisierung, der Visualisierung von DLMs, unterschieden. Die beiden genannten Prozesse werden in der Verar-

beitung von OSM-Daten auf einen Vorgang reduziert, das Rendering. Die Durchführung desselben erfolgt in Abhängigkeit von den Spezifikationen innerhalb der Style-Datei⁸. Darin ist spezifiziert, welche Objekte und wie diese in Abhängigkeit vom Maßstab visualisiert werden. Das bedeutet, für jeden Maßstabsbereich bestehen exakte Definitionen über abzubildende Informationen und Symbolisierung dieser. Demnach wird Modellgeneralisierung und kartographische Generalisierung durch die abstrakten Angaben innerhalb der Style-Datei realisiert. Dies gilt vor allem für die Berechnung von „Slippy-Maps,“ da für die Berechnung von Einzelkarten, zumindest theoretisch, spezielle Style-Dateien eines bestimmten Maßstabes erstellt werden können. Zur Erzielung optimaler Generalisierungsergebnisse durch die Erstellung komplexer maßstabsunabhängiger Style-Dateien ist jedoch ein hohes Maß an kognitiver Leistung erforderlich, die im Vorfeld geleistet werden muss. Zur Erstellung dieser Style-Dateien existiert momentan keine grafische Benutzeroberfläche, weshalb derzeit sehr wenige Generalisierungsvorgänge vorhanden sind (vgl. Kapitel 3.1). In Abbildung 3.5 ist dargestellt, wie die Anpassung des konzeptionellen Modells nach Grünreich (1985) auf den derzeit implementierten Generalisierungsprozess von OSM aussehen würde.



8 Umsetzung ist abhängig vom Renderingprogramm

Zusätzlich könnte die OSM-Datenbank ebenfalls als Datenbank mit multiplen Repräsentationen (MRDB - Kilpeläinen 1992) bezeichnet werden. Deutlich wird der Sachverhalt am Beispiel von Siedlungsgebieten, da teilweise Geometrien von Gebäuden und Siedlungsflächen in der Datenbank vorhanden sind. Dementsprechend können Siedlungsgebiete, in Abhängigkeit vom Maßstab, unterschiedlich repräsentiert werden. Anzumerken ist, dass dies nicht für alle Gebiete der Erde zutrifft. Schließlich werden die Daten von Communitymitgliedern eingestellt, wodurch keine konstante Datendichte aller Gebiete gewährleistet werden kann. Das konzeptionelle Modell (MRDB) wird jedoch nur eingeschränkt umgesetzt, wie in Abbildung 3.2 zu erkennen ist. Es werden beide Repräsentationsformen gleichzeitig abgebildet. Beispielsweise sind in der rechten Karte Gebäude als Grundrisse, Siedlungsflächen in grau und das Areal der TU-Dresden in hellgrün zu finden. In der linken Karte ist eine Darstellung der Gebäude überflüssig, wodurch ein sehr dichtes Kartenbild entsteht. Demnach wird deutlich, dass das Potential zur Umsetzung einer MRDB in Ansätzen vorhanden ist. Die daraus resultierenden Möglichkeiten werden indessen nicht vollständig ausgenutzt.

Überdies lassen sich für den derzeitigen Verarbeitungsprozess von OSM-Daten keine Verbindungen zu anderen klassischen Konzepten ziehen. Begründet liegt der Sachverhalt sicherlich darin, dass die Notwendigkeit von Generalisierung bisher zu wenig beachtet wird. Obwohl Steve Chilton bereits feststellte, dass Generalisierung eines der drei klassischen Probleme bei der Ausgabe computergenerierter Karten ist (Chilton 2010). Der Fokus liegt vorrangig auf der Optimierung der automatischen Prozessierung der Daten, gerade im Hinblick auf „*Slippy-Maps*.“ Während einer kartographisch korrekten Darstellung von Informationen eine untergeordnete Bedeutung zugeordnet wird.

4 Theoretische Überlegungen

In Kapitel 3 wurde bereits deutlich, dass für den aktuellen Generalisierungsgrad bei der Verarbeitung von OpenStreetMap-Daten einige Schwächen auftreten. Deshalb werden in diesem Kapitel verschiedene theoretische Möglichkeiten zur Optimierung vorhandener Generalisierungsmaßnahmen sowie zur Einbindung zusätzlicher Funktionalitäten untersucht.

Während der Untersuchungen wurde deutlich, dass Implementierungen am vorteilhaftesten über Mapnik zu realisieren sind. Damit ist ein vollständiger Zugriff auf den Quellcode möglich und die Verarbeitung der Daten geschieht wesentlich transparenter, als bei Osmarender oder Maperative. Berechnungen mittels Osmarender hängen grundlegend vom verwendeten XSLT-Prozessor ab. Einheitliche Implementierungsansätze wären damit kaum umsetzbar, da hierfür zahlreiche Möglichkeiten verfügbar sind. Bei Mapnik hingegen, verwendet jeder Nutzer exakt die gleiche Software. Maperative befindet sich zum aktuellen Zeitpunkt noch in der Entwicklung. Deshalb und insbesondere da eine umfassende Dokumentation noch aussteht, ist es problematisch die Software vollständig zu erproben und Implementierungsvorschläge zu entwerfen. Aufgrund dessen liegt der Fokus dieser Arbeit ausschließlich auf der Entwicklung von Lösungen zur Einbindung in Mapnik.

In diesem Sinne wird im ersten Teil das Potential zur Einbindung von Web Generalisierungsservices in Form des WebGen-WPS ermittelt. Daran schließt sich eine Analyse möglicher Generalisierungsmaßnahmen durch Verwendung von PostGIS-Funktionen an. Im letzten Unterkapitel werden daraufhin die Chancen einer Generalisierungscommunity, innerhalb des OpenStreetMap-Projektes, verdeutlicht. Da es sich hierbei um eine abstrakte Lösung zur Optimierung von Generalisierung innerhalb von OSM handelt, ist dieser Ansatz selbstverständlich nicht auf eine reine Umsetzung mittels Mapnik beschränkt.

4.1 Einbindung des WebGen-WPS

In Kapitel 2.2.2 wurden bereits grundlegende Funktionen des Generalisierungsdienstes WebGen-WPS aufgezeigt. Dabei ist dessen Eignung zur Anwendung elementarer geometrischer Generalisierungsfunktionen auf räumliche Daten deutlich geworden. Eine Einbindung des WebGen-WPS in die Verarbeitung von OSM-Daten basiert auf dem grundlegenden Gedanken, dass diese Daten an den WPS gesendet, dort verarbeitet und anschließend wieder in den Renderingvorgang eingebunden werden.

Es ergeben sich zwei elementare Optionen, wie dies umgesetzt werden kann. Zum Einen kann WebGen-WPS direkt in den automatischen Renderingprozess eingebunden werden, sodass weiterhin keine Benutzereingriffe notwendig sind. Zum Anderen ist ebenfalls eine Vorverarbeitung der räumlichen Informationen möglich, welche anschließend in einer MRDB (Kilpeläinen 1992) abgespeichert werden.

4.1.1 Direkteinbindung des WebGen-WPS

Durch Implementierung von Funktionen, über welche der WebGen-WPS aufgerufen wird, ist eine direkte Einbindung dieses Web Services umsetzbar. Dazu werden Erweiterungen für Mapnik benötigt, die es ermöglichen, die räumlichen Informationen während der Berechnung abzufangen, an den Server zu senden und die resultierenden Daten in den regulären Ablauf zurück zu speichern.

Zunächst ist es notwendig, Elemente des Softwarecodes zu finden, in welchen das Eingreifen in den Renderingvorgang sowie Abfangen von Daten möglich ist. Inwiefern dies für Mapnik praktisch gelöst werden kann, wird in Kapitel 5.2.1 detailliert untersucht. Diesbezüglich ergibt sich die Frage, in welcher Form die Daten verwendet und an den Server gesendet werden? Einerseits besteht die Möglichkeit vollständige Datensätze an den Server zu senden, andererseits ist jedoch auch die Verarbeitung einzelner Features realisierbar. Der Vorteil beim Senden eines kompletten Datensatzes liegt darin, dass räumliche Beziehungen von Features jederzeit abgefragt werden können. Insbesondere, wenn durch geometrische Veränderungen räumliche Konflikte (z.B.: Überschneidungen) zwischen Objekten entstehen, können diese direkt lokalisiert und korrigiert werden. Entstehen jedoch Fehler am Server, können Probleme auftreten. Fehler, die im Rahmen der Verarbeitung ganzer Datensätze auftreten, sind sehr schwierig identifizierbar und korrigierbar.

Einzelnen Features kann ein auftretendes Problem direkt zugeordnet werden. Das Abfangen von Fehlern, die während der Verarbeitung auf dem Generalisierungsserver entstehen, ist ein allgemeines Problem, das für eine solche Umsetzung auftritt. Diese Problematik wird in Kapitel 5.2.1 näher beschrieben, da im Kontext der praktischen Durchführbarkeit dieser Implementierung eine differenzierte Beschreibung der daraus resultierenden Einschränkungen möglich ist.

Der „Execute“-Befehl, inklusive Daten, muss in Form von HTTP/POST an den WebGen-WPS gesendet werden (vgl. Kapitel 2.2.1). Dementsprechend wird das Anlegen einer XML-Datei sowie das Senden dieser an den Server notwendig. Solche Serveraufrufe können über das kostenlos erhältliche Programm „cURL“ und der dazugehörigen Programm-bibliothek „libcurl“ (<http://curl.haxx.se/>) realisiert werden. Für die Umsetzung in Mapnik ist besonders vorteilhaft, dass diese Bibliothek über C++ direkt genutzt werden kann. Zusätzlich ist jedoch ebenfalls, mit Hilfe des Interfaces „PycURL“, über die Programmiersprache Python der Zugriff auf die Programm-bibliothek („libcurl“) möglich. Für die Analyse der praktischen Durchführbarkeit wurde ein Serveraufruf über „libcurl“ in C++ implementiert. Der entsprechende Programmcode ist in Anhang C nachvollziehbar.

Innerhalb der benötigten XML-Datei müssen sämtliche Informationen enthalten sein, die für eine Auswertung am Server notwendig sind. Darunter zählen unter anderem der „Execute“-Befehl, zum Starten der Verarbeitung, der „Identifizier“ der gewählten Funktion, die Eingangsdaten sowie die Form der Serverantwort. Diese Angaben unterliegen den Spezifikationen von OGC für Web Processing Services (OGC 2007).

Nach erfolgreicher Prozessierung wird vom Server ein XML-Antwortdokument zurückgeliefert, welches die resultierenden Daten enthält. Sowohl für das Senden als auch für die Auswertung des Ergebnisses ist es notwendig, dass zusätzliche Prozeduren implementiert werden.

Für eine exakte Angabe, welche Geometrien generalisiert werden sollen, ist grundsätzlich die Implementierung eines zusätzlichen Kennzeichens („Generalize“) im XML-basierten Stylefile notwendig. Dieses könnte, ähnlich dem Kennzeichen „Filter“, innerhalb der Regeln („Rules“), als boolescher Operator eingesetzt werden. Zusätzlich müssen, im Bereich des Kennzeichens „Generalize“, weitere Kennzeichen implementiert werden, worüber Generalisierungsgrad sowie -funktionalität spezifiziert werden.

Darüber hinaus ergeben sich für die Einbindung von Serveraufrufen zwei im Grad der Komplexität differierende Möglichkeiten. Die Implementierung eines statischen Serveraufrufs ist als „simple Umsetzung“ zu verstehen. Demnach werden im Vorfeld die vom Nutzer

angestrebten Generalisierungsfunktionen exakt vordefiniert. Infolgedessen müssen, in einem vorbereitendem Schritt, am Server verfügbare Funktionen sowie dafür zusätzlich notwendige Attribute ermittelt werden. Daraus resultierend müssten dann die korrespondierenden HTTP/GET-Befehle im Code bereits vorhanden sein. Diese Umsetzung funktioniert nur innerhalb eines exakt vorgegebenen Rahmens.

Anzuraten wäre eine „komplexe Implementierung,“ bei welcher der Serveraufruf dynamisch umgesetzt wird, sodass über „GetCapabilities“ automatisch alle verfügbaren Funktionen und die dafür notwendigen Attribute („DescribeProcess“) abgefragt werden. In Abhängigkeit von den Angaben innerhalb des „Generalize“ - Kennzeichens würden daraufhin verschiedene Operationen angesprochen. Daraus resultiert jedoch ein wesentlich höherer Implementierungsaufwand der benötigten Erweiterungen für Mapnik, als für eine „simple Implementierung.“ Andererseits würde dies den Umfang der Vorarbeiten, die jeder Nutzer durchführen muss, wesentlich reduzieren. Konsequenterweise ist auf lange Sicht, die direkte Einbindung des WebGen-WPS in einer „komplexen Implementierung“ anzustreben.

Zur direkten Einbindung des WebGen-WPS werden dementsprechend zahlreiche Modifikationen in Mapnik notwendig, deren vollständige Umsetzung den Umfang der vorliegenden Arbeit übersteigen würde. Aus diesem Grund werden in Kapitel 5.2.1 lediglich die Teile des Softwarecodes von Mapnik dargestellt und untersucht, welche für eine Implementierung praktikabel sind.

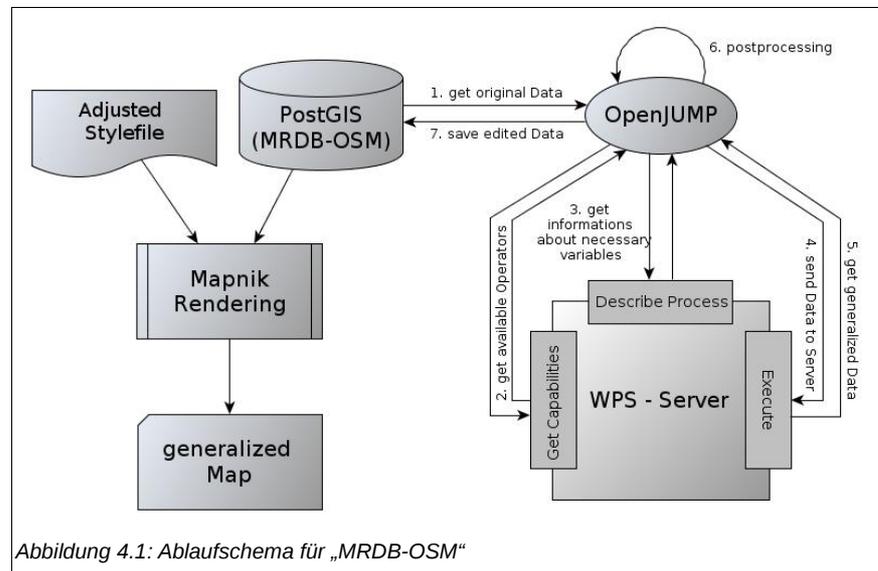
4.1.2 Einbindung von WebGen-WPS für „MRDB-OSM“

Im Kontrast zur vorangegangenen Umsetzung werden OSM-Daten hierbei nicht während der Renderinglaufzeit, sondern bereits in einem vorbereitendem Arbeitsschritt generalisiert. Diesbezüglich wird der Ansatz von Kilpeläinen (1992) aufgegriffen und das Anlegen einer PostGIS-Datenbank mit multiplen Repräsentationen für OpenStreetMap-Daten („MRDB-OSM“) implementiert. Somit können verschiedene Generalisierungsstufen von Objekten unterschiedlicher Maßstabsbereiche vorab berechnet und in einzelnen Tabellen abgespeichert werden. Diese Implementierung ist über Mapnik sehr leicht durchzuführen, da alle grundsätzlichen Voraussetzungen bereits vorhanden sind.

Bei der Verarbeitung mit Mapnik werden OSM-Daten zunächst in einer PostGIS-Datenbank abgespeichert. Damit ist es möglich, Daten in ein Geoinformationssystem zu laden und dort zu verarbeiten. Insbesondere da für das frei erhältliche GIS bereits ein Plug-in zum Aufruf von WebGen-WPS existiert (vgl. Kapitel 2.2.2), wird die Verwendung von OpenJUMP vorgeschlagen. Wenn WPS-Aufrufe durchführbar sind, kann gleichfalls ein alternatives GIS verwendet werden. Empfehlenswert ist die Verwendung von OpenJUMP, da hiermit Daten aus PostGIS gelesen und in der Datenbank abgespeichert werden können.

In Abbildung 4.1 ist zu sehen, wie das Schema dieser Variante aussehen könnte. Es werden, über OpenJUMP, spezifische Objekte aus der Datenbank gelesen und an den WPS-Server gesendet.

Über das WPS-Plug-in können in diesem Zusammenhang alle verfügbaren Funktionen abgerufen und die zusätzlich notwendigen Variablen definiert werden. Nach erfolgreicher Verarbeitung, ist die Nachbearbeitung resultierender Objekte im GIS durch-



föhrbar, was besonders bei Auftreten von Konflikten notwendig ist. Abschließend werden die resultierenden Daten in der Datenbank als spezifische Tabelle abgespeichert. Um Redundanzen zu vermeiden, wird dabei lediglich eine Geometriespalte sowie eine Spalte mit der Identifikationsnummer („*osm_id*“) angelegt. Über dieses Attribut sind anschließend korrespondierende Objekte untereinander referenzierbar. Dieser Vorgang kann für verschiedene Arten von Features, Generalisierungsfunktionen und angestrebtem Maßstab durchgeführt werden.

Die einzige zusätzliche Modifikation, die durchgeführt werden muss, betrifft das Stylefile. Dort müssen die zusätzlichen Tabellen in der Datenabfrage innerhalb des Kennzeichens „*Layer*“ eingebunden werden. Praktische Umsetzungen des vorgestellten Ansatzes sowie erhaltene Resultate werden in Kapitel 5.2.2 näher untersucht.

4.2 PostGIS-Funktionen

PostGIS ist eine Erweiterung der objekt-relationalen Datenbank PostgreSQL. Sie bietet die Möglichkeit, geographische Daten in einem freien Datenbanksystem zu verarbeiten und liefert gleichzeitig wichtige Funktionen, die zur Verarbeitung von räumlichen Informationen nötig sind. Da beim Rendering mit Mapnik Daten aus der PostGIS-Datenbank abgerufen werden, ist es möglich diese Funktionen auch im Sinne von Generalisierung zu nutzen. Durch Einbindung elementarer Generalisierungsvorgänge könnte der derzeit sehr geringe Generalisierungsgrad von OSM-Karten gesteigert werden. Insbesondere für die Unterstützung von bisher nicht implementierten Vorgängen wie Verdrängung, Zusammenfassung und Klassifizierung sind Anwendungen vorstellbar.

Die im Folgenden betrachteten Funktionen basieren auf PostGIS-Version 1.5.2-2, welches in der PostgreSQL-Version 8.4 eingebunden wurde. Es werden lediglich die Namen verwendbarer Funktionen genannt, da deren spezifische Inhalte in der PostGIS-Dokumentation (<http://postgis.refractions.net/documentation/manual-1.5>) nachvollziehbar sind.

Für PostGIS-Funktionen existieren drei Optionen zur Nutzung. Zum Einen indem sie direkt in die Abfrage eingebunden werden, welche innerhalb des Stylefiles zum Aufruf einer Datenquelle integriert ist (vgl. Abbildung 2.5). Dies ist allerdings nur für sehr einfache geometrische Funktionen umsetzbar. Des Weiteren ist es vorstellbar, komplexe Verarbeitungsschritte in einem vorangestellten Verarbeitungsschritt auszuführen und resultierende Daten zusätzlich in der Datenbank abzuspeichern. Es ist erdenklich, dies in einer Abwandlung von MRDB (Kilpeläinen 1992) zu lösen. Die anfänglich beschriebene Möglichkeit ist ohne Weiteres in den automatischen Verarbeitungsprozess integrierbar. Dies ist für komplexe Prozesse nicht möglich. Aus diesem Grund wird, als dritte Möglichkeit, das Ablegen komplexer Verfahrensabläufe in Form spezifisch editierter PostGIS-Funktionen im Datenbanksystem vorgeschlagen. Daraus folgt, dass aufwendige Vorverarbeitungen entfallen und komplexe Prozesse simpel in das Stylefile integrierbar wären.

Für Klassifikation, im Sinne einer Zusammenlegung verschiedener polygonaler Objekte zu einem übergeordneten Polygon, bietet sich die Funktion „*ST_Collect()*“ an. Diese wäre beispielsweise hilfreich, wenn abgetrennte Waldflächen, in kleinen Maßstäben, in einer einzigen Beschriftung dargestellt werden sollen. Aktuell wird Zusammenfassung von Flächen nur über die Darstellungsweise realisiert, sodass sie nur scheinbar aggregiert sind. In

Folge wird jedem einzelnen Polygon eine Beschriftung zugewiesen, was durch diese Funktionen verhindert werden könnte. Eine Möglichkeit, die bereits Steve Chilton (2010) erkannte.

Des Weiteren kann Unterschreitung von Mindestabständen identifiziert werden. Beispielsweise durch Anwendung von Abfragefunktionen räumlicher Beziehungen, wie „*ST_Distance()*“, „*ST_MaxDistance()*“ und „*ST_Intersects()*.“ Daraufhin könnten Objekte beispielsweise mittels „*ST_Collect()*“ oder „*ST_SymDifference()*“ zusammengefasst oder durch die Funktionen „*ST_Translate()*“ bzw. „*ST_Trans_Scale()*“ Verdrängungen realisiert werden.

Zusätzlich besteht das Potential, den Vorgang Auswahl, der bisher rein auf Maßstab (bzw. Zoomstufe) und Semantik basiert, durch Auswahl nach Mindestlängen zu erweitern. Hierfür eignet sich hervorragend die Funktion „*ST_Length()*“, welche allerdings nur für die Verarbeitung linearer Objekte anwendbar ist. Für flächenhafte Entitäten wird dementsprechend eine Auswahl nach Mindestgrößen benötigt. Dies wird durch die Funktion „*ST_Area()*“ zur Verfügung gestellt. In diesem Fall wäre eine spezielle PostGIS-Funktion nicht zwingend notwendig. Das Programm „Osm2pgsql“ legt in der betreffenden Datenbanktabelle bereits eine Spalte für den Flächeninhalt („*way_area*“) an. Diese Spalte wird allerdings im SQL-Variablentyp *real* abgespeichert, dessen Größe auf vier Bytes mit einer Genauigkeit von sechs Dezimalstellen begrenzt ist (www.postgresql.org/docs/8.2/static/datatype-numeric.html). Aus diesem Grund kann es zu Rundungsfehlern kommen, weshalb vielmehr auf die Funktion „*ST_Area()*“ zurückgegriffen werden sollte.

Ebenso besteht die Möglichkeit zur Vereinfachung von Geometrien durch Einbindung der Funktionen „*ST_Simplify()*“ und „*ST_SimplifyPreserveTopology()*.“ „*ST_Simplify()*“ ist eine reine Umsetzung des weit verbreiteten Douglas-Peucker-Algorithmus (Douglas & Peucker 1973). Mittels „*ST_SimplifyPreserveTopology()*“ werden zusätzlich topologische Zusammenhänge validiert. In den spezifischen Dokumentationen⁹ wird der resultierende Effekt jedoch nicht ausreichend beschrieben. Sie beschreibt lediglich, dass durch zusätzliche topologische Auswertung die Ableitung ungültiger Geometrien vermieden wird. Dies soll, laut Dokumentation, vor allem für die Ableitung vereinfachter Polygone vorteilhaft sein. Darüber hinaus wurde der Douglas-Peucker-Algorithmus prinzipiell nur für Linienver-

⁹ „*ST_Simplify()*“ - http://postgis.refrations.net/documentation/manual-1.5/ST_Simplify.html (31.03.2011)
„*ST_SimplifyPreserveTopology()*“ - http://postgis.refrations.net/documentation/manual-1.5/ST_SimplifyPreserveTopology.html (31.03.2011)

einfachungen entwickelt. Aus diesen Gründen wird zunächst davon ausgegangen, dass „*ST_Simplify()*“ auf Linien und „*ST_SimplifyPreserveTopology()*“ auf Polygone anwendbar ist.

Abschließend ist festzustellen, dass im beschriebenen Bereich unterschiedlich starkes, theoretisches Potential zur Umsetzung von Generalisierungsregeln bei der Visualisierung von OSM-Daten vorhanden ist. Inwiefern diese theoretischen Möglichkeiten umsetzbar sind, wird durch praktische Anwendungen in Kapitel 5.3 analysiert.

4.3 OpenStreetMap - Generalisierungscommunity

In den vorangegangenen Kapiteln zeigten sich zahlreiche Mängel in der Anwendung von Generalisierung, die nicht durch das Untersuchungsgebiet der vorgegebenen Aufgabenstellung abgedeckt beziehungsweise gelöst werden konnten. Als weiterer Lösungsansatz soll zusätzlich die Entwicklung einer Generalisierungscommunity für OpenStreetMap aufgezeigt werden. Es bietet sich somit die Option, alle Bereiche der kartographischen Generalisierung in Bezug auf OSM-Kartendarstellungen zentral zu sammeln und stetig weiter zu entwickeln.

Ebenso wäre dadurch Wissen um klassische Generalisierungsmethoden jedem Anwender, innerhalb der OSM-Gemeinschaft, besser zugänglich. Schwierig gestaltet sich vor allem das Treffen allgemeingültiger Aussagen, da sich für das gesamte Projekt vielschichtige Möglichkeiten ergeben. Dementsprechend sind verschiedene Faktoren zu beachten, die Einfluss auf eine optimale Darstellungsform von OSM-Daten haben (Abbildung 4.2).

- *Maßstab (bzw. Zoomstufe)*
- *Semantik der Objekte*
- *Gebiet (Flachland vs. Hochgebirge)*
- *Generalisierungsgrad (niedrig vs. hoch)*
- *Datendichte (niedrig vs. hoch)*
- *Kartenthema (z.B.: Allgemein, Fahrradkarte, Pistenkarte, etc.)*
- *Ausgabemedium (Bildschirm vs. Druck)*
- *Möglichkeiten der Renderingsoftware*

Abbildung 4.2: Einflussfaktoren auf optimale Darstellung von OpenStreetMap-Daten

In Bezug auf diese Faktoren könnten innerhalb der Gemeinschaft Empfehlungen gegeben, diskutiert und jedem Nutzer zugänglich gemacht werden.

Zur Umsetzung bieten sich verschiedene Varianten an. Zum Einen in Form eines festen Regelkatalogs. Vorteilhaft könnte dies zur Optimierung der Style-Dateien sein, woraus eine Optimierung bereits vorhandener Generalisierungsmaßnahmen resultieren würde. So gibt es zum Beispiel in der Optimierung der Auswahlkriterien großen Bedarf (vgl. Kapitel 3.1). Ebenso könnten allgemeine Regeln der kartographischen Kartengestaltung, die für OSM anwendbar sind, darin aufgeführt werden. Denn kartographische Generalisierung steht immer auch im direkten Zusammenhang mit den Grundregeln der Kartengestaltung, weshalb im besten Fall eine direkte Verknüpfung implementiert wird. Fachleute könnten diesen festen Regelkatalog entwickeln und anschließend bereitstellen. Einerseits wären diese Informationen nahezu uneingeschränkt verlässlich, da sie von Experten zusammengestellt wurden. Andererseits bliebe jedoch die Entwicklung von Regeln ausschließlich auf einen kleinen Personenkreis beschränkt.

Als weitere Option bietet sich eine Umsetzung in Form eines frei erweiterbaren Wiki an. Daran könnte jeder Nutzer mitwirken und eigene Regeln und Empfehlungen aufstellen. Hierbei ist natürlich auf die Selbstkontrolle einer solchen Gemeinschaft zu setzen, da nicht jede verfügbare Information von einer administrativen Instanz geprüft werden könnte.

Zusätzlich zu festen Regeln wäre es denkbar, Empfehlungen für verschiedene Umsetzungsmöglichkeiten zu geben, wie die Anwendung von Generalisierungsoperatoren über Web Processing Services (Kapitel 4.1). Folglich werden innerhalb der Generalisierungscommunity alle notwendigen Informationen zusammengetragen und ständig neue Entwicklungen veröffentlicht. Dementsprechend wäre die Gemeinschaft immer aktuell und bliebe flexibel für neue Entwicklungen. In diesem Sinne entstünde eine ständige kritische Auseinandersetzung mit aktuellen Umsetzungen. Für Diskussionen, sowie die Klärung spezieller Fragen, bietet sich die Aufstellung einer Mailingliste an. Innerhalb der OSM-Gemeinschaft ist das bereits gängige Praxis und funktioniert so gut, dass eine solche Umsetzung uneingeschränkt empfehlenswert ist.

Dieser Ansatz ist keine Einbindung von Generalisierungsfunktionalitäten, sondern vielmehr ein abstrakter Vorschlag, um die Entwicklungen besser voranzutreiben. Denn über solch eine Generalisierungscommunity wäre es möglich, weltweites Wissen zu sammeln und zur Anwendung zu bringen.

5 Implementierungen & Ergebnisse

Im vorangegangenen Kapitel wurden bereits einige theoretische Möglichkeiten aufgezeigt, inwiefern es möglich wäre, den aktuellen Grad der Generalisierung innerhalb des Bereiches Visualisierung von OpenStreetMap-Daten zu steigern. Infolgedessen werden in diesem Kapitel einige praktische Implementierungen veranschaulicht und daraus resultierende Ergebnisse vorgestellt. Wie bereits beschrieben, wären Umsetzungen für alle verfügbaren Renderinglösungen einerseits sehr aufwendig und andererseits nicht in allen Bereichen gleichermaßen umsetzbar. Aus diesem Grund wurden lediglich Implementierungen für Mapnik praktisch umgesetzt. Dies stellt jedoch keine nennenswerte Einschränkung dar, da Mapnik derzeit die meisten Möglichkeiten für zusätzliche Implementierungen liefert.

Zunächst werden kurz die Systemvoraussetzungen, unter welchen die Implementierungen vollzogen wurden, sowie das gewählte Testgebiet vorgestellt. Daraufhin werden Beispiele für Einbindungen des WebGen-WPS in Mapnik aufgezeigt, wobei mit der Einbindung in den automatischen Renderingprozess von Mapnik begonnen wird. Einschränkend muss allerdings angemerkt werden, dass eine vollständige Implementierung den Umfang dieser Arbeit überschritten hätte. Sodass lediglich der Quellcode auf Implementierungsmöglichkeiten untersucht wurde. Die gefundenen Möglichkeiten werden jedoch vollständig dargestellt. Daraufhin folgt die Vorstellung der praktischen Implementierung einer „MRDB-OSM“ sowie eine Analyse der resultierenden Ergebnisse. Abgeschlossen wird dieses Kapitel mit der Auswertung von Ergebnissen, die über Einbindung von PostGIS-Funktionen erzielt wurden. Da die Implementierungen den Kern der gesamten Arbeit bilden, werden sie bewusst sehr ausführlich beschrieben, um einzelne Verarbeitungsschritte exakt nachvollziehbar und reproduzierbar darzulegen.

5.1 Technische Voraussetzungen

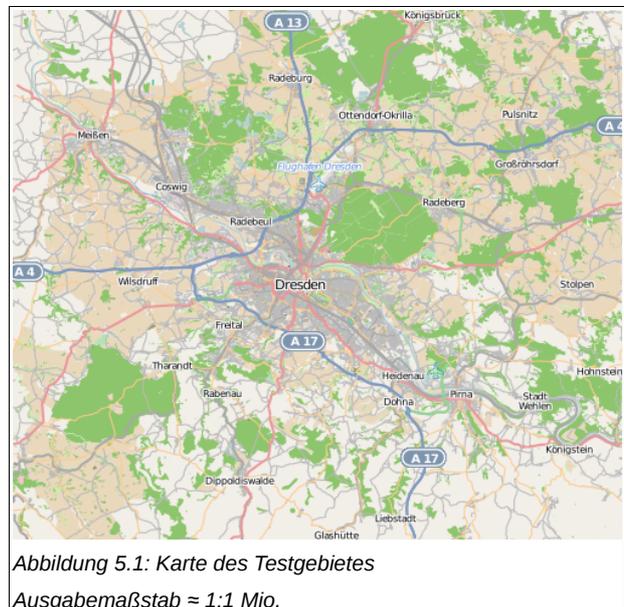
5.1.1 Systemvoraussetzungen

Hardware	
Prozessor	AMD Athlon XP 2800+
Arbeitsspeicher	1002,4 MB
Grafikkarte	ATI Radeon 9600 (256MB)
Motherboard	ASRock K7S8X
Betriebssystem	
Ubuntu 10.04 (lucid)	
Kernel Linux 2.6.32-30-generic	
GNOME 2.30.2	

Tabelle 3: Verwendete Systemvoraussetzungen bei Implementierungen

5.1.2 Testgebiet

Zur Veranschaulichung der Implementierungen wurde ein Testgebiet aus dem Großraum von Dresden gewählt (Abbildung 5.1). Es handelt sich um ein rund 7000 km² großes Gebiet mit den Rahmenkoordinaten $B_{\text{box}} = (13.381, 50.845, 14.139, 51.271)$. Solch ein großer Datensatz kann nicht direkt aus der OpenStreetMap-Datenbank, über die Exportfunktion der API auf www.openstreetmap.org, heruntergeladen werden. Deshalb wurde der Datensatz aus einem Datenauszug von Deutschland, über die Software Osmosis (<http://wiki.openstreetmap.org/wiki/Osmosis>), erstellt. Komprimiert ergibt der Datensatz eine virtuelle Größe von 13,6 Megabyte.



Folglich ist eine aussagekräftige versuchsweise Implementierung von Generalisierungsfunktionalitäten möglich. Nahezu alle Objekte einer topographischen Karte, wie Straßen- und Gewässernetz, Siedlungsgebiete, administrative Grenzen, Points of Interest

sowie Vegetations- und Siedlungsflächen, sind darin enthalten. Für die exemplarischen Implementierungen wurden daraufhin verschiedene Teilbereiche des Datensatzes verwendet, welche für das jeweilige Beispiel aussagekräftige Objekte enthalten.

5.2 Einbindung des WebGen-WPS in Mapnik

5.2.1 Einbindung in den automatischen Prozess

In Kapitel 2.1.3. wurde Mapnik bereits im Detail erläutert, wobei deutlich erkennbar ist, dass der Renderingprozess über ein spezielles Pythonskript ausgeführt wird. Der allgemeine Ablauf ist in Abbildung 5.2 skizziert. Zunächst wird eine „Mapnik-Map“ bezüglich des spezifischen Stylefiles initialisiert und daraus die finale Repräsentation der OSM-Daten mit PostGIS als Datenquelle berechnet. Zwar erfolgt die Definition der Datenquellen bereits im Stylefile, jedoch findet deren Abruf erst während des Renderingvorganges statt. In Anhang A befindet sich das vollständige Pythonskript, welches für die exemplarischen Umsetzungen verwendet wurde. Dieses ist ausschließlich zur Berechnung von Einzelkarten geeignet, da sich hierbei die einzelnen Verarbeitungsschritte wesentlich besser nachvollziehen lassen, als bei der Berechnung von Kacheln für „Slippy-Maps“.

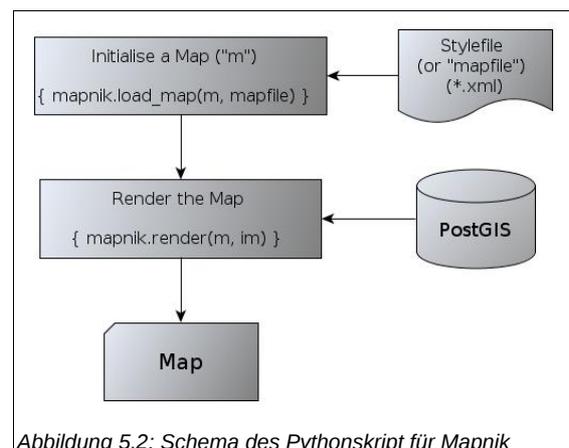


Abbildung 5.2: Schema des Pythonskript für Mapnik

Zwar erfolgt die Definition der Datenquellen bereits im Stylefile, jedoch findet deren Abruf erst während des Renderingvorganges statt. In Anhang A befindet sich das vollständige Pythonskript, welches für die exemplarischen Umsetzungen verwendet wurde. Dieses ist ausschließlich zur Berechnung von Einzelkarten geeignet, da sich hierbei die einzelnen Verarbeitungsschritte wesentlich besser nachvollziehen lassen, als bei der Berechnung von Kacheln für „Slippy-Maps“. Klassen, die zum Rendering notwendig sind, kommen sowohl bei der Erstellung von Einzelkarten, als auch bei der von „Slippy-Maps“ zur Anwendung. Dementsprechend sind hier entworfene Implementierungen uneingeschränkt auf die Berechnung von „Tiles“ anwendbar.

Wie bereits erläutert, wird es für diesen Ansatz notwendig zusätzliche Klassen zu entwerfen, die eine Kommunikation mit dem WPS-Server der WebGen-Plattform ermöglichen. Es bietet sich an, die zusätzlichen Klassen als WebGen-WPS-Plug-in für Mapnik zu implementieren. Darüber hinaus müssen zusätzliche Kennzeichen (z.B.: „Generalize“) im Stylefile definiert werden, welche Generalisierungsart und -grad vorgeben. Zum besseren Verständnis wird an erster Stelle der Ablauf der Berechnungen innerhalb des Programmcodes aufgezeigt und anschließend die Klassen des Quellcodes für mögliche Einbindungen eines WPS-Aufrufs erläutert.

5.2.1.1 Allgemeiner Programmablauf

Daten können erst abgefangen werden, wenn sie von Mapnik aus der Datenbank eingelesen wurden. Dies geschieht im Programmcode während des expliziten Renderingvorganges. Das bedeutet, wenn alle Informationen in der Karte visualisiert werden. Innerhalb des Pythonskripts wird der explizite Renderingvorgang über den Befehl „`mapnik.render(m, im)`“ aufgerufen. Dabei werden, zwischen den Klammern, die zuvor initialisierte Mapnik-Map („`m`“) sowie die Imagedatei („`im`“), in welcher die Karte ausgegeben werden soll, übergeben. Während der Initialisierung der Mapnik-Map wurde zunächst das XML-basierte Stylefile interpretiert und alle notwendigen Informationen (z.B.: „`Styles`“, „`Rules`“ & „`Layer`“) in dem Objekt abgelegt. Die wichtigsten Klassen, die das Rendering ausführen, sind „`feature_style_processor.hpp`“ sowie „`agg_renderer.cpp`.“ Über „`feature_style_processor.hpp`“ werden alle Features in Relation zu den damit verbundenen Styleinformationen gebracht. Die eigentliche Verarbeitung der Features geschieht dann in der Klasse „`agg_renderer.cpp`.“

Beim Rendering werden die Ebenen („`Layer`“) sukzessive verarbeitet. Dafür läuft zunächst eine Schleife über alle korrespondierenden „`Styles`“ der aktuellen Ebene. Daraufhin werden die notwendigen Daten aus der PostGIS-Datenbank gelesen und in einem Featureset abgespeichert. Diese Abfrage wird mit Hilfe der Klasse „`postgis.cpp`“ des mitgelieferten PostGIS-Plug-in vollzogen. Über das Featureset läuft anschließend wiederum eine Schleife, wodurch jedes Feature einzeln verarbeitet wird. Danach laufen Schleifen über alle Regeln („`Rules`“) des aktuellen „`Style`“ sowie die darin angegebenen Symbolisierungen („`Symbolizer`“). Alle gesammelten Informationen gehen abschließend in die Visualisierung (bzw. Symbolisierung) des aktuell zu verarbeitenden Features ein. Dies wird innerhalb der Klasse „`agg_renderer.cpp`“ durchgeführt. Daraufhin wird jedes Feature, in Abhängigkeit von dessen Geometrie (Punkt, Linie, Polygon) und den Symbolisierungsparametern („`Symbolizer`“) des Stylefiles, abgebildet. Die einzelnen Schritte werden so lange wiederholt bis alle Features, Regeln, Styles & Ebenen, die im Stylefile angegeben wurden, abgearbeitet sind. Der entsprechende schematische Verfahrensablauf kann in Abbildung 5.4 nachvollzogen werden.

Eine Generalisierung über WebGen-WPS könnte demnach während der Verarbeitung der Daten im „`feature_style_processor.hpp`“ oder im „`agg_renderer.cpp`“ implementiert werden. Wie in Kapitel 4.1.1 bereits beschrieben, gibt es allerdings die Optionen, Daten entweder komplett oder als einzelne Features an den WPS-Server zu senden. Einbinden lassen sich beide Möglichkeiten jedoch nur in der Klasse „`feature_style_processor.hpp`,“

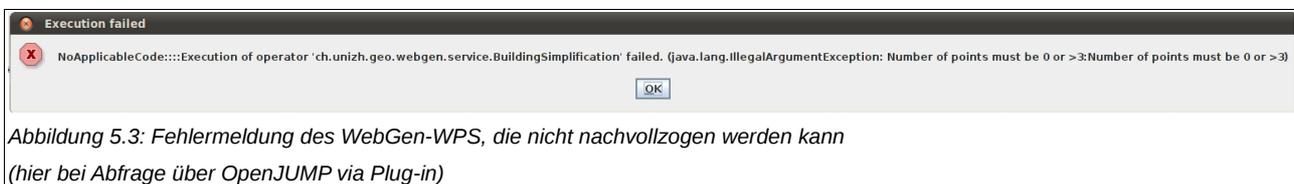
da in „*agg_renderer.cpp*“ lediglich einzelne Features verarbeitet werden. Zusätzlich sollen Generalisierungsoptionen im Stylefile implementiert werden, worüber die Art und der Grad der Generalisierung angegeben wird. Die Verknüpfung dieser Angaben mit den jeweiligen Daten ist nur in „*feature_style_processor.hpp*“ möglich, weshalb nur dort eine Implementierung empfohlen werden kann. Geometrien werden dementsprechend zunächst generalisiert und diese modifizierten Geometrien für die abschließende Visualisierung an „*agg_renderer.cpp*“ übergeben. Je nach vorgesehener Verarbeitung der Daten, müssten zusätzliche Befehle an verschiedenen Stellen in den Code integriert werden. Um einen vollständigen Datensatz zu senden, besteht die Möglichkeit, komplette Featuresets zu verwenden und das Senden vor der Schleife über einzelne Features zu implementieren. Dementsprechend könnten innerhalb dieser Schleife einzelne Features an den Server gesendet werden.

5.2.1.2 Implementierungsansätze

Die zusätzlichen Klassen, benötigt für das Senden von Daten an den WPS-Server, müssten so gestaltet sein, dass zunächst aus den Eingangsdaten sowie den voreingestellten Generalisierungsinformationen eine entsprechende XML-Datei generiert wird. Ein Beispiel dafür kann in Anhang B eingesehen werden. Die Schwierigkeit besteht darin, das interne Datenformat von Mapnik ins, für den Server notwendige, GML-Datenformat zu transformieren. Anschließend kann diese Datei über die Programmibliothek „*libcurl*“ an den Server gesendet werden. Beim Herunterladen des Mapnik-Quellcodes werden bereits Klassen zur Serveransprache mitgeliefert („*basiccurl.cpp*“ & „*basiccurl.h*“). Allerdings müssten diese modifiziert werden, da sie nicht die nötigen Einstellungen liefern, die für einen Aufruf des WebGen-WPS notwendig sind. Aus diesem Grund wäre anzuraten, eine neue Klasse zu konzipieren, welche im WebGen-WPS-Plug-in integriert ist. Dass eine solche generische Ansprache des WebGen-WPS möglich ist, wurde ausführlich getestet. Zur praktischen Umsetzung wurde der Server über Kommandozeile („*cURL*“), Python („*PycURL*“) sowie in einem C++- Programm („*libcurl*“) aufgerufen. Da ein entsprechendes Plug-in in C++ realisiert werden muss, wird der Programmcode eines korrespondierenden Serveraufrufs in Anhang C abgebildet.

Hierbei trat ein Problem auf, welches Auswirkungen auf die gesamte Implementierung hat, wie bereits in Kapitel 4.1.1 kurz angesprochen. Bei manueller Ansprache des WebGen-WPS wurde häufig für den Service („BuildingSimplification“ - Tabelle 1) eine Fehlermeldung (Abbildung 5.3) zurückgeliefert, die nicht nachvollziehbar ist und dementspre-

chend sehr schwierig in einem automatisierten Vorgang interpretiert werden kann. Gerade beim Senden von großen Datenmengen (Featuresets) kann diese auftreten und hat zum Inhalt, dass einige Geometrien nicht die notwendige Anzahl von Punkten aufweisen, die für die Berechnung notwendig sind. Durch Prüfung der Daten ließ sich jedoch feststellen, dass derartige Geometrien nicht im Datensatz vorhanden sind. Darüber hinaus kann die Fehlermeldung durch Angabe einer alternativen Mindestlänge teilweise umgangen werden. Im Kontext der Fehlermeldung dürfte das jedoch keinen Effekt haben. Logische Schlussfolgerung ist, dass es sich um einen Fehler des Servers, beziehungsweise des aufgerufenen Dienstes, handeln muss.



Aus diesem Grund wurde lediglich das Senden von singulären Features implementiert. Dadurch werden Geometrien, die nicht verarbeitet werden können, zunächst ungeneralisiert verwendet. Eine Umsetzung, die nicht zufriedenstellend jedoch derzeit nicht anders realisierbar ist. Demzufolge ist der WebGen-WPS, zumindest bis zur Behebung des Fehlers, nicht vollständig tauglich für eine Einbindung in die vollautomatisierte Verarbeitung von OSM-Daten.

Nachdem die Verarbeitung im Server gestartet wurde, liefert dieser eine XML-basierte Antwort, in welchem der Speicherort des ebenfalls XML-basierten Statusdokumentes zurückgeliefert wird. So ist es ständig möglich, den aktuellen Status der laufenden Berechnung abzufragen. Dieser kann auf „*ProcessAccepted*“, „*ProcessStarted*“, „*ProcessPaused*“, „*ProcessSucceeded*“ sowie „*ProcessFailed*“ stehen (OGC 2007 - Table 55). Demnach ergibt sich die Notwendigkeit eine Prozedur zu entwickeln, welche die Serverantwort interpretiert, das Statusdokument einliest und es permanent auswertet. Wird dabei „*ProcessFailed*“ gelesen, muss die gesamte Prozedur abgebrochen oder eine adäquate Fehlerauswertung implementiert werden. Für „*ProcessSucceeded*“ ist die Berechnung am Server abgeschlossen und die modifizierten Daten sind verfügbar. Um das wiederum XML-basierten Ergebnisdokument abzurufen, wird im Statusdokument die entsprechende URL angegeben. Es liegt auf der Hand, dass die Implementierung einer Klasse notwendig

ist, welche diese Daten interpretiert und zurück in den automatischen Renderingprozess von Mapnik speichert. Hierbei ist erneut eine Umwandlung vom GML-Format in das interne Datenformat von Mapnik erforderlich.

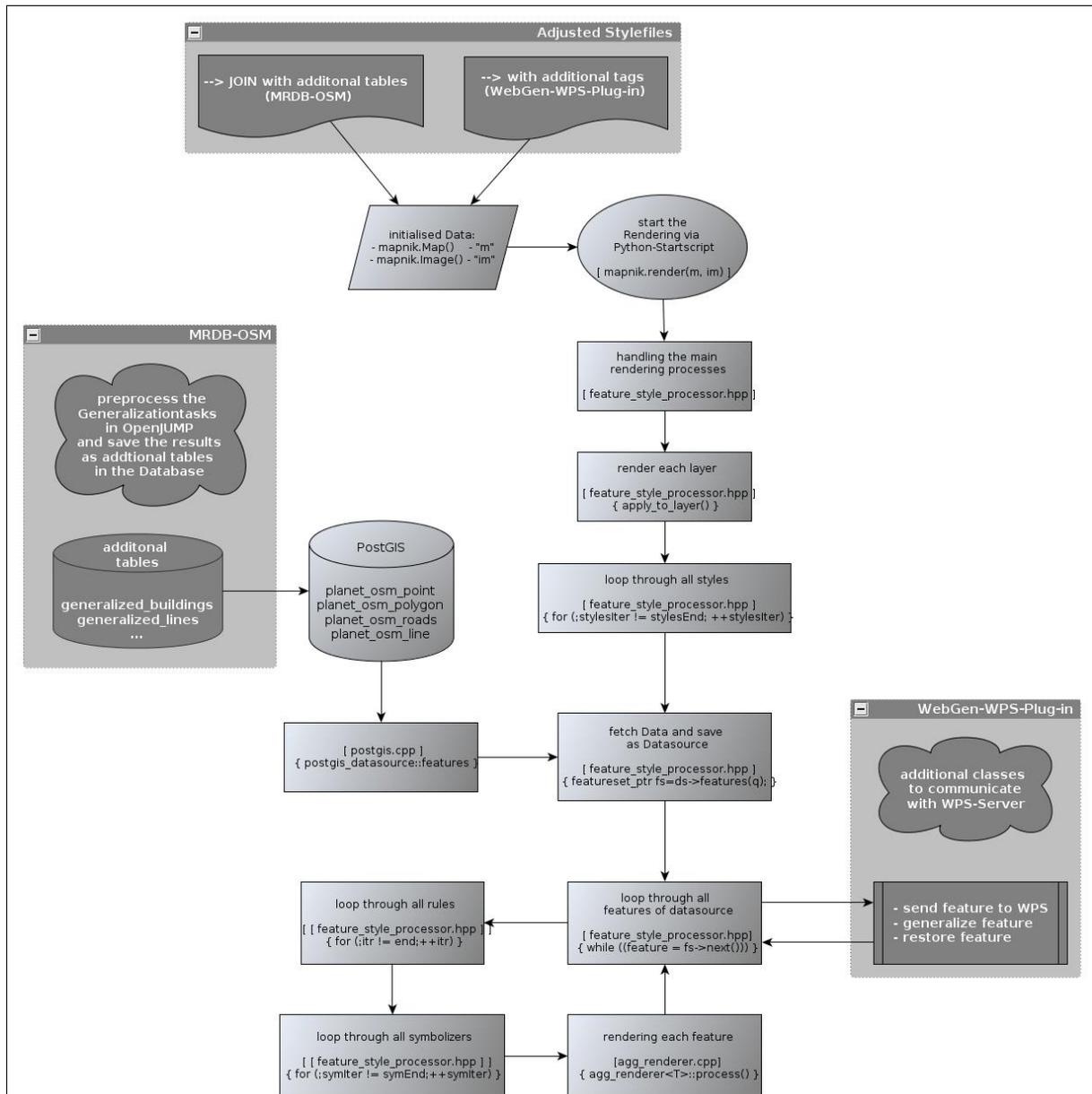


Abbildung 5.4: Schematische Darstellung der wichtigsten Klassen des Mapnik-Quellcodes

Alle Klassen, welche für die Berechnung von Karten notwendig sind – zusätzlich werden beide Implementierungen (dunkelgrau) veranschaulicht [WebGen-WPS-Plug-in = Kapitel 4.2.1. & MRDB-OSM = Kapitel 4.2.2.]

Es wurde demnach gezeigt, dass der Aufruf des WebGen-WPS in die vollautomatische Verarbeitung von OSM-Daten mit Mapnik bereits einbindbar ist. Allerdings sind noch verschiedene Probleme zu überwinden. So muss zunächst ein Plug-in für Mapnik entwickelt werden, welches die aufgezeigten Möglichkeiten umsetzt. Das größte Problem dürfte die

Verarbeitung von Fehlermeldungen darstellen, die am Server auftreten können. Dies steht im direkten Zusammenhang zu den Interaktionsmöglichkeiten bei Verkettung von Diensten (Alameh 2003, Foerster 2010). Einmal gestartet, wird das Rendering bei Mapnik vollständig opak verarbeitet, sodass keinerlei Zugriff auf die Verarbeitung möglich ist. Bei Einbindung von Ansprachen des WPS-Servers müssten demzufolge die Interaktionsmöglichkeiten erweitert werden, damit Nutzer während der Verarbeitung auf mögliche Probleme reagieren können. Vor allem die zugrunde liegenden Algorithmen der Generalisierungsservices müssen wesentlich flexibler generiert werden, sodass lediglich Fehler auftreten, die nachvollziehbar und innerhalb der Daten korrigierbar sind. Der aktuelle Ansatz, problematische Daten zunächst ungeneralisiert weiter zu verarbeiten, stellt dementsprechend nur eine vorübergehende Lösung dar.

5.2.2 Praktische Umsetzung einer „MRDB-OSM“

Im Gegensatz zur Implementierung in die vollautomatische Verarbeitung von OSM-Daten mittels Mapnik, ist dieser Ansatz ohne erheblichen Programmieraufwand direkt umsetzbar. Wie bereits beschrieben, sollen Daten hierbei zunächst im freien GIS OpenJUMP vorprozessiert und anschließend in unterschiedlichen Repräsentationsformen in der PostGIS-Datenbank abgespeichert werden. Daraufhin können sie, während des automatischen Rendering, abgefragt werden. In Abbildung 5.4 wird eine solche Implementierung schematisch dargestellt. Für die Versuche wurde mit OpenJUMP-Version 1.4.0.1 und PostGIS-Version 1.3.1a gearbeitet, wobei die Datenbank auf PostgreSQL-Version 8.4 mit PostGIS-Version 1.5.2-2 eingerichtet ist. Im Folgenden wird die praktische Implementierung einer „MRDB-OSM“ exemplarisch anhand einer Polygonvereinfachung, in Form von Gebäude-daten, veranschaulicht, woraufhin eine Analyse der daraus resultierenden Ergebnisse folgt. Abschließend werden die Ergebnisse der Glättung linearer Geometrien analysiert, welche zusätzlich über diesen Ansatz durchgeführt wurde.

5.2.2.1 Verfahrensablauf

Im ersten Schritt müssen relevante Daten aus der Datenbank in OpenJUMP geladen werden, was problemlos möglich ist, da hierfür lediglich eine „Datenbankebene“ angelegt wird. Während der Abfrage der Datenbank besteht die Option zusätzliche Bedingungen, entsprechend der SQL-Bedingung „*WHERE*,“ anzugeben. Für das Beispiel - Gebäudevereinfachung - muss

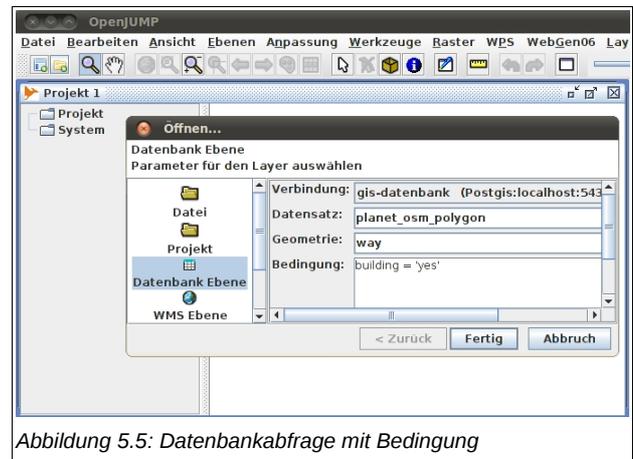


Abbildung 5.5: Datenbankabfrage mit Bedingung

eine Abfrage der Polygontabelle erfolgen. Innerhalb dieser Tabelle werden Gebäude durch die Spalte „building“ repräsentiert. Wurde dieses Attribut auf den Wert 'yes' gesetzt, repräsentiert das betreffende Polygon ein Gebäude. Zu beachten ist hierbei, dass nahezu alle Attributspalten im Textformat angelegt wurden, nicht als boolesche Variable. Das bedeutet, die Abfragebedingung muss ebenfalls textbasiert sein (Abbildung 5.5). Im Ergebnis wird eine neue Ebene angelegt, welche lediglich Features enthält, die Gebäude repräsentieren.

In Folge können Features zur Verarbeitung an den Server gesendet werden. Dazu muss das zuvor installierte WPS-Plug-in für OpenJUMP geöffnet und die korrespondierende URL der „*GetCapabilities*“-Abfrage (vgl. Abbildung 2.6) in die vorgesehene Textbox („*Server*“) eingegeben werden. Dadurch wird der Befehl an den Server gesendet, woraufhin eine automatische Auswertung der erhaltenen Antwort sowie eine Auflistung aller verfügbaren Generalisierungsservices (vgl. Tabelle 1) folgt. Im Beispiel wurde der Dienst „*BuildingSimplification*“ gewählt, woraus die Anzeige aller notwendigen Variablen in der rechten Spalte resultiert (vgl. Abbildung 2.7). Zunächst muss die geometrische Variable spezifiziert und die Übrigen entfernt werden, da beim Laden der OSM-Daten alle Spalten übertragen wurden. Wie in Abbildung 5.6 zu erkennen, ist das Entfernen sämtlicher semantischer Attribute notwendig, sodass

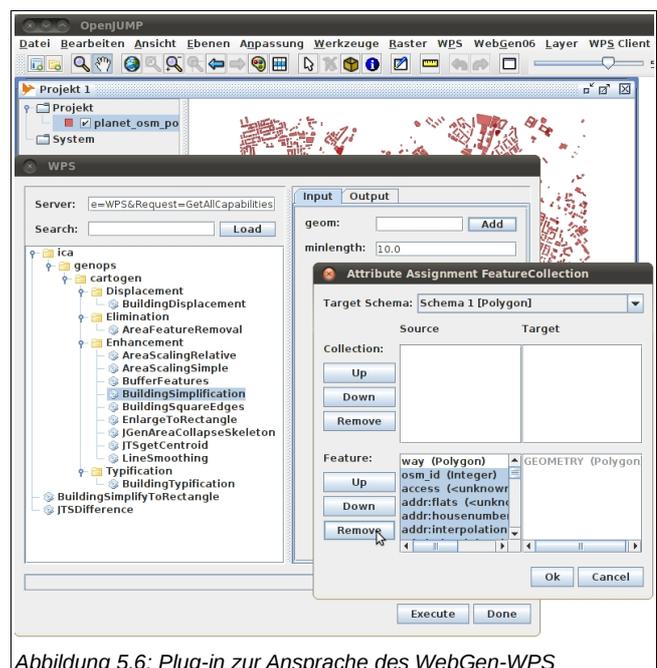


Abbildung 5.6: Plug-in zur Ansprache des WebGen-WPS

allein das geometrische Attribut „way“ verbleibt. Das ist zwingend erforderlich, da das Plug-in (bzw. WebGen-WPS) lediglich Geometrien verarbeitet. Der Nachteil dieser Verarbeitung wird im weiteren Verlauf deutlich. Daraufhin erfolgt die Spezifikation zusätzlich notwendiger Variablen. Entsprechende Angaben sind stets abhängig vom gewählten Dienst. Im Beispiel ist es lediglich die Mindestlänge der zu verarbeitenden Geometrie, wobei der Standardwert beibehalten wurde. Als problematisch ist die fehlende Angabe der Einheiten zusätzlicher Variablen anzusehen. Es ist anzunehmen, dass es sich dabei um die Einheit „Meter“ handelt. Eine Bestätigung gibt es dafür jedoch nicht. Nach Einstellung aller notwendigen Angaben wird der Server über den „Execute“-Knopf aufgerufen und die Daten verarbeitet. An dieser Stelle kann es, wie im vorangegangenen Kapitel bereits beschrieben, zu Fehlermeldungen des Servers kommen. Ein adäquater Umgang ist auch für diese Implementierung sehr schwierig, da die Ursache nicht nachvollziehbar ist. Dementsprechend ist auch diese Implementierung derzeit nur eingeschränkt nutzbar.

Wird die Berechnung am Server erfolgreich abgeschlossen, können resultierende Daten als neue Ebene abgelegt werden. An diesem Punkt des Verfahrensablaufs tritt ein weiteres Problem bei der Verarbeitung von Daten über WebGen-WPS auf. Es wurde bereits darauf hingewiesen, dass alle zusätzlichen semantischen Attribute verworfen werden müssen, da ausschließlich Geometriedaten verarbeitet werden können. Dementsprechend sind in der Ebene der generalisierten Daten lediglich die Attribute Geometrie und eine automatisch generierte GIS-interne Identifikationsnummer („*FID*“) enthalten. Um Redundanzen innerhalb der „*MRDB-OSM*“ zu vermeiden, werden in den zusätzlichen Tabellen allein die Geometrien abgespeichert und Features über eine Art Primärschlüssel untereinander referenziert. Aus diesem Grund wird die Identifikationsnummer korrespondierender Objekte („*osm_id*“) benötigt. Jedoch wurde dieses Attribut für den Serveraufruf verworfen. Der Umstand muss nun aufwendig umgangen werden, indem die Attribute der Ausgangsdaten auf die generalisierten Daten übertragen werden. Es müsste eine Modifikation des WebGen-WPS erfolgen, damit die Identifikationsnummer („*osm_id*“) während des Verarbeitungsprozesses direkt übernommen wird. Nur so ist eine optimale Implementierung realisierbar.

Derzeit ist dies nicht möglich, sodass nun die Attribute der ursprünglichen Ebene übertragen werden müssen. Durch Nutzung des Werkzeugs „Übertrag von Attributen...“ ist das in OpenJUMP realisierbar (Abbildung 5.7). Die besten Ergebnisse konnten erzielt werden, indem die Combobox „Beziehung“ auf „*innerhalb einer Distanz*“ (Parameter = 5.0) eingestellt wurde. Es kommt zwar zur Erzeugung zahlreicher doppelter Geometrien, sie lassen

sich jedoch anschließend mit Hilfe des Werkzeugs „Doppelte Geometrien löschen...“ entfernen. Die Ergebnisse der Verarbeitungen werden wiederum in zusätzlichen Ebenen abgelegt. Da lediglich die Spalte „osm_id“ benötigt wird, müssen anschließend die überflüssigen Attributspalten aus dem Datenblatt entfernt werden. Zusätzlich ist die Umbenennung der benötigten Spalte notwendig, da sie bei der Attributübertragung als „A_osm_id“ bezeichnet wurde. Dieser Schritt ist über „Schema anzeigen / bearbeiten“ durchführbar.

Solange die Identifikationsnummer bei der Verarbeitung am WebGen-WPS verworfen wird, ist diese Verarbeitung leider die einzige Möglichkeit. Allerdings dürfte eine solche Übertragung sehr leicht zu implementieren sein, weshalb es nur ein aktuelles Problem darstellt.

Zusätzlich muss die Identifikationsnummer (SRID) des zugrunde liegenden Referenzsystems angegeben werden, da sie beim Import in OpenJUMP nicht übertragen wurde. Regulär werden OpenStreetMap-Daten in Mercator-Projektion (SRID = 900913) verarbeitet, was bei den Versuchen beibehalten wurde.

Die generalisierten Daten können letztendlich in der PostGIS-Datenbank abgespeichert werden (Abbildung 5.8). Um eine „MRDB-OSM“ zu realisieren ist das Anlegen einer neuen Tabelle notwendig. Hierbei ist es ausgesprochen wichtig, dass zum Einen die SRID korrekt eingestellt wurde und zum Anderen Administrationsrechte an der Datenbank vorhanden sind. Anderenfalls können bei der Speicherung Fehler auftreten. Im Ergebnis wurde eine Datenbank multipler Repräsentationsformen für Gebäudedaten erstellt.

Um generalisierte Daten zu nutzen, ist es erforderlich das Stylefile für die Abfrage der „MRDB-OSM“ anzupassen. Durch Modifizierung der Datenbankabfrage innerhalb des Kennzeichens „Layer“ wird dies umgesetzt. Generalisierte Daten können über die

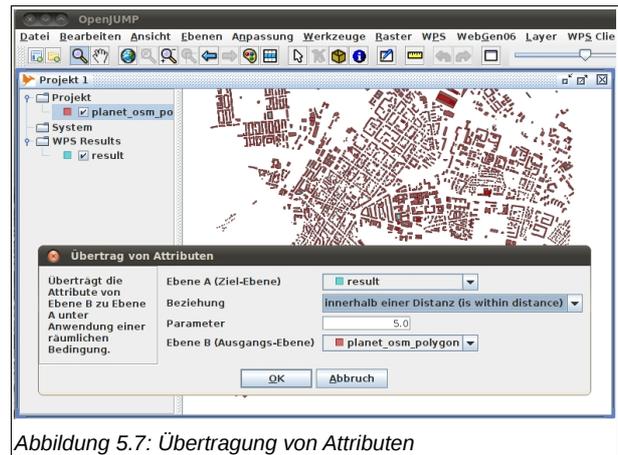


Abbildung 5.7: Übertragung von Attributen

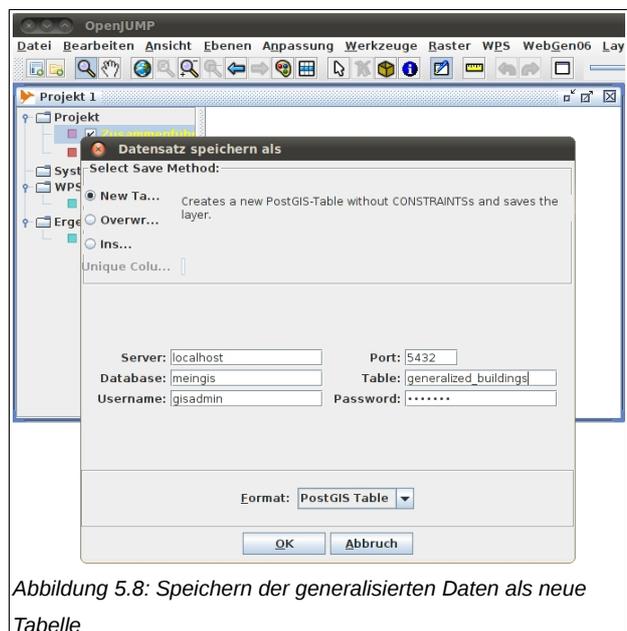


Abbildung 5.8: Speichern der generalisierten Daten als neue Tabelle

Identifikationsnummer („*osm_id*“) mit den Ausgangsdaten referenziert werden. Dies ist notwendig, da die korrespondierenden Attribute lediglich in der Ausgangsdatentabelle enthalten sind. Referenzierung wird in SQL über „*JOIN*“ realisiert. In Abbildung 5.9 wird die Umsetzung des Beispiels generalisierter Gebäudedaten dargestellt.

```
SELECT b.geometry as way
FROM planet_osm_polygon as a INNER JOIN generalized_buildings as b
ON a.osm_id = b.osm_id
WHERE (a.building = 'yes')
```

Abbildung 5.9: Beispiel einer modifizierten Datenbankabfrage für „MRDB-OSM“
(simple Umsetzung ohne semantische Attribute)

Für die Versuche wurde lediglich eine Einzelkarte des Testgebietes berechnet. Da hierfür keine maßstabsbedingten Unterscheidungen erforderlich sind, ist die Möglichkeit der mehrfachen Repräsentation nicht direkt notwendig und es wurden ausschließlich generalisierte Daten für die Berechnung verwendet. Beim Rendering von „*Slippy-Maps*“ müsste in größeren Maßstabsbereichen eine Standardabfrage der Gebäudedaten, unter Nutzung ungeneralisierter Daten, zur Anwendung kommen. Für kleine Maßstabsbereiche könnten, durch Angabe eines zusätzlichen „*Style*“ und eines weiteren „*Layer*“, generalisierte Daten über einen „*JOIN*“ abgerufen werden.

5.2.2.2 Polygonvereinfachung

Zur Analyse wurden verschiedene Kartenausschnitte mittels Mapnik gerendert. Zur Verwendung in „*Slippy-Maps*“ werden „*Tiles*“ regulär mit einer Auflösung von 256 x 256 Pixeln erstellt (www.openstreetmap.org). Für eine Ausgabe auf Bildschirmen ist dies ausreichend. Um einen Druck der Karten zu ermöglichen, wurden Bilddateien (*.png) der Größe 500 x 500 Pixeln (px) berechnet, welche in der Größe von 14 x 14 cm ausgegeben werden. Innerhalb des Pythonskripts ist die Ausgabe des Bildschirmmaßstabs (M_B) der entsprechenden Berechnung möglich. Dieser ist abhängig von der Projektionsart, hier Mercator-Projektion, sowie der Größe des Bildausschnitts. M_B gibt das Verhältnis zwischen einem Bildpunkt und der dadurch repräsentierten Fläche in der Realität an (<http://trac.mapnik.org/wiki/ScaleAndPpi>). Es muss allerdings zwischen Bildschirmmaßstab (M_B) und Ausgabemaßstab (M_A) unterschieden werden. Für einen Kartenausschnitt mit den geographischen Koordinaten $B_{box} = (13.70772, 51.0189, 13.73914, 51.03825)$ und den oben angegebenen Werten einer Bilddatei, wird von Mapnik der Bildschirmmaßstab $M_B = 7,00$ ausgegeben.

Demnach ergibt sich ein Bildausschnitt mit den realen Ausmaßen von rund 12 km² (3,498km x 3,498km). Für eine Ausgabegröße von 14 cm ergibt sich somit der Ausgabe- maßstab $M_A \approx 1:25.000$.

In den Ausführungen des SGK (2002) zur Formgeneralisierung sollten ab Maßstab 1:25.000 folgende Kriterien für Gebäudedarstellungen erfüllt werden:

- Grundsätzlich maßstäbliche Darstellung der Gebäude ab Mindestgröße 0,35 mm
- Unbedeutende Kleinstformen (kleine Anbauten und Einsprünge) sind eliminiert
- Charakteristische Grundformen sind betont
- Formmerkmale der Siedlungsstruktur sind erhalten

Tabelle 4: Merkmale der Formgeneralisierung von Gebäuden
im Maßstab 1:25.000 (SGK 2002 – S.71)

Die aufgeführten Kriterien werden durch den WebGen-WPS-Dienst „BuildingSimplification“ annähernd umgesetzt. Aufgrund dessen wurden verschiedene Karten mit den beschriebenen Werten erstellt (Anhang D). Um dem geforderten Maßstab zu entsprechen, kam lediglich ein Ausschnitt des gewählten Testgebietes zur Anwendung. Diese Karten wurden mit den vereinfachten Gebäudegeometrien und zum Vergleich mit den originalen Geometrien angefertigt. Zur optimalen Veranschaulichung von Unterschieden wurde zusätzlich ein Differenzbild der beiden Karten sowie eine kombinierte Darstellung aus Differenzbild und Karte mit generalisierten Geometrien erstellt. Diese Visualisierungen werden in Anhang D im Ausgabemaßstab $M_A \approx 1:25.000$ abgebildet, während in Abbildung 5.10 die kombinierte Darstellung zu sehen ist. Auf einen Blick ist erkennbar, wo Kleinstformen von Gebäuden entfernt wurden („Erosion“) und wo Lücken aufgefüllt beziehungsweise Formen hinzugefügt wurden („Delatation“). Die Begriffe „Erosion“ und „Delatation“ werden allerdings nur als Schlagworte verwendet, um eine simple Legendendarstellung zu gewährleisten. Sie sind nicht mit Dilatations- und Erosionsoperatoren, beispielsweise aus der digitalen Bildverarbeitung, gleichzusetzen.

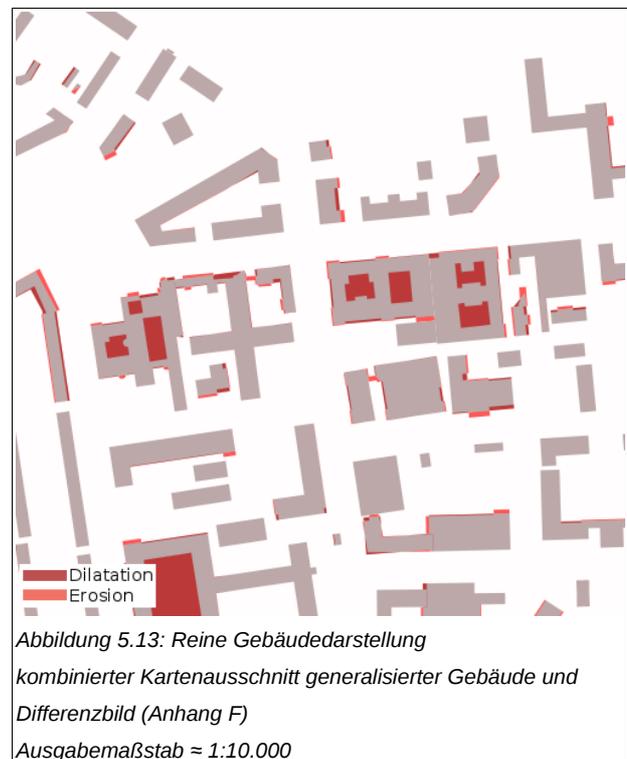


In dieser Abbildung sind bereits deutlich mehr Veränderungen zu erkennen. Bei nahezu allen Gebäuden, die einen Innenhof besitzen, wurden Lücken aufgefüllt. Dies ist akzeptabel, da im entsprechenden Maßstabsbereich vor allem der Charakter von Objekten gewahrt bleiben sollte, was in den meisten Fällen zutrifft. Lediglich im unteren Bereich der Karte fällt ein Gebäude auf, welches sehr starke Abweichungen von der charakteristischen Form aufweist.

Einer kritischen Betrachtung müssen allerdings Gebäude unterzogen werden, die von einer rechteckigen Form in eine dreieckige Form vereinfacht wurden. Das ist nicht akzeptabel. Es bestünde die Möglichkeit, solche Fehler bereits bei der Verarbeitung in OpenJUMP zu identifizieren. Allerdings ließen sich die fehlerhaften Veränderungen nicht simpel umgehen, da eine Einflussnahme auf die Verarbeitung am Server unmöglich ist. Wobei

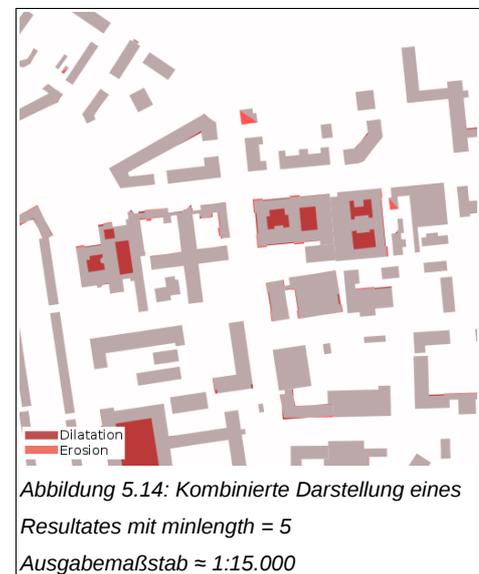
sich die Frage ergibt, wie die Interaktionsmöglichkeiten bei einer Verkettung von Diensten zu gestalten sind (vgl. Kapitel 2.2.3). Überdies ist eine nachträgliche, optische Analyse für den verwendeten Datensatz problematisch, da rund 3500 Geometrien verarbeitet werden. Dementsprechend ist es für größere Datensätze nicht erdenklich. Die Verbesserung des zugrunde liegenden Algorithmus stellt eine triviale Lösung dar, um solche Fehler zu vermeiden.

Innerhalb des vergrößert dargestellten Bereiches ist bereits ansatzweise zu erkennen, dass zahlreiche Kleinstformen an den Gebäuden generalisiert wurden. Um eine bessere Veranschaulichung zu gewährleisten, wurden zusätzlich Karten mit den Rahmenkoordinaten $B_{box} = (13.72088, 51.02682, 13.72873, 51.03166)$ erstellt. Daraus resultiert der Bildschirmmaßstab $M_B = 1,75$. Die Bildgröße bleibt konstant (500 x 500 Pixel) und die Ausgabegröße wurde auf 8 cm verringert. Damit ergibt sich der Ausgabemaßstab $M_A \approx 1:10.000$. Diese Kartenausschnitte wurden ebenfalls mit originalen und generalisierten Gebäudegeometrien sowie dem daraus resultierenden Differenzbild und einer kombinierten Darstellung erstellt (Anhang F). In Abbildung 5.12 wird abermals die Kombination aus einer Karte mit generalisierten Gebäuden und dem Differenzbild aufgezeigt. Zusätzlich wurden diese Kartenausschnitte ebenfalls ausschließlich mit Gebäudegeometrien errechnet (Anhang F), deren kombinierte Darstellung in Abbildung 5.13 zu sehen ist.



Es ist deutlich zu erkennen, dass unnötige Kleinstformen generalisiert wurden, wie von SGK (2002) gefordert (vgl. Tabelle 4). In diesem Kontext kann das generalisierte Resultat als maßstabsgerechte Generalisierung der Gebäude angesehen werden. Unter Ausschluss des Entstehens dreieckiger Geometrien, was als gravierender Fehler betrachtet werden muss, lassen sich durch die Verarbeitung über WebGen-WPS sehr gute Ergebnisse erzeugen. Korrekt verarbeitete Geometrien wurden nahezu optimal vereinfacht und der Objektcharakter bleibt zumeist sehr gut gewahrt. Der zugrunde liegende Algorithmus bedarf jedoch noch einer wesentlich flexibleren Implementierung, damit keine unplausiblen Fehler oder fehlerhafte Geometrien auftreten. Erst dann ist eine Vereinfachung von OSM-Gebäudegeometrien durch die Implementierung einer „MRDB-OSM“ uneingeschränkt empfehlenswert.

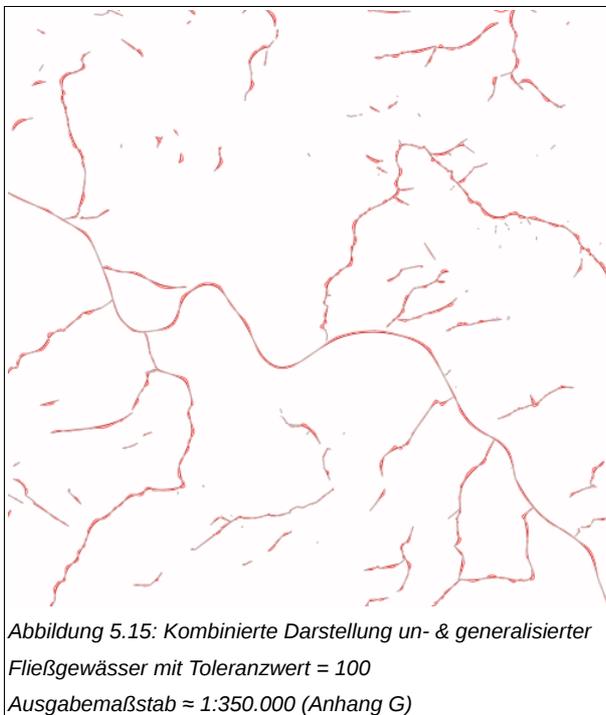
An dieser Stelle muss noch einmal angemerkt werden, dass bei der Erstellung der vorgestellten Ergebnisse immer eine Mindestlänge von `minlength = 10` eingestellt wurde. Durch zahlreiche Versuche wurde festgestellt, dass sich damit die besten Ergebnisse erzielen lassen, gerade in Bezug auf die Eliminierung von Kleinstformen. Das ist in Abbildung 5.14, einer kombinierten Darstellung mit `minlength = 5`, sehr gut zu erkennen. Einerseits werden viel zu wenige Kleinstformen der Gebäude entfernt, andererseits entsteht eine höhere Anzahl an fehlerhaften Geometrien, wie am Gebäude im oberen Bereich zu erkennen ist.



Abschließend bedarf ein Problem einer kurzen Erwähnung, das bei der Nachbearbeitung der Identifikationsnummer auftrat. Es ist nie gelungen, exakt die gleiche Anzahl an Geometrien in die „MRDB-OSM“ zu schreiben wie zuvor ausgelesen wurden. Es war im Mittelwert stets eine zusätzliche Geometrie vorhanden. Sie wurden bei der „Entfernung doppelter Geometrien“ nicht korrekt identifiziert. Ein Auffinden ist nahezu unmöglich, weshalb der Fakt bei den Versuchen zunächst übergangen wurde. Selbstverständlich ist dadurch eine vollständige Implementierung derzeit nicht möglich. Diesbezüglich wird die Relevanz der Implementierung einer Übernahme zusätzlicher Attribute noch einmal sehr deutlich. Durch automatische Übernahme der Identifikationsnummer, innerhalb des WPS-Servers, würde die gesamte Nachbearbeitung wegfallen und der Fehler natürlich nicht mehr auftreten.

5.2.2.3 Linienvereinfachung

Des Weiteren wurde der Generalisierungsdienst „LineSmoothing“ getestet. Zur Veranschaulichung wurden innerhalb eines wesentlich größeren Testgebietes mit den Rahmenkoordinaten $B_{box} = (13.6291, 50.9965, 13.8804, 51.1436)$ lediglich lineare Geometrien, die ein Flussnetz darstellen, verarbeitet und im Ausgabemaßstab $M_A \approx 1:350.000$ ($M_B = 55,95$ Druckgröße 8 cm) abgebildet. Der Verfahrensablauf zum Anlegen der „MRDB-OSM“ entspricht exakt dem Beispiel der Gebäudegeneralisierung. Jedoch wurden hierbei verschiedene Generalisierungsstufen, also multiple Repräsentationsformen der Geometrien angelegt.



Der optimale Toleranzwert musste empirisch bestimmt werden, wobei für diesen Maßstabsbereich sehr gute Ergebnisse mit einem Wert von 100 erzielt werden konnten. Dadurch ist ein guter Kompromiss zwischen Vereinfachung und topologischen Veränderungen umsetzbar (Abbildung 5.15). Ab einem Toleranzwert von über 200 treten starke Lageverschiebungen auf, die nicht übernommen werden dürfen. In Abbildung 5.16 ist anhand des „Elbbogens“ erkennbar, dass die Elbe bei einem Wert von 500 topologisch stark verändert wurde und komplett das Flussbett verlässt. Zu beachten ist, dass die Elbe bedingt durch deren Breite, doppelt (linear und flächenhaft) abgebildet wird. Die vollständigen Resultate für die Toleranzwerte 50, 100, 200 und 500 werden in Anhang G verdeutlicht. Nachteilig ist wiederum, dass keinerlei Maßeinheit des Toleranzwertes angegeben wird.

Es ist deutlich zu erkennen, dass es sich um einen komplexen Algorithmus zur Linienglättung, im Gegensatz zu „einfachen“ Algorithmen zur Linienvereinfachung (vgl. Douglas-Peucker 1973), handelt. Zwar kann es zu topologischen Lageverschiebungen kommen, jedoch bleiben grundlegende Charakteristiken stets erhalten. Dementsprechend ist dieser Algorithmus nahezu ohne Einschränkungen verwendbar. Auch die Verarbeitung über WebGen-WPS sowie die Übertragung von Attributen ist bei den Versuchen stets problemlos verlaufen. Das bedeutet, es konnte immer die gleiche Anzahl an Geometrien in die Datenbank zurück gespeichert werden, wie zuvor eingelesen wurde. Der einzige Nachteil ist der Zeitaufwand der Nachbearbeitung. Da OpenJUMP auf dem verwendeten System nicht vollständig flüssig arbeitet, hat die Verarbeitung eines Datensatzes durchschnittlich zehn Minuten in Anspruch genommen.

Demzufolge ließ sich zeigen, dass derzeit bereits ansprechende Ergebnisse durch die Implementierung einer „MRDB-OSM“ zu erzielen sind. Die verwendeten Generalisierungsoperatoren zeigen großes Potential für eine zukünftige Einbindung in den Renderingprozess von OSM-Daten. Gerade das Beispiel der Linienvereinfachungen beweist, wie vorteilhaft der Zugriff auf komplexe Algorithmen über den WebGen-WPS sein kann. So lassen sich wesentlich bessere Generalisierungsergebnisse als mit konventionellen – durch kommerzielle oder freie Software zur Verfügung gestellte - Algorithmen erzielen. Zum Vergleich wird in Kapitel 5.3 die Verarbeitung dieses Flussnetzes noch einmal mit einem klassischen Douglas-Peucker-Algorithmus veranschaulicht, wodurch grundlegend differierende Ergebnisse entstanden.

Es gilt allerdings noch einige Schwachstellen zu überwinden. So ist zum Einen der zugrunde liegende Algorithmus der Gebäudegeneralisierung noch nicht vollständig ausgereift, um diesen Service problemlos aufrufen zu können. Zweifelhafte Generalisierungsergebnisse können auftreten, die in der Form nicht in ein fertiges Kartenbild übertragbar sind. Es stellt sich die Frage, wie aktuell damit verfahren werden soll? Konsequenterweise müssten die Daten aufwendig im Nachhinein bearbeitet werden, was allerdings ebenfalls keine zufriedenstellende Lösung ergibt. Demnach ist die einzige Möglichkeit, die zugrunde liegenden Algorithmen der Generalisierungsservices so flexibel zu implementieren, dass fehlerhafte Generalisierungsergebnisse vermieden werden. Zum Anderen ist die Übertragung von zusätzlichen Attributen derzeit zu umständlich und zeitaufwendig. Dies sollte sich durch die Implementierung einer Attributübernahme innerhalb des WPS-Servers sehr leicht lösen lassen.

5.3 Implementierung von PostGIS-Funktionen

Wie in Kapitel 4.2 beschrieben, ist es ebenfalls möglich PostGIS-Funktionen in den automatischen Renderingprozess von OpenStreetMap einzubinden und zur Generalisierung der Daten zu verwenden. Es wurde darauf hingewiesen, dass einfache und komplexe Implementierungen möglich sind. In dieser Arbeit werden ausschließlich Beispiele für einfache und direkt implementierbare Lösungen getestet. Die Anwendung komplexer Umsetzungen würde den geforderten Umfang dieser Arbeit in zu hohem Maße übersteigen.

Eingeleitet wird dieses Kapitel mit der Untersuchung zur Implementierung eines zusätzlichen Auswahlkriteriums, woraufhin die Möglichkeit zur Betonung von Objekten vorgestellt wird. Im weiteren Verlauf werden Resultate der Vereinfachung eines Flussnetzes analysiert.

5.3.1 Auswahl

Zunächst wurde die Funktion „*ST_Length()*“ als Auswahlkriterium von Mindestlängen getestet. Dazu wurde diese als zusätzliches Abfragekriterium innerhalb der Datenbankabfrage des Stylefile integriert (Abbildung 5.17).

```
SELECT way, waterway
FROM planet_osm_line
WHERE waterway is not NULL AND ST_Length(way)
BETWEEN 251 AND 1000000
```

Abbildung 5.17: Beispiel für die Einbindung der PostGIS-Funktion "ST_Length()"

Somit ergibt sich die Möglichkeit, zusätzlich zur rein semantischen Auswahl linearer Geometrien, diese auch in Abhängigkeit einer zuvor eingestellten Mindestlänge auszuwählen. Zur Verdeutlichung wurde es auf das Flussnetz des gesamten Testgebietes angewendet. Dabei wurden Flüsse länger als 250, 500, 1000, 2000, 3000 und 5000 Metern dargestellt. Die resultierenden Ergebnisse werden in Anhang H ausschließlich als Flussdarstellungen visualisiert. Für eine Druckgröße von 8 cm und $M_B = 168,76$ ergibt sich ein Ausgabemaßstab von $M_A \approx 1:1\text{Mio}$.

5.3.2 Betonung

Über diese Funktion ist es ebenfalls möglich, eine Betonung linearer Geometrien zu realisieren. Dementsprechend wird die Funktion „*ST_Length()*“ nicht als Auswahlkriterium, sondern als Darstellungskriterium verwendet. Da eine Abfrage der Mindestlängen nicht innerhalb der „*Style*“-Spezifizierungen des Stylefiles möglich ist, muss für unterschiedliche Längen jeweils ein „*Style*“ sowie ein „*Layer*“ angelegt werden. Bei der Umsetzung wird eine Schwachstelle der Einbindung dieser Funktion in Bezug auf OSM-Daten deutlich. In Abbildung 5.18 werden Geometrien schwach (kürzer als 7000 Meter) und verstärkt (länger als 7000 Meter) dargestellt. Es ist direkt ersichtlich, dass die Auswahl nicht exakt funktioniert. Es werden einige Flussabschnitte dünn dargestellt, obwohl sie zu einem sehr langen Fluss gehören. Es lässt sich damit begründen, dass lineare Geometrien, in OpenStreetMap, als Segmente abgespeichert werden.

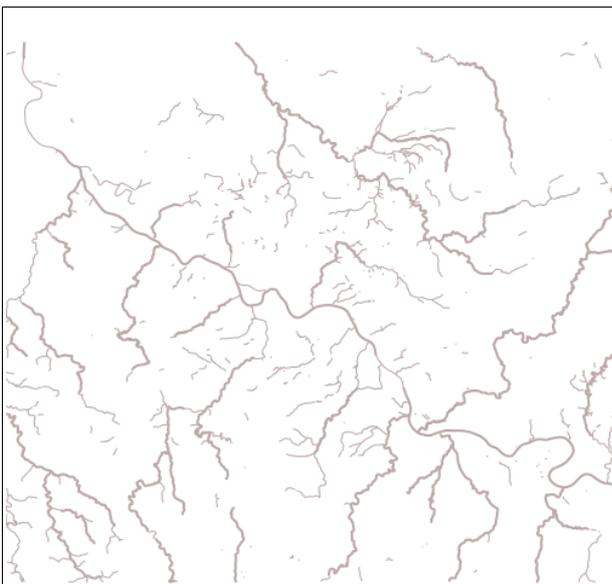


Abbildung 5.18: Beispiel für Betonung nach Mindestlängen anhand eines Flussnetzes - Ausgabemaßstab $\approx 1:1\text{Mio}$.

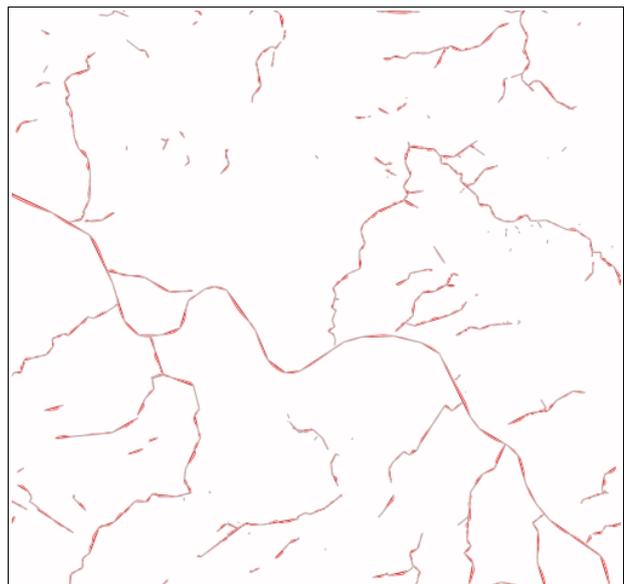


Abbildung 5.19: Generalisierte lineare Geometrien (PostGIS) Kombinierte Darstellung un- & generalisierter Fließgewässer mit Toleranzwert = 100 Ausgabemaßstab $\approx 1:350.000$ (Anhang I)

Resultierend ist eine simple Implementierung der PostGIS-Funktion „*ST_Length()*“ sowohl als Auswahlkriterium, aber auch als Darstellungskriterium nicht umsetzbar. Eine Umgehung des Fehlers wäre möglich, indem Segmente zusammenfasst und in einer Geometrie abgelegt werden, sodass Mindestlängen in Bezug auf die gesamte Entität abgefragt werden. Dementsprechend besteht im Bereich komplex implementierter Funktionen deutliches Potential. Da hier jedoch ausschließlich einfache Implementierungen untersucht werden, wird auf eine diesbezügliche praktische Umsetzung verzichtet.

5.3.3 Linienvereinfachung

Als weitere Möglichkeit wurde bereits beschrieben, dass mittels „*ST_Simplify()*“ die Vereinfachung linearer Geometrien in Form des Douglas-Peucker-Algorithmus umsetzbar ist. Um einen direkten Vergleich zur Linienvereinfachung mittels WebGen-WPS zu schaffen, wurde das gleiche Testgebiet wie in Kapitel 5.2.2 (Linienvereinfachung) gewählt. Es wurden Darstellungen mit den Toleranzabstufungen 50, 100, 200, 500 erstellt, welche in Anhang I dargestellt sind. Gute Ergebnisse lassen sich ebenfalls mit einem Toleranzwert von 100 erreichen (Abbildung 5.19). Es ergibt sich allerdings ein sehr kantiges Linienbild, das für diesen Generalisierungsalgorithmus typisch ist. Dies steht im extremen Gegensatz zu den Ergebnissen über den WPS-Server, wo runde Linien erzeugt wurden. Eine sehr leichte Umsetzbarkeit dieser Implementierung steht dem jedoch gegenüber. Somit kann durch simple Einbindung der Funktion innerhalb der Datenbankabfrage eine Vereinfachung von linearen Geometrien realisiert werden.

```
SELECT ST_Simplify(way,100) as way, waterway
FROM planet_osm_line
WHERE waterway is not NULL
```

Abbildung 5.20: Beispiel für eine Implementierung der PostGIS-Funktion „*ST_Simplify()*“

Bei Anwendung der Funktion „*ST_SimplifyPreserveTopology()*“ werden exakt die gleichen Resultate abgeleitet, sodass kein Effekt der topologischen Validierung zu erkennen ist. Aus diesem Grund wurde zusätzlich die Ableitung polygonaler Geometrien implementiert, da dies in Kapitel 4.2 bereits in Betracht gezogen wurde.

5.3.4 Polygonvereinfachung

In diesem Zusammenhang wurde zunächst überprüft, ob sich bei Ableitung vereinfachter Polygone über die Funktionen „*ST_SimplifyPreserveTopology()*“ und „*ST_Simplify()*“ Differenzen ergeben. Dazu erfolgte die Berechnung einer Darstellung reiner Gebäudegeometrien (vgl. Kapitel 5.2.2.2) innerhalb der Rahmenkoordinaten $B_{box} = (13.70772, 51.0189, 13.73914, 51.03825)$ unter Verwendung beider Funktionen. Im Differenzbild der beiden Resultate (Abbildung 5.21) ist deutlich zu erkennen, dass sich für einen Toleranzwert von 15 erhebliche Unterschiede ergeben.



Abbildung 5.21: Differenzbild der Funktionen „ST_Simplify()“ & „ST_SimplifyPreserveTopology()“
Toleranzwert 15 - Ausgabemaßstab $\approx 1:43.000$ (Anhang J)



Abbildung 5.22: Beispiel für Gebäudegeneralisierung über „ST_SimplifyPreserveTopology()“
Toleranzwert 10 - Ausgabemaßstab $\approx 1:10.000$

Rot dargestellte Geometrien gehen bei einer Vereinfachung über „ST_Simplify()“ vollständig verloren und fehlen in der resultierenden Darstellung. Demgegenüber bleiben, bei einer Vereinfachung über „ST_SimplifyPreserveTopology()“, alle Geometrien vollständig erhalten. Um den gravierenden Datenverlust aufzuzeigen, wurden die vollständigen Daten zunächst nur im Maßstab $M_A \approx 1:43.000$ dargestellt. Da sich auch Unterschiede in den Ergebnissen beider Funktionen ergeben, wurden diese zusätzlich innerhalb des Teilausschnittes $B_{\text{box}} = (13.72088, 51.02682, 13.72873, 51.03166)$ im Maßstab $M_A \approx 1:10.000$ ausgegeben. Dort fallen die fehlenden Geometrien, der Ableitung mittels „ST_Simplify()“, deutlich auf (Anhang J - Abbildung 8.47). In dieser Abbildung ist allerdings auch erkennbar, dass ein Objekt fehlerhaft verarbeitet wurde (oben rechts). Es kam zu erheblichen topologischen Verzerrungen. Wenn an solch ein Gebäude direkt eine Straße angrenzt, könnte es zu Überschneidungen kommen.

Daraus resultierend, muss zum Ergebnis gelangt werden, dass über „ST_SimplifyPreserveTopology()“ abgeleitete Geometrien topologisch konsistenter verarbeitet werden als mit „ST_Simplify()“. Demnach konnte die in Kapitel 4.2 aufgestellte Vermutung bestätigt werden. Folglich wird ausschließlich die Funktion „ST_SimplifyPreserveTopology()“ durch Untersuchung damit abgeleiteter Daten (Toleranzwert 10) analysiert. In Abbildung 5.22 ist ersichtlich, dass Kleinstformen nur spärlich entfernt wurden. Zusätzlich ergeben sich neue Formen, welche der Entfernung von Kleinstformen entgegenstehen. Darüber hinaus werden Lücken nicht geschlossen, was für den angestrebten Ausgabemaßstab

$M_A \approx 1:25.000$ (Abbildung 5.23) jedoch notwendig ist. Innerhalb dieses Maßstabsbereiches ist es, gerade bei Vereinfachung von Gebäudegeometrien, erforderlich, dass charakteristische Grundformen der Geometrien erhalten bleiben. Dies ist für das Resultat nicht der Fall. Im vergrößerten Bereich von Abbildung 5.23 ist das gut zu erkennen. Dort hat sich eine zylindrische Form ausgebildet, welche die Charakteristik der Geometrie grundlegend verändert. In Anhang K werden Grunddaten und abgeleitete Geometrien direkt nebeneinander gestellt. Obwohl sich wesentlich bessere Resultate, als über „*ST_Simplify()*“ erzielen lassen, ist von einer Anwendung dieser Funktion schlicht abzuraten. Es ist nicht möglich, qualitativ ansprechende Generalisierungsergebnisse zu erzielen, welche dem kartographischen Anspruch bezüglich der Vereinfachung von Gebäudegeometrien entsprechen. Vor allem mit Blick auf die Resultate, die über den WebGen-WPS erzielt werden konnten.



Abbildung 5.23: Kartenausschnitt generalisierter Gebäude in reiner Gebäudedarstellung
bearbeitet mit „*ST_SimplifyPreserveTopology*“ (Toleranzwert 10) inklusive Bereich mit doppelter Vergrößerung
Ausgabemaßstab $\approx 1:25.000$ - Rahmenkoordinaten = (13.70772, 51.0189, 13.73914, 51.03825) (Anhang K)

Im Ergebnis bleibt festzuhalten, dass eine Einbindung von PostGIS-Funktionen in den automatischen Prozess durchaus ohne Weiteres möglich ist. Allerdings lässt sich dies nur mit Einschränkungen empfehlen. Es ist vorteilhaft, dass sie zumindest für einfache Implementierungen zum derzeitigen Zeitpunkt direkt eingebunden werden können. Vor allem die Verwendung der Funktion „*ST_Simplify()*“ stellt eine Alternative zur Generalisierung über WebGen-WPS dar. Lineare Geometrien können sehr simpel vereinfacht werden. Nachteilig ist, dass dies lediglich über einen einfachen Douglas-Peucker-Algorithmus realisiert wird. Vereinfachung von Polygonen ist nicht zu empfehlen. Die Erweiterung von Auswahlkriterien kann nur eingeschränkt angewandt werden, da OSM-Daten dies nicht direkt zulassen. Jedoch kann daran das Potential von PostGIS-Funktionen erkannt werden, weshalb innerhalb dieses Bereiches weitere Untersuchungen angestrebt werden sollten.

6 Schlussfolgerungen und Ausblicke

Im letzten Kapitel werden Schlussfolgerungen gezogen, die sich aus den Resultaten der vorliegenden Arbeit ergeben. Zunächst erfolgt eine Diskussion der erhaltenen Ergebnisse in Relation zu den einführend aufgestellten Fragen. Innerhalb der Diskussion werden bestehende Probleme zusammengefasst und parallel dazu resultierender Forschungsbedarf aufgezeigt. Abschließend wird ein kurzes Fazit über die gesamte Arbeit gezogen.

6.1 Diskussion der Ergebnisse

Das Ziel dieser Arbeit ist die Untersuchung unterschiedlicher Generalisierungsfunktionalitäten auf deren Eignung zur Integration in die automatische Erstellung von Karten aus OpenStreetMap-Daten. Nach einführender Erläuterung der notwendigen Grundlagen wurde zunächst der aktuelle Grad kartographischer Generalisierung im Projekt OpenStreetMap in Bezug auf elementare Generalisierungsoperatoren untersucht. Anschließend erfolgten theoretische Überlegungen bezüglich Implementierung und Effektivitätssteigerung von Generalisierungsfunktionalitäten in OpenStreetMap. Woraufhin einige Ansätze praktisch implementiert und entstandene Resultate analysiert wurden. Im Folgenden werden die einleitend aufgestellten Fragen aufgegriffen und in Relation zu den Ergebnissen dieser Arbeit diskutiert.

Besteht die Notwendigkeit zur Einbindung kartographischer Generalisierung?

Anhand der Weltkartenansicht („*Slippy-Map*“) von Osmarender ist bereits ersichtlich, dass es großen Bedarf zur Untersuchung des Projektes OpenStreetMap in Bezug auf kartographische Generalisierung gibt, da Auswahl mangelhaft umgesetzt wird. Durch eingehende Betrachtung der mit Mapnik erstellten „*Slippy-Map*“ in Bezug auf elementare Generalisierungsvorgänge (nach Hake et al. 2002), wurde die Notwendigkeit zur Einbindung kartographischer Generalisierung klar verdeutlicht. So konnten ausschließlich elementare Vorgänge *Auswahl*, *Überhaltung* (Vergrößerung) und *Signaturierung* in den Kartendarstellungen identifiziert werden. Es kommt lediglich zur Realisierung trivialer Vorgänge, ohne die Kartendarstellungen nicht umsetzbar sind. Vorgänge, die aufwendige Verarbeitungen der Daten zur Folge hätten, werden jedoch nicht angewandt.

Selbst bereits verwendete Vorgänge weisen erhebliche Mängel auf. Beispielsweise wird semantische Auswahl nicht in Anlehnung an wissenschaftliche Erkenntnisse umgesetzt, wie anhand der Darstellung von Siedlungsflächen zu erkennen ist. Gebäudegrundrisse werden, wenn in der OSM-Datenbank vorhanden, selbst in viel zu kleinen Maßstabsbereichen noch abgebildet obwohl das menschliche Auge sie nicht mehr einzeln auflösen kann. Begründet liegt es in der Methode zur Erstellung der Style-Dateien, auf welchen die Visualisierung von OSM-Daten grundlegend basiert. Unabhängig von der gewählten Softwarelösung werden jene Dateien rein textbasiert und dementsprechend sehr abstrakt erstellt. Dieser Mangel ist nicht durch Integration vorhandener Funktionalitäten behebbbar, weshalb in dieser Arbeit keine Lösungsansätze aufgezeigt werden konnten.

Ist es möglich, Generalisierungsfunktionalitäten in den Berechnungsvorgang von OpenStreetMap-Karten zu integrieren?

Die Klärung der Frage erfolgte nicht innerhalb eines speziellen Kapitels. Es wurde im Grundlagenkapitel deutlich, dass OSM-Karten vollständig automatisiert, durch verschiedene Softwarelösungen erstellt werden. Das in C++ umgesetzte Mapnik hat in diesem Zusammenhang großes Potential zur Integration von Generalisierungsfunktionalitäten gezeigt. Indem zusätzliche Klassen in die vorhandene Programmbibliothek eingefügt werden, ergibt sich die Möglichkeit zum Aufruf des angestrebten Generalisierungsdienstes WebGen-WPS. Darüber hinaus verwendet Mapnik als Datenquelle die objekt-relationale Datenbank PostgreSQL inklusive PostGIS, der entsprechenden Erweiterung für räumliche Daten. Folglich wurde schnell deutlich, dass zusätzlich verschiedene Funktionalitäten über interne Funktionen der PostGIS-Erweiterung integriert werden können. Aus diesen Erkenntnissen erfolgte anschließend die Entwicklung von Lösungsansätze.

Die Softwarelösung Osmarender wurde allerdings direkt ausgeschlossen. Für Osmarender sind unterschiedliche XSLT-Prozessoren verwendbar, wodurch die Integration zusätzlicher Funktionalitäten wesentlich aufwendiger ist. Ungeachtet dessen muss hier ebenfalls eine bessere Lösung für die Erstellung der Style-Dateien gefunden werden.

Ebenso wurde Maperative grundlegend ausgeschlossen, da sich dies noch zu stark in der Entwicklungsphase befindet.

Welche Funktionalitäten sind geeignet und wie können diese implementiert werden?

Grundlegend wurden in dieser Arbeit zwei Funktionalitäten identifiziert, welche Potential zur Integration von Generalisierungsvorgängen aufweisen. Zum Einen der Aufruf des Generalisierungsdienstes WebGen-WPS und zum Anderen die Einbindung von Funktionen, welche durch PostGIS zur Verfügung stehen.

Über WebGen-WPS können verschiedene geometrische Modifikationen der OSM-Daten angewendet und generalisierte Geometrien abgeleitet werden. Darauf basierend, wurden zwei Implementierungsansätze durch theoretische Überlegungen entworfen. Die direkte Einbindung eines Aufrufs des WebGen-WPS in den automatischen Renderingvorgang von Mapnik sowie die Nutzung dieses Dienstes in Form einer speziell entworfenen „MRDB-OSM“ - Architektur.

Für den Ansatz zur direkten Einbindung eines Aufrufs des WebGen-WPS wurde bereits in den theoretischen Überlegungen sehr rasch festgestellt, dass eine vollständige praktische Umsetzung im Rahmen dieser Arbeit nicht durchgeführt werden kann. Dennoch wurde erörtert, wie die Implementierung erfolgen könnte und welche Faktoren in diesem Zusammenhang beachtet werden sollten, sodass weitere Forschungsarbeiten direkt an die Ergebnisse dieser Arbeit anknüpfen können. Dementsprechend wird es notwendig, ein Plug-in für Mapnik zu entwickeln, welches alle beschriebenen Operationen ausführt. Zusätzlich verdeutlichte sich, dass für ein solches Plug-in eine „komplexe Implementierung“ anzustreben ist. Im Vergleich zu einer statischen „simplen Implementierung“ ist in dieser Form eine bessere Anwendbarkeit für alle OSM-Nutzer erzielbar.

Im Gegensatz dazu erwies sich die Implementierung einer „MRDB-OSM“ als nahezu trivial, da alle notwendigen Bedingungen in der Verarbeitung von Mapnik bereits vorhanden sind. In dem Kontext wurde überdies das freie GIS OpenJUMP verwendet, um den Generalisierungsdienst anzusprechen. Sodass spezielle OSM-Daten aus der PostGIS-Datenbank ausgelesen, über WebGen-WPS generalisiert und anschließend als multiple Repräsentationsformen in zusätzlichen Tabellen abgespeichert werden. Aus diesem Konzept resultiert allerdings ein hohes Maß an manuellen Vorarbeiten, weshalb es vorwiegend zur Ableitung von Einzelkarten aus kleinen Datensätzen geeignet ist. Zur Erstellung von „Slippy-Maps“ müssten Datensätze der gesamten Erde manuell vorverarbeitet werden. Durch die stetige Zunahme der Datendichte des gesamten Projektes müssten sie

zusätzlich ständig manuell aktualisiert werden. In diesem Zusammenhang hält nur eine vollständig automatisierte Lösung einer Gegenüberstellung von Aufwand und Nutzen stand.

Als weiterer Implementierungsansatz kam die Option in Betracht, einige PostGIS-Funktionen einzubinden. In theoretischen Überlegungen wurde festgestellt, dass damit nicht nur geometrische Modifikationen, sondern auch sachliche Generalisierung mit geometrischem Effekt, wie *Auswahl* oder *Betonung*, umgesetzt werden könnten. In diesem Kontext sind ebenfalls unterschiedliche Möglichkeiten zur Einbindung aufgezeigt worden, obgleich lediglich einfache, bereits vorhandene Funktionen praktisch implementiert wurden. Das ist zur praktischen Demonstration vorhandener Möglichkeiten vollständig ausreichend. Dieser Lösungsansatz wird implementiert, indem ausgewählte Funktionen trivial in die SQL-Datenbankabfrage des „*Mapnik-Stylefile*“ eingebunden werden. Demzufolge sind über die Einbindung von PostGIS-Funktionen verschiedene Generalisierungsfunktionalitäten in das vollständig automatisierte Rendering integrierbar.

Durch welche Funktionalitäten können ansprechende Resultate erzeugt werden?

Über den Generalisierungsdienst WebGen-WPS werden sehr gute Algorithmen zur Ableitung geometrisch generalisierter OSM-Daten zur Verfügung gestellt, wie die Ergebnisse der praktischen Umsetzungen des Konzeptes „*MRDB-OSM*“ gezeigt haben.

Bei Aufruf des Dienstes „*BuildingSimplification*“ zur Vereinfachung von Gebäudegeometrien konnten ansprechende topologisch konsistente Resultate erzeugt werden, wobei der grundlegende Charakter von Geometrien zumeist sehr gut erhalten blieb. Allerdings scheint der zugrunde liegende Algorithmus noch relativ unflexibel entworfen. So wurden einerseits teils fehlerhafte Geometrien erzeugt, die nicht ohne weiteres in eine finale Karte übernommen werden können. Andererseits werden bestimmte Geometrien offensichtlich falsch verarbeitet, da beim Aufruf des Dienstes eine Fehlermeldung auftreten kann, die nicht nachvollziehbar ist. Diesbezüglich müssen zunächst einige Modifikationen am zugrunde liegenden Algorithmus vollzogen werden, bevor fehlerfreie Implementierungen möglich sind.

Insbesondere die nicht nachvollziehbare Fehlermeldung hat erheblichen Einfluss auf die Umsetzbarkeit der aufgezeigten Implementierungsansätze. Bevor ein WPS-Plug-in für Mapnik entwickelt wird, ist es unbedingt notwendig, dieses Problem zu beheben. Zwar muss ohnehin der Umgang mit auftretenden Fehlern implementiert werden, jedoch ist es

ratsam, so viele Fehlerquellen wie möglich im Vorfeld zu eliminieren. In diesem Kontext könnte vorweg eine Untersuchung aller vorhandenen Generalisierungsdienste angestrebt und auf Konsistenz und Flexibilität getestet werden.

Der Generalisierungsdienst „LineSmoothing“ zur Glättung linearer Geometrien hat hervorragende Ergebnisse erzeugt. Er ist ohne Einschränkungen zu empfehlen, zumal während der gesamten Versuche keine Fehler auftraten.

Während der praktischen Implementierungen des Konzeptes „MRDB-OSM“ wurde ein erheblicher Mangel beim Aufruf von WebGen-WPS deutlich. Da es derzeit unmöglich ist, zusätzliche Attribute bei der Verarbeitung von OSM-Daten in das Resultat zu übernehmen, sind arbeits- und zeitintensive Nachbearbeitungen notwendig. Innerhalb des WPS-Plug-in für OpenJUMP ist bereits die Auswahl verschiedener Profile vorgesehen. Diesbezüglich könnte dort ein OSM-Profil angelegt werden, in welchem die Übernahme des Kennzeichens „*osm_id*“ ermöglicht wird. So wäre eine uneingeschränkte Verwendung realisierbar. Ungeachtet dessen kann „MRDB-OSM“ derzeit bereits angewendet werden.

Die Ergebnisse in Bezug auf Integration verschiedener PostGIS-Funktionen sind sehr ernüchternd. Lediglich die Funktion „*ST_Simplify()*“ zur Vereinfachung linearer Geometrien ist anwendbar. Allerdings muss dies mit großer Vorsicht geschehen. Da es sich um eine simple Umsetzung des Douglas-Peucker-Algorithmus handelt, könnten topologisch inkorrekte Resultate erzeugt werden. Weitere Funktionen konnten sich nicht bewähren. Entweder wurden inkorrekte Resultate erzeugt (z.B.: „*ST_SimplifyPreserveTopology()*“) oder es war eine Anwendung in Bezug auf OSM-Daten schlicht nicht möglich (z.B.: „*ST_Length()*“). Dies gilt jedoch nur in Bezug auf triviale Anwendung der Funktionen. Gerade für die Verwendung von „*ST_Length()*“ besteht noch erhebliches Potential, indem sie in komplexe Funktionen eingebunden wird.

Wie kann die Anwendung kartographischer Generalisierung innerhalb des Projektes zusätzlich gesteigert werden?

Vor allem für die derzeitige Anwendung semantischer Auswahlkriterien sind erhebliche Mängel bei der Ableitung von OSM-Karten deutlich geworden. In Folge bietet es sich an, eine grafische Benutzeroberfläche zu entwerfen, wodurch visuelle Mängel direkt erkennbar und vermeidbar sind. Darüber hinaus ergeben sich Forschungsbereiche, in welchen klassische wissenschaftliche Erkenntnisse der Zeichenwahrnehmung (z.B.: Bollmann 1981) in Relation zu den Bedürfnissen von OSM-Kartendarstellungen untersucht und

angepasst werden. In diesem Zusammenhang sind äußere Umstände, wie Ausgabe über Monitor oder automatisiertes Rendering als Bedürfnisse zu verstehen. Ebenso mangelhaft angewandte Vorgänge wie *Überhaltung* und *Signaturierung* sollten in diese Untersuchungen parallel eingeschlossen werden.

Überdies wurde im Zuge theoretischer Überlegungen der Vorschlag zur Entwicklung einer OSM-Generalisierungscommunity unterbreitet. Damit würde eine Plattform zur Sammlung, Überprüfung und Diskussion möglicher Implementierungen entstehen. Vor allem eine Beteiligung der Wissenschaft könnte den gesamten Grad angewandter Generalisierungsmaßnahmen erheblich steigern, sodass Experten und Anwender gemeinsam optimierte Lösungen für das Gemeinschaftsprojekt OpenStreetMap finden. Ebenso würde die Entwicklung neuer Funktionalitäten und Implementierungen schneller voran getrieben.

Aufgrund der Vielzahl von Projekten, die bereits entwickelt wurden, ist ersichtlich, dass bei praktischer Umsetzung mit einer hohen Beteiligung zu rechnen ist. Dadurch könnten sehr rasch Ergebnisse erzielt und jedem zugänglich gemacht werden. Dieser Ansatz entspricht außerdem exakt der grundlegenden Idee des OpenStreetMap-Projektes, da sich die gesamte OSM-Gemeinschaft an der Weiterentwicklung von Generalisierung im Kontext von OpenStreetMap beteiligen könnte.

Einer praktischen Umsetzung des Ansatzes sollte eine wissenschaftliche Untersuchung vorausgehen, wodurch die Ziele und Ansprüche einer OSM-Generalisierungscommunity exakt definiert werden.

6.2 Fazit

Das primäre Ziel des Projektes OpenStreetMap ist bis dato die Bereithaltung freier, nicht proprietärer Geodaten. Aus diesem Grund, aber auch, da jeder die Möglichkeit hat, selbstständig an diesem Projekt mitzuwirken, ergeben sich zahlreiche Vorteile im Vergleich zu anderen kommerziellen Webkarten, wie „Google maps.“ Es konnte jedoch gezeigt werden, dass wissenschaftliche Aspekte der kartographischen Generalisierung bei der Visualisierung dieser freien Daten bisher viel zu spärlich beachtet werden. Dadurch können erhebliche visuelle sowie inhaltliche Mängel in den resultierenden Kartenprodukten entstehen. Gerade um weiterhin eine breite Akzeptanz des gesamten Projektes zu erzielen, müssen unterschiedlichste Kriterien der kartographischen Generalisierung sukzessive in die Visualisierung von OpenStreetMap-Daten integriert werden.

Mit dieser Arbeit wurde ein erster Grundstein der Forschungen in diesem Bereich gelegt. Der Bedarf an wissenschaftlicher Beteiligung im Bereich der Generalisierung von OSM-Daten ist enorm hoch. Deshalb sollten entworfene Implementierungen unbedingt umgesetzt werden. Darüber hinaus müssen jedoch auch zusätzliche Lösungen entwickelt werden, da nicht alle auftretenden Mängel durch diese Arbeit abgedeckt werden konnten.

7 Quellennachweise

7.1 Literaturverzeichnis

- [1] ALAMEH, NADINE (2003): *Chaining geographic information Web services*. IEEE Computer Society, Internet Computing, Volume 7, Issue 5, S.22-29, September 2003
- [2] ARNBERGER, ERIK; KRETSCHMER, INGRID (1975): *Wesen und Aufgaben der Kartographie – Topographische Karten*. Teil1/Textband, Franz Deuticke, Wien
- [3] BEARD, KATE (1991): *Constraints on rule formation*. In: B. BUTTENFIELD & R. B. McMASTER (Hrsg.): *Map Generalization: Making Rules for Knowledge Representation*. S. 121–135. Longman, London
- [4] BERTIN, JACQUES (1967): *Sémiologie graphique*. Gauthier-Villars, Paris
- [5] BOLLMANN, JÜRGEN (1981): *Aspekte kartographischer Zeichenwahrnehmung*. Kirschbaum Verlag, Bonn-Bad-Godesberg
- [6] BOLLMANN, JÜRGEN; KOCH, WOLF GÜNTHER (2001): *Lexikon der Kartographie und Geomatik – in zwei Bänden*. Spektrum Akademischer Verlag GmbH, Heidelberg – Berlin
- [7] BRAUNER, JOHANNES (2008): *Web Processing Service Schnittstelle für GIS Funktionalitäten*. Münster: Westfälische Wilhelms-Universität Münster, Institut für Geoinformatik, Diplomarbeit, 2008
- [8] BURGHARDT, DIRK; NEUN, MORITZ (2006): *Automated sequencing of generalisation services based on collaborative filtering*. In: 4th International Conference, Geographic Information Science, IfGIprints 28, S. 41-46
- [9] CHILTON, STEVE (2010): *What we'd like to do with Mapnik*. State of the Map 2010, 4th Annual International OpenStreetMap Conference, Girona, Spain. Zugriff am 16.03.2011, URL http://media.mapnik.org/images/mcs01/chilton_codesprint-what-I%E2%2580%2599d-like-to-do-with-mapnik.pdf

- [10] DOUGLAS, DAVID H.; PEUCKER, THOMAS K. (1973): *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*. In: *The Canadian Cartographer*, Vol. 10, No. 2, S. 112 - 122
- [11] ECKERT, MAX (1921): *Die Kartenwissenschaft – Forschungen und Grundlagen zu einer Kartographie als Wissenschaft*. Vereinigung wissenschaftlicher Verleger Walter de Gruyter & Co, Berlin & Leipzig
- [12] EDWARDS, ALISTAIR, BURGHARDT, DIRK; NEUN, MORITZ (2005): *Interoperability in Map Generalisation Research*. Proceedings of the International Symposium on Generalization of Information, S. 160 – 174, Berlin
- [13] EDWARDS, ALISTER; BURGHARDT, DIRK; BOBZIEN, MATTHIAS; HARRIE, LARS; LEHTO, LASSI; REICHENBACHER, TUMASCH; SESTER, MONIKA; WEIBEL ROBERT (2003): *Map Generalisation Technology: Addressing the need for a common research platform*. Proceedings of the 21st International Cartographic Conference (ICC), Durban, South Africa, 10 – 16 August 2003
- [14] FOERSTER, THEODOR (2010): *Web-based architecture for on-demand maps – integrating meaningful generalization processing*. Enschede: International Institute for Geo-Information Science and Earth observation, Enschede, Dissertation, 2010
- [15] FOERSTER, THEODOR; BURGHARDT, DIRK; NEUN, MORITZ; REGNAULD, NICOLAS; SWAN, JERRY; WEIBEL, ROBERT (2008): *Towards an Interoperable Web Generalisation Services Framework – Current Work in Progress*. Proceedings 11th ICA Workshop on Generalisation and Multiple Representation, Montpellier
- [16] GRÜNREICH, DIETMAR (1985): *Untersuchungen zu den Datenquellen und zur rechnergestützten Herstellung des Grundrisses großmaßstäbiger topographischer Karten*. Hannover: Universität Hannover, WAVH Nr.132, Dissertation, 1985
- [17] HAKE, GÜNTER; GRÜNREICH, DIETMAR; MENG, LIQIU (2002): *Kartographie: Visualisierung raum-zeitlicher Informationen*. 8., vollst. neu bearb. und erw. Auflage, Walter de Gruyter & Co, Berlin

- [18] HAKLAY, MORDECHAI (2009): *OpenStreetMap and Ordnance Survey Meridian 2 – Progress Maps*. Zugriff am 03.03.2011, URL <http://povesham.wordpress.com/2009/11/14/openstreetmap-and-ordnance-survey-meridian-2-progress-maps/>
- [19] HAKLAY, MORDECHAI (2010): *How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets*. *Environment and Planning B: Planning and Design* 37(4) 682-703
- [20] HAKLAY, MORDECHAI; BASIOUKA, SOFIA; ANTONIOU, VYRON; ATHER, AAMER (2010): *How Many Volunteers Does it Take to Map an Area Well? The Validity of Linus' Law to Volunteered Geographic Information*. In: *The Cartographic Journal*, Vol. 47, No. 4, S. 315-322, November 2010
- [21] HARRIE, LARS; WEIBEL, ROBERT (2007): *Modelling the Overall Process of Generalisation*. In: W. M. MACKANESS, A. RUAS, L. T. SARJAKOSKI (Hrsg.): *Generalisation of Geographic Information: Cartographic Modelling and Applications*, Elsevier Ltd., 2007
- [22] KILPELÄINEN, TIINAA (1992): *Multiple Representations and Knowledge-Based Generalization of Topographical Data*. In: T. C. WAUGH & R. G. HEALEY (Hrsg.): *Advances in GIS Research, Proceedings of the 6th International Symposium on Spatial Data Handling*. S. 882-893, IGU Commission on GIS and Association for Geographic Information, Edinburgh, GB
- [23] KOUNADI, OURANIA (2009): *Assessing the quality of OpenStreetMap data*. London: University College of London Department of Civil, Environmental And Geomatic Engineering, Masterarbeit, 2009
- [24] LAMY, SYLVIE; RUAS, ANNE; DEMATEAU, YVES; JACKSON, MIKE; MACKANESS, WILLIAM; WEIBEL, ROBERT (1999): *The Application of Agents in Automated Map Generalisation*. 19th ICA Meeting Ottawa, Aug 14-21, 1999
- [25] LUDWIG, INA (2010): *Abbildung von Straßendaten für Qualitätsuntersuchungen - Ein Vergleich von OpenStreetMap mit Navteq*. Bonn: Rheinische Friedrich-Wilhelms Universität Bonn, Geographisches Institut, Diplomarbeit, 2010

- [26] McMASTER, ROBERT B.; SHEA, K. STUART (1992): *Generalization in Digital Cartography*. Association of American Geographers, Washington
- [27] MENG, LIQUI (2008): *Kartographie im Umfeld moderner Informations- und Medientechnologien*. In: *Kartographische Nachrichten*, (KN 1/2008):3-10, 2008
- [28] MONMONIER, MARK S. (1991): *How to lie with maps*. University of Chicago Press, Chicago
- [29] MORRISON, JOEL (1974): *A Theoretical Framework for Cartographic Generalization with Emphasis on the Process of Symbolization*. In: *International Yearbook of Cartography*, Number 14, S. 115 - 127
- [30] NEIS, PASCAL; ZIELSTRA, DENNIS; ZIPF, ALEXANDER; STRUNCK, ALEXANDER (2010): *Empirische Untersuchungen zur Datenqualität von OpenStreetMap – Erfahrungen aus zwei Jahren Betrieb mehrerer OSM-Online-Dienste*. AGIT 2010. Symposium für Angewandte Geoinformatik, Salzburg
- [31] NEUN, MORITZ; BURGHARDT, DIRK (2005): *Web Services for an open generalisation research platform*. 8th ICA Workshop on Generalisation and Multiple Representation, A Coruña, Spain
- [32] OGC - OPEN GEOSPATIAL CONSORTIUM (2002): *Styled Layer Descriptor Implementation Specification*. Open Geospatial Consortium Inc., Version 1.0.0, 2002-09-19, Zugriff am 04.04.2011, URL <http://www.opengeospatial.org/standards/sld>
- [33] OGC - OPEN GEOSPATIAL CONSORTIUM (2005): *OpenGIS Web Services Common Specification*. Open Geospatial Consortium Inc., Version 1.0.0, 2005-11-22, Zugriff am 04.04.2011, URL <http://www.opengeospatial.org/standards/common>
- [34] OGC - OPEN GEOSPATIAL CONSORTIUM (2007): *Open GIS Web Processing Services*. Open Geospatial Consortium Inc., Version 1.0.0, 2007-06-08, Zugriff am 04.04.2011, URL <http://www.opengeospatial.org/standards/wps>
- [35] PETERSON, MICHAEL (2005): *Maps and the Internet: An Introduction*. In: M. PETERSON (Hrsg.): *Maps and the Internet*., Elsevier Ltd., 2005
- [36] PÖNISCH, JENS (2010¹⁰): *OpenStreetMap*. Tutorial, Zugriff am 01.04.2011, URL <http://www.qucosa.de/fileadmin/data/qucosa/documents/5696/data/index.html>

10 Diese Jahreszahl bezieht sich die letzte Aktualisierung der Homepage

- [37] RAMM, FREDERIK; TOPF, JOCHEN (2010): *OpenStreetMap – Die freie Weltkarte nutzen und mitgestalten*. 3. überarbeitete und erweiterte Auflage, Lehmanns Media, Berlin
- [38] RATAJSKI, LECH (1967): *Phenomens des points de generalisation*. In: Internationales Jahrbuch der Kartographie, Band 7, Gütersloh, S.143-152
- [39] REGNAULD, NICOLAS; McMASTER, ROBERT B. (2007): *A Synoptic View of Generalisation Operators*. In: W. M. MACKANESS, A. RUAS, L. T. SARJAKOSKI (Hrsg.): *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier Ltd., 2007
- [40] ROBINSON, ARTHUR; SALE, RANDAL; MORRISON, JOEL (1978): *Elements of Cartography*. John Wiley & Sons Inc, 4th Edition, New York, US
- [41] SARJAKOSKI, L. TIINA (2007): *Conceptual Models of Generalisation and Multiple Representation*. In: W. M. MACKANESS, A. RUAS, L. T. SARJAKOSKI (Hrsg.): *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier Ltd., 2007
- [42] SHEA, K. STUART; McMASTER, ROBERT B. (1989): *Cartographic Generalization in a Digital Environment: When and How to generalize*. 9th International Symposium on Computer-Assisted Cartography, AUTO-CARTO IX, Baltimore, American Society of Photogrammetry and Remote Sensing and American Congress on Surveying and Mapping, S.56-67
- [43] SGK- SCHWEIZERISCHE GESELLSCHAFT FÜR KARTOGRAPHIE (2002): *Topografische Karten - Kartengrafik und Generalisierung*. Schweizerische Gesellschaft für Kartographie, Kartografische Publikationsreihe Nr.16
- [44] TÖPFER, FRIEDRICH (1979): *Kartographische Generalisierung*. 2. Auflage, VEB Hermann Haack, Geographisch-Kartographische Anstalt Gotha/Leipzig
- [45] W3C – WORLD WIDE WEB CONSORTIUM (2004): *Web Services Glossary*. W3C Working Group Note 11 February 2004, Zugriff am 05.03.2011, URL <http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/>
- [46] WRIGHT, JOHN K. (1942): *Map makers are human – comments on the subject in maps*. American Geographical Society, New York

- [47] ZOLLINGER, STEFAN (2008): *Kartenkritik an OpenStreetMap - Eine systematische Beurteilung der durch Mapnik visualisierten freien Weltkarte mit Fokus auf die Schweiz*. Zürich: Institut für Kartografie, ETH Zürich, Bericht zur Semesterarbeit in der Lehrveranstaltung Geomatics, 2008

7.2 Internetquellennachweis (ohne eindeutige Autoren)

- [1] *Aktuellster kompletter Datenauszug - „planet-file“* - Zugriff am 31.03.2011
URL <http://planet.openstreetmap.org/planet-latest.osm.bz2>
- [2] *Allgemeines freies Lexikon zu OpenStreetMap* - Zugriff am 18.04.2011
URL http://wiki.openstreetmap.org/wiki/Main_Page
- [3] *Blog mit Erläuterungen von Arbeitsabläufen zu OpenStreetMap* - Zugriff am 02.03.2011
URL <http://weait.com/>
URL <http://weait.com/content/build-your-own-openstreetmap-server>
- [4] *cURL- Ansprache von Server über Kommandozeile* - Zugriff am 18.04.2011
URL <http://curl.haxx.se/>
- [5] *Datenimports in OSM* - Zugriff am 08.03.2011
URL <http://wiki.openstreetmap.org/wiki/Import>
- [6] *Erläuterungen zum Begriff „Geometriespalte“ in PostGIS* - Zugriff am 04.04.2011
URL www.giswiki.org/wiki/PostGIS_Tutorial#Tabelle_mit_Geometriespalte_erstellen
- [7] *Lizenzwechsel von OpenStreetMap* - Zugriff am 03.03.2011
URL <http://www.openstreetmap.de/lizenzaenderung.html>
- [8] *Mapnik-Tile-Server* - Zugriff am 03.03.2011
URL (exemplarisch) <http://tile.openstreetmap.org/z/x/y.png>
URL (Kachel der Zoomstufe 0) <http://tile.openstreetmap.org/0/0/0.png>
- [9] *OpenStreetMap* - Zugriff am 18.04.2011
URL www.openstreetmap.org
- [10] *Osm2pgsql – Übertragung von OSM-Daten in PostGIS* - Zugriff am 19.04.2011
URL <http://wiki.openstreetmap.org/wiki/DE:Osm2pgsql>

- [11] *Osmarender - Erläuterungen zur Software* - Zugriff am 18.04.2011
URL <http://wiki.openstreetmap.org/wiki/Osmarender/Howto>
- [12] *Osmosis – Hilfsprogramm zur Verarbeitung von OSM-Daten* - Zugriff am 14.03.2011
URL <http://wiki.openstreetmap.org/wiki/Osmosis>
- [13] *Erläuterungen zur Maßstabsausgabe in Mapnik* - Zugriff am 25.03.2011
URL <http://trac.mapnik.org/wiki/ScaleAndPpi>
- [14] *PostGIS – allgemeine Dokumentation* - Zugriff am 05.03.2011
URL <http://postgis.refractions.net/documentation/manual-1.5>
- [15] *PostGIS – Dokumentation der Funktion „ST_Simplify“* - Zugriff am 31.03.2011
URL http://postgis.refractions.net/documentation/manual-1.5/ST_Simplify.html
- [16] *PostGIS – Dokumentation der Funktion „ST_SimplifyPreserveTopology“*
- Zugriff am 31.03.2011
URL http://postgis.refractions.net/documentation/manual-1.5/ST_SimplifyPreserveTopology.html
- [17] *PostgreSQL – Datentypen* - Zugriff am 05.03.2011
URL www.postgresql.org/docs/8.2/static/datatype-numeric.html
- [18] *SLD-Standard* - Zugriff am 31.03.2011
URL <http://www.opengeospatial.org/standards/sld>
- [19] *Statistische Daten zum Projekt OpenStreetMap* - Zugriff am 03.03.2011
URL <http://wiki.openstreetmap.org/wiki/Stats>
- [20] *Tiles@Home* - Zugriff am 18.04.2011
URL (exemplarisch) <http://tah.openstreetmap.org/Tiles/tile/z/x/y.png>
URL (Kachel der Zoomstufe 0) <http://tah.openstreetmap.org/Tiles/tile/0/0/0.png>

8 Anhang

Anhang A - Beispiel eines Pythonskripts zum Ausführen von Mapnik

Dieses Pythonskript wurde für die exemplarischen Implementierung verwendet.
(In den letzten sechs Zeilen wurde eine zusätzliche Ausgabe & Berechnung des aktuellen Maßstabes eingefügt.)

→ die verwendeten XML-Stylefiles sind auf der beiliegenden CD zu finden
(/Anhang/Anhang A/Style-Dateien)

```
#!/usr/bin/python
#
# Generates a single large PNG image for a UK bounding box
# Tweak the lat/lon bounding box (ll) and image dimensions
# to get an image of arbitrary size.
#
# To use this script you must first have installed mapnik
# and imported a planet file into a Postgres DB using
# osm2pgsql.
#
# Note that mapnik renders data differently depending on
# the size of image. More detail appears as the image size
# increases but note that the text is rendered at a constant
# pixel size so will appear smaller on a large image.

import mapnik
import sys, os

if __name__ == "__main__":
    try:
        mapfile = os.environ['MAPNIK_MAP_FILE']
    except KeyError:
        #Here are the different Outputpoptions implemented
        # just uncomment the designated

        #standard
        mapfile = "normal_style.xml"
        map_uri = "Bilder_DA/my_image.png"

        #WebGen-WPS
        #####

        #generalization
        #mapfile = "osm_generalized_style.xml"
```

```

#map_uri = "Bilder_DA/my_generalized_image.png"
#--> the JOIN can be found between the lines 8138 - 8144

#just buildings
#mapfile = "osm_just_buildings_style.xml"
#map_uri = "Bilder_DA/just_buildings.png"

#just generalized buildings
#mapfile = "osm_just_buildings_generalized_style.xml"
#map_uri = "Bilder_DA/just_buildings_generalized.png"

#just rivers wps
#mapfile = "osm_just_rivers_wps_style.xml"
#map_uri = "Bilder_DA/just_rivers_wps.png"
#generalized rivers in complete map
#mapfile = "osm_rivers_generalized_style.xml"
#map_uri = "Bilder_DA/rivers_generalized.png"

#PostGIS-Functions

#selection
#mapfile = "osm_just_rivers_selection_style.xml"
#map_uri = "Bilder_DA/PostGIS-Funktionen/just_rivers_selection.png"
#accent
#mapfile = "osm_just_rivers_accent_style.xml"
#map_uri = "Bilder_DA/PostGIS-Funktionen/just_rivers_accent.png"
#simplify rivers (complete map)
#mapfile = "osm_rivers_simplify_style.xml"
#map_uri = "Bilder_DA/PostGIS-Funktionen/rivers_simplify.png"
#simplify just rivers
#mapfile = "osm_just_rivers_simplify_style.xml"
#map_uri = "Bilder_DA/PostGIS-Funktionen/just_rivers_simplify.png"
#Building simplification
#mapfile = "osm_just_buildings_generalized_postgis_style.xml"
#map_uri = "Bilder_DA/PostGIS-Funktionen/just_buildings_generalized_postgis.png"

#-----
# Change this to the bounding box you want
#
ll = (13.72088, 51.02682, 13.72873, 51.03166) #dresden campus ca. 1:10.000 --> 8cm
#ll = (13.70772, 51.0189, 13.73914, 51.03825) #dresden campus ca. 1:25.000 --> 14 cm
#ll = (13.381, 50.845, 14.139, 51.271) #dresden greater area (Testarea) ca. 1:1Mio --> 8cm
#ll = (13.6291, 50.9965, 13.8804, 51.1436) #dresden...for line simplification ca. 350.000 --> 8cm
#-----

z = 10

imgx = 50 * z
imgy = 50 * z

m = mapnik.Map(imgx,imgy)

```

```
mapnik.load_map(m,mapfile)
    prj = mapnik.Projection("+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m
+nadgrids=@null +no_defs +over")
c0 = prj.forward(mapnik.Coord(11[0],11[1]))
c1 = prj.forward(mapnik.Coord(11[2],11[3]))
if hasattr(mapnik,'mapnik_version') and mapnik.mapnik_version() >= 800:
    bbox = mapnik.Box2d(c0.x,c0.y,c1.x,c1.y)
else:
    bbox = mapnik.Envelope(c0.x,c0.y,c1.x,c1.y)
m.zoom_to_box(bbox)
im = mapnik.Image(imgx,imgy)
mapnik.render(m, im)
view = im.view(0,0,imgx,imgy) # x,y,width,height
view.save(map_uri,'png')
print "The scale of this map is: ", m.scale()
lengthx_reality = m.scale()*imgx
lengthy_reality = m.scale()*imgy

#an Output of the actual Scale in the comandlinewindow
#--> the calculation of the printed scale("printed_scale") depends manly on the printed size ("printed_size")
print "So the whole map has a size of: ", lengthx_reality , " meters x ", lengthy_reality, " meters"
printed_size = 8
printed_scale = lengthx_reality*100/printed_size
print "Real Scale is for a printed size of:", printed_size, " --> 1 : ", printed_scale
```

Anhang B - Beispiel einer OGC-konformen XML-Datei zum Senden von Geodaten

Diese Datei enthält alle notwendigen Parameter, die für eine Execute-Anfrage am WebGe-n-WPS notwendig sind

- Execute-Befehl
- Eingabedaten
- Datenausgabeform

```
<ns:Execute service="WPS" version="1.0.0" xmlns:ns="http://www.opengis.net/wps/1.0.0">
  <ns1:Identifier xmlns:ns1="http://www.opengis.net/ows/1.1">ch.unizh.geo.webgen.service.BuildingSimplification</ns1:Identifier>
  <ns:DataInputs>
    <ns:Input>
      <ns1:Identifier xmlns:ns1="http://www.opengis.net/ows/1.1">minlength</ns1:Identifier>
      <ns1:Title xmlns:ns1="http://www.opengis.net/ows/1.1">minlength</ns1:Title>
      <ns1:Abstract xmlns:ns1="http://www.opengis.net/ows/1.1">minimum length</ns1:Abstract>
      <ns:Data>
        <ns:LiteralData>10.0</ns:LiteralData>
      </ns:Data>
    </ns:Input>
    <ns:Input>
      <ns1:Identifier xmlns:ns1="http://www.opengis.net/ows/1.1">geom</ns1:Identifier>
      <ns1:Title xmlns:ns1="http://www.opengis.net/ows/1.1">geom</ns1:Title>
      <ns1:Abstract xmlns:ns1="http://www.opengis.net/ows/1.1">layer with geometries</ns1:Abstract>
      <ns:Data>
        <ns:ComplexData>
          <wps:FeatureCollection xmlns:wps="http://www.icaci.org/genmr/wps">
            <wps:Feature>
              <wps:AttributeGEOMETRY>
                <wps:Value srsName="0" wpstype="AttributeTypeGeometryPolygon">
                  <gml:outerBoundaryIs xmlns:gml="http://www.opengis.net/gml">
                    <gml:LinearRing>
                      <gml:coord>
                        <gml:X>783945.0299329287372529506683349609375</gml:X>
                        <gml:Y>151840.2400824803044088184833526611328125</gml:Y>
                      </gml:coord>
                      <gml:coord>
                        <gml:X>783953.33996594883501529693603515625</gml:X>
                        <gml:Y>151839.4899533681455068290233612060546875</gml:Y>
                      </gml:coord>
                      <gml:coord>
                        <gml:X>783952.999935419647954404354095458984375</gml:X>
                        <gml:Y>151834.45994532821350730955600738525390625</gml:Y>
                      </gml:coord>
                      <gml:coord>
                        <gml:X>783963.60999526944942772388458251953125</gml:X>
```

```
<gml:Y>151833.18993798433803021907806396484375</gml:Y>
</gml:coord>
<gml:coord>
  <gml:X>783961.320087259518913924694061279296875</gml:X>
  <gml:Y>151810.46001400394015945494174957275390625</gml:Y>
</gml:coord>
<gml:coord>
  <gml:X>783956.3500284538604319095611572265625</gml:X>
  <gml:Y>151810.91994300144142471253871917724609375</gml:Y>
</gml:coord>
<gml:coord>
  <gml:X>783955.95997658907435834407806396484375</gml:X>
  <gml:Y>151807.0499717356287874281406402587890625</gml:Y>
</gml:coord>
<gml:coord>
  <gml:X>783941.94999329070560634136199951171875</gml:X>
  <gml:Y>151808.45992426635348238050937652587890625</gml:Y>
</gml:coord>
<gml:coord>
  <gml:X>783945.0299329287372529506683349609375</gml:X>
  <gml:Y>151840.2400824803044088184833526611328125</gml:Y>
</gml:coord>
</gml:LinearRing>
</gml:outerBoundaryIs>
</wps:Value>
</wps:AttributeGEOMETRY>
</wps:Feature>
</wps:FeatureCollection>
</ns:ComplexData>
</ns>Data>
</ns:Input>
</ns>DataInputs>
<ns:ResponseForm>
  <ns:ResponseDocument lineage="false" storeExecuteResponse="true" status="false">
    <ns:Output asReference="true" mimeType="text/xml" encoding="UTF-8">
      <ns1:Identifier xmlns:ns1="http://www.opengis.net/ows/1.1">result</ns1:Identifier>
      <ns1:Title xmlns:ns1="http://www.opengis.net/ows/1.1">result</ns1:Title>
      <ns1:Abstract xmlns:ns1="http://www.opengis.net/ows/1.1">simplified building polygons</ns1:Abstract>
    </ns:Output>
  </ns:ResponseDocument>
</ns:ResponseForm>
</ns:Execute>
```

Anhang C - Codebeispiel in C++ zum WPS-Aufruf mittels cURL (libcurl)

```
/*
    Very simple example to request the WebGen-WPS via C++ using cURL
    --> reads a XML-File that contains geographic data and informations about the server
    --> just sends that data to WebGen-WPS
    --> analysis of response is not implemented

    Author: Ralf Klammer
    Source: Example by Todd Papaioannou (http://www.luckyspin.org/?p=28 [January 3rd, 2006])
*/

#include <string>
#include <fstream>
#include <iostream>
#include "curl/curl.h"

using namespace std;

// This is the writer call back function used by curl
static int writer(char *data, size_t size, size_t nmemb, std::string *buffer)
{
    // What we will return
    int result = 0;
    // Is there anything in the buffer?
    if (buffer != NULL)
    {
        // Append the data to the buffer
        buffer->append(data, size * nmemb);
        // How much did we write?
        result = size * nmemb;
    }
    return result;
}

int main()
{
    // main processes for datafile
    //read the XML-File & write to a string
    fstream f;
    f.open("/media/Speicher/WPS_Linux/Workspacelfk_WebGen/Serveransprachen/data.xml", ios::in);
    string s, datei = "";
    while (!f.eof())
    {
        getline(f, s);
        datei = datei + s;
    }
}
```

```
f.close();
//transform string to char (cause curl doesn't read string)
const char *data;
data = datei.c_str();

// main part where curl is used
//set up serveradress
const char *url;
url = "http://kartographie.geo.tu-dresden.de/webgen_wps/wps";

//initialise curl
CURL *curl;
CURLcode result;
string buffer;

// initialise curl handling
curl = curl_easy_init();

// set up all curl-options
if (curl)
    {
        curl_easy_setopt(curl, CURLOPT_URL, url);
        curl_easy_setopt(curl, CURLOPT_HEADER, "Content-Type: text/xml");
        curl_easy_setopt(curl, CURLOPT_POSTFIELDS, data);
        curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, writer);
        curl_easy_setopt(curl, CURLOPT_WRITEDATA, &buffer);
    }

// perform server request
result = curl_easy_perform(curl);

// cleanup
curl_easy_cleanup(curl);

// print the response to commandline
cout << buffer << "\n";
}
```

Anhang D - OpenStreetMap-Karten im Maßstabsbereich von etwa 1:25.000

Gebäudegeneralisierung (-vereinfachung) mittels WebGen-WPS „BuildingSimplification“ über die Implementierung einer „MRDB-OSM“

Abbildungen: 8.1, 8.2, 8.3, 8.4

$B_{\text{box}} = (13.70772, 51.0189, 13.73914, 51.03825)$

$M_B = 7,00$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 14 cm

$M_A \approx 1:25.000$





Abbildung 8.2: Karte mit generalisierten Gebäudedaten

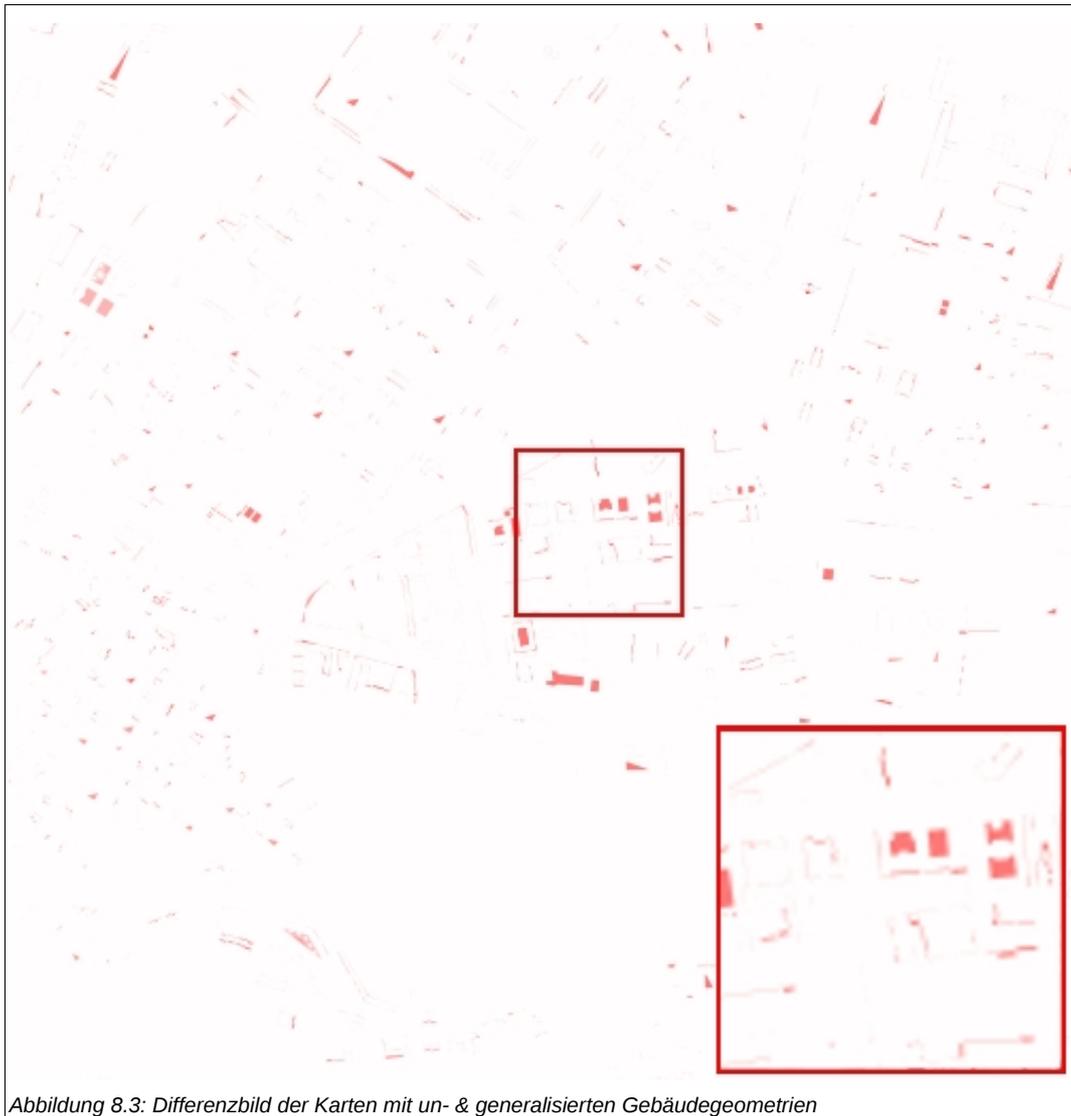


Abbildung 8.3: Differenzbild der Karten mit un- & generalisierten Gebäudegeometrien

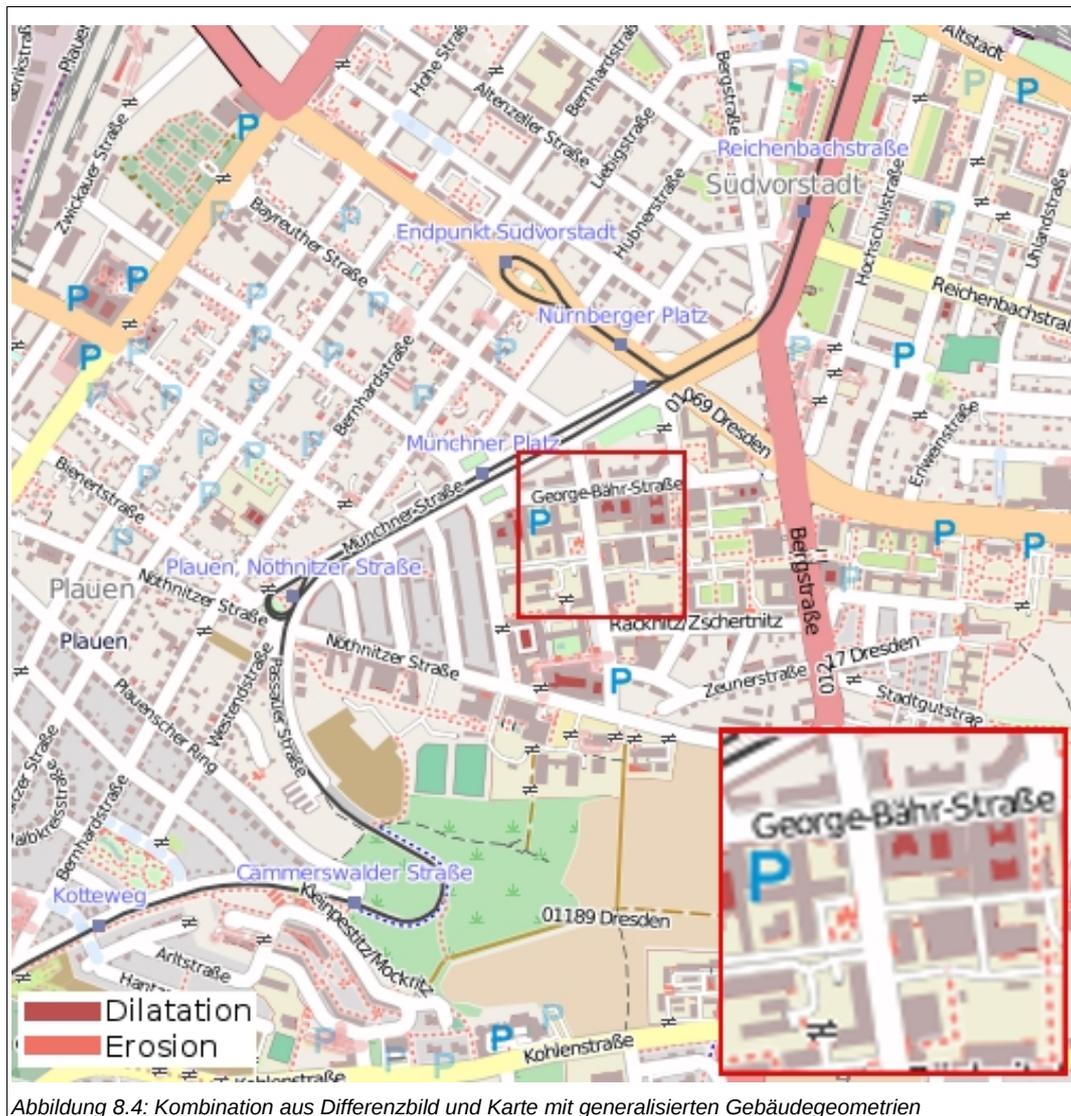


Abbildung 8.4: Kombination aus Differenzbild und Karte mit generalisierten Gebäudegeometrien

Anhang E - Gebäudegeometrien im Maßstabsbereich von etwa 1:25.000

Gebäudegeneralisierung (-vereinfachung) mittels WebGen-WPS „BuildingSimplification“ über die Implementierung einer „MRDB-OSM“

Abbildungen: 8.5, 8.6, 8.7, 8.8

$B_{\text{box}} = (13.70772, 51.0189, 13.73914, 51.03825)$

$M_B = 7,00$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 14 cm

$M_A \approx 1:25.000$



Abbildung 8.5: Reine, unbearbeitete Gebäudegeometrien



Abbildung 8.6: Reine, Generalisierte Gebäudegeometrien

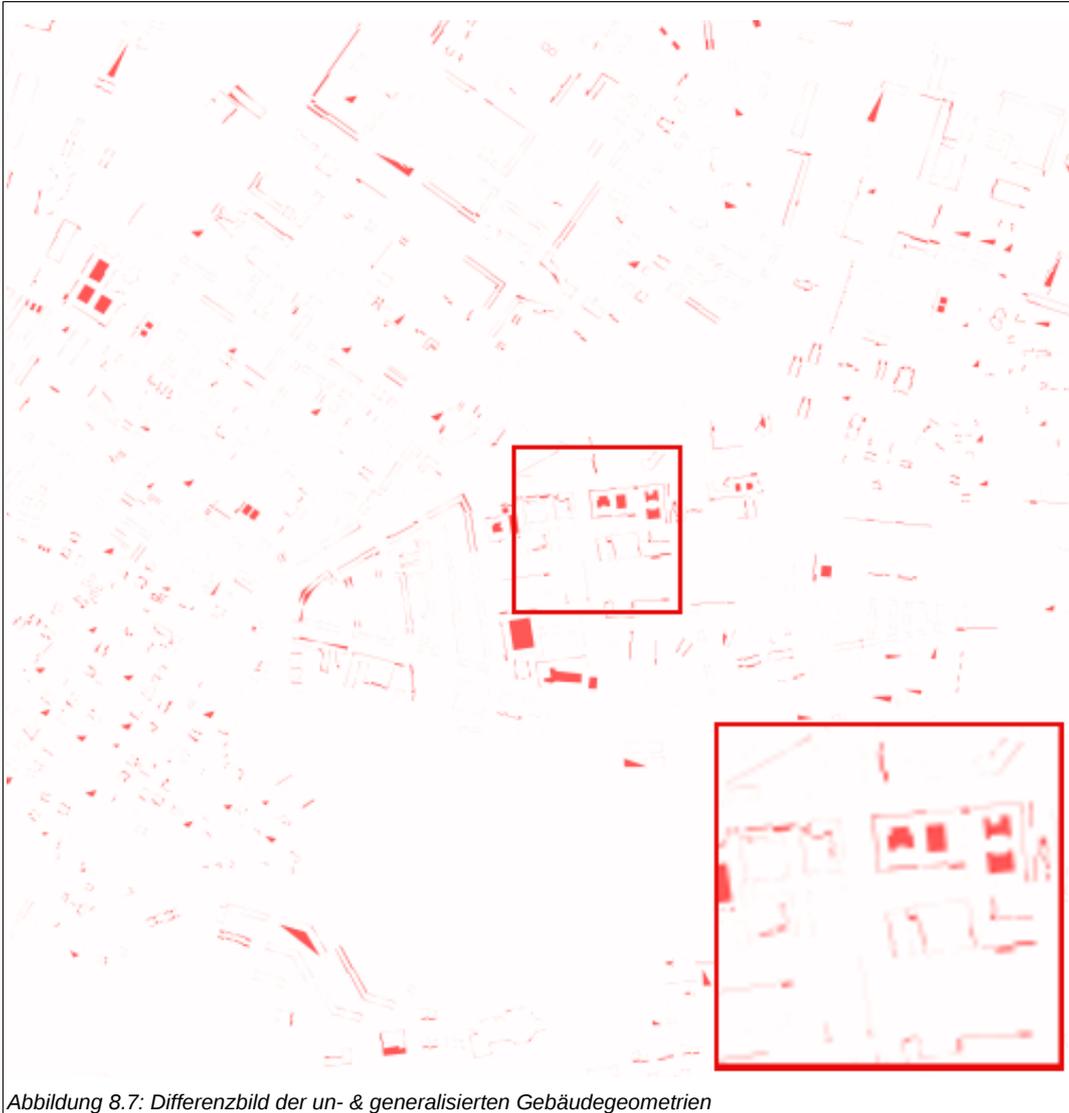


Abbildung 8.7: Differenzbild der un- & generalisierten Gebäudegeometrien



Anhang F - OpenStreetMap-Karten im Maßstabsbereich von etwa 1:10.000

Gebäudegeneralisierung (-vereinfachung) mittels WebGen-WPS „BuildingSimplification“ über Implementierung einer „MRDB-OSM“

Abbildungen: 8.9, 8.10, 8.11, 8.12, 8.13, 8.14, 8.15, 8.16

$B_{\text{box}} = (13.72088, 51.02682, 13.72873, 51.03166)$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 8 cm

$M_B = 1,75 M_A \approx 1:10.000$

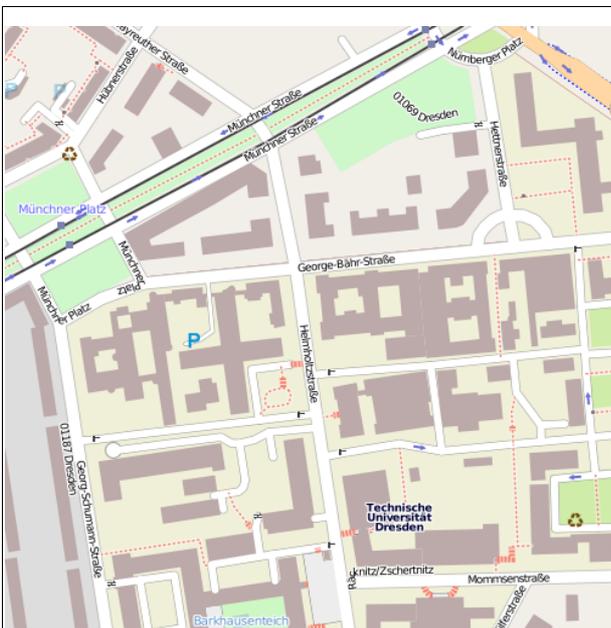


Abbildung 8.9: Karte mit unbearbeiteten Gebäudedaten

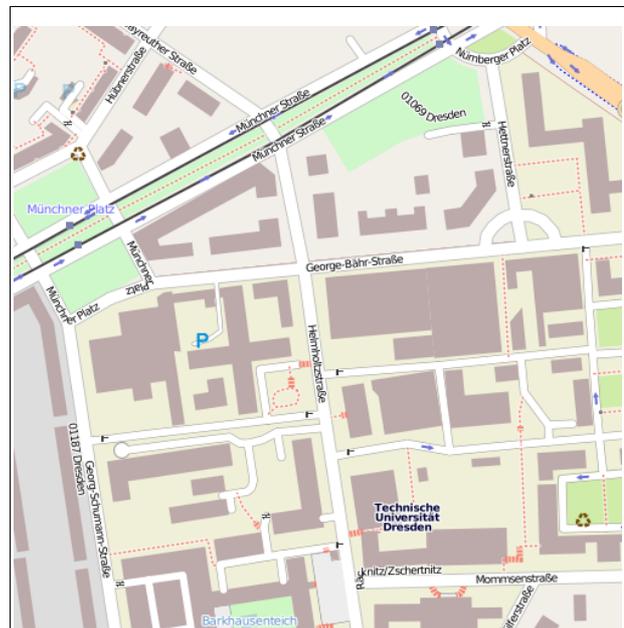


Abbildung 8.10: Karte mit generalisierten Gebäudedaten



Abbildung 8.11: Differenzbild der Karten mit un- & generalisierten Gebäudegeometrien

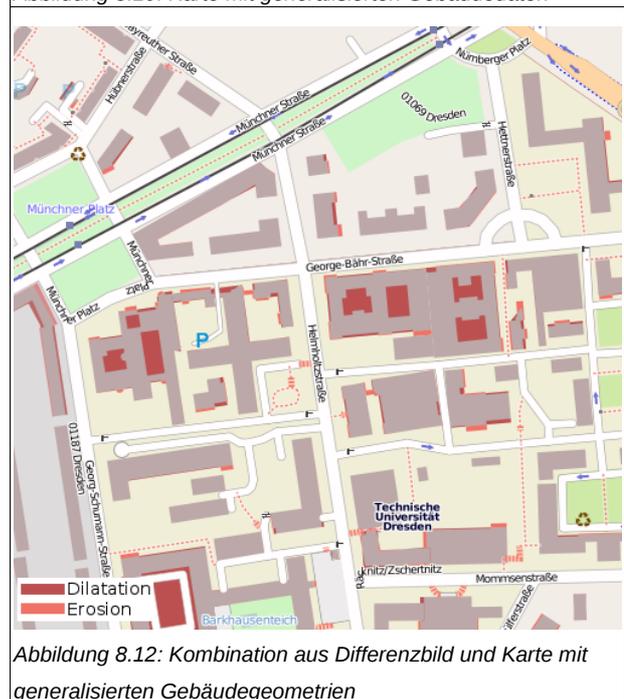


Abbildung 8.12: Kombination aus Differenzbild und Karte mit generalisierten Gebäudegeometrien

Darstellung reiner Gebäudegeometrien im Maßstabsbereich von etwa 1:10.000



Abbildung 8.13: Reine, unbearbeitete Gebäudegeometrien



Abbildung 8.14: Generalisierte Gebäudegeometrien



Abbildung 8.15: Differenzbild der un- & generalisierten Gebäudegeometrien



Abbildung 8.16: Kombination aus Differenzbild und generalisierten Gebäudegeometrien

Anhang G - Liniengeneralisierung mittels WebGen-WPS-Service „LineSmoothing“

Linienglättung mittels WebGen-WPS „LineSmoothing“ über Implementierung einer „MRDB-OSM“

Abbildungen: 8.17, 8.18, 8.19, 8.20, 8.21, 8.22, 8.23, 8.24, 8.25

$B_{\text{box}} = (13.6291, 50.9965, 13.8804, 51.1436)$

$M_B = 55,95$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 8 cm

$M_A \approx 1:350.000$

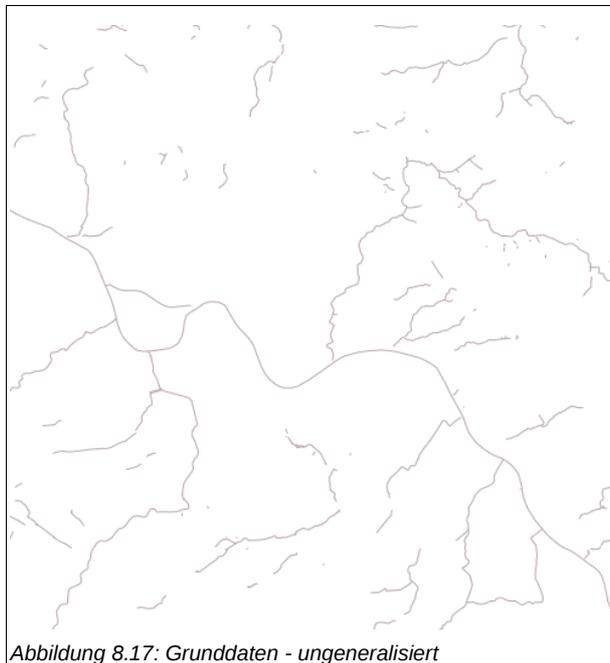


Abbildung 8.17: Grunddaten - ungeneralisiert



Abbildung 8.18: Generalisierte Daten - Toleranzwert 50

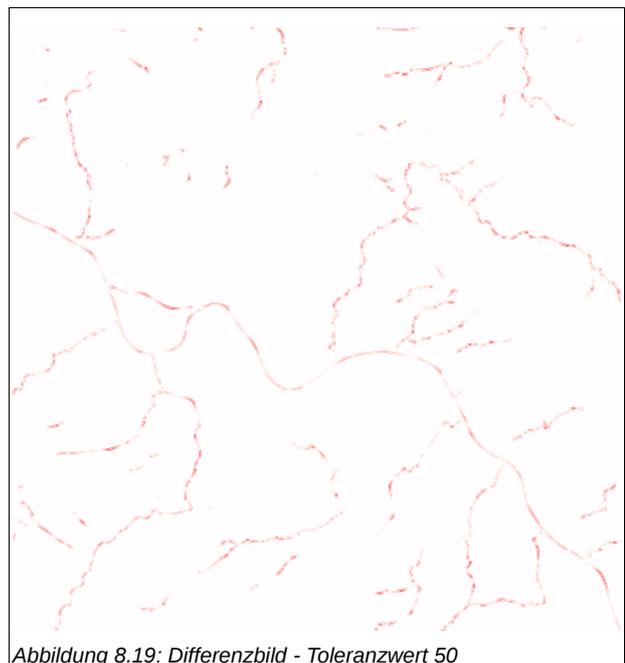


Abbildung 8.19: Differenzbild - Toleranzwert 50



Abbildung 8.20: Generalisierte Daten - Toleranzwert 100

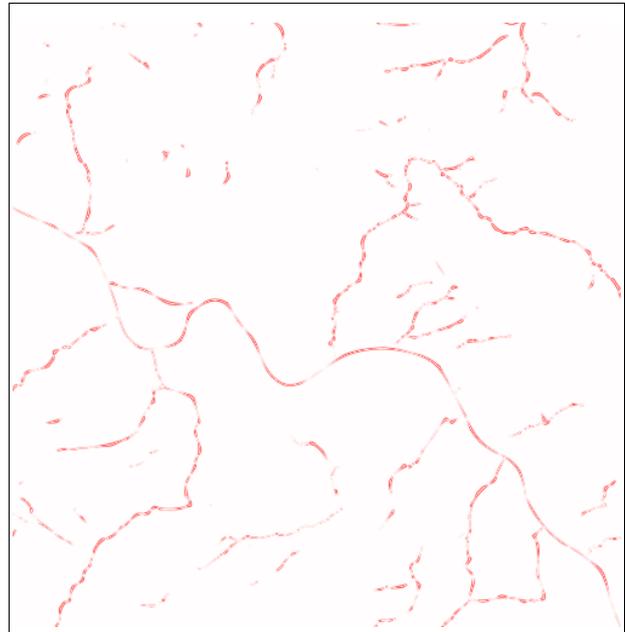


Abbildung 8.21: Differenzbild - Toleranzwert 100



Abbildung 8.22: Generalisierte Daten - Toleranzwert 200

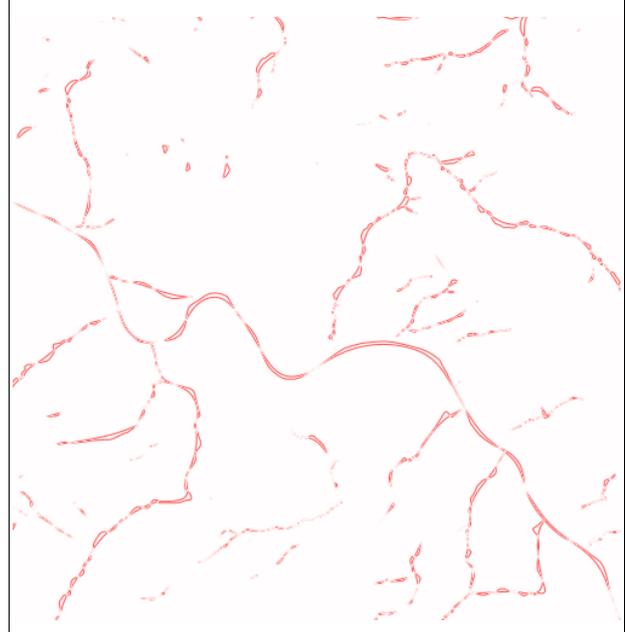


Abbildung 8.23: Differenzbild - Toleranzwert 200



Abbildung 8.24: Generalisierte Daten - Toleranzwert 500

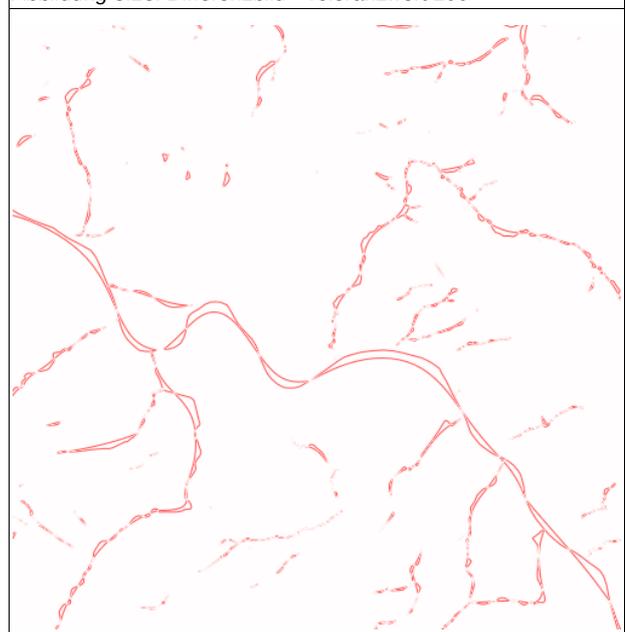


Abbildung 8.25: Differenzbild - Toleranzwert 500

Anhang H - Auswahl nach Mindestgrößen mittels PostGIS-Funktion „ST_Length“

Abbildungen: 8.26, 8.27, 8.28, 8.29, 8.30, 8.31, 8.32

$B_{\text{box}} = (13.381, 50.845, 14.139, 51.271)$

$M_B = 168,76$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 8 cm

$M_A \approx 1:1 \text{ Mio.}$

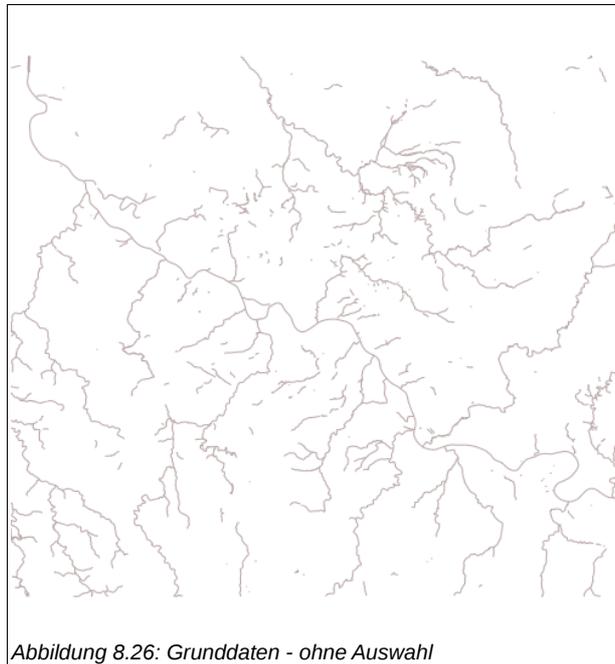


Abbildung 8.26: Grunddaten - ohne Auswahl

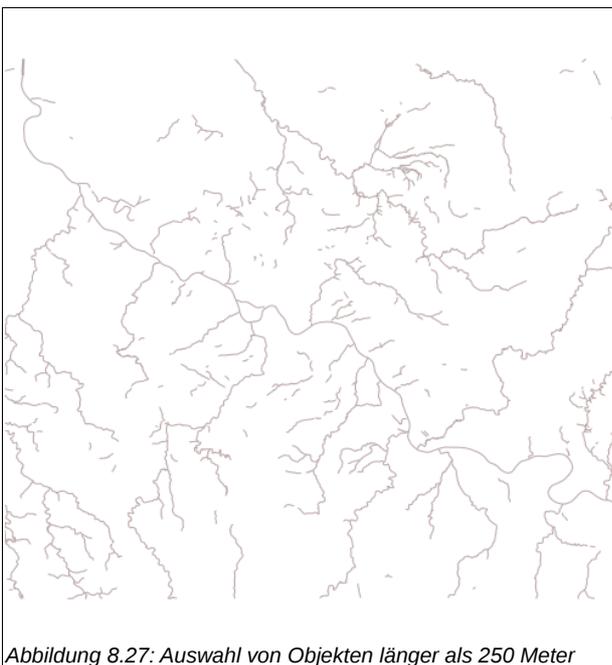


Abbildung 8.27: Auswahl von Objekten länger als 250 Meter

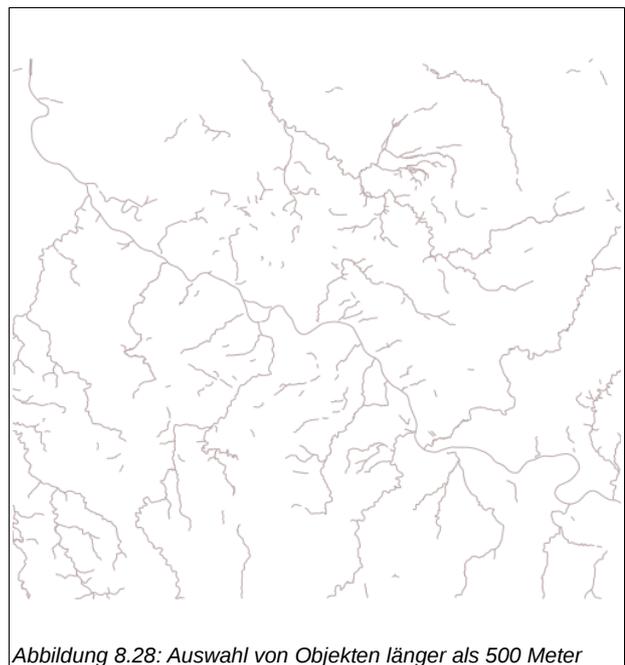


Abbildung 8.28: Auswahl von Objekten länger als 500 Meter

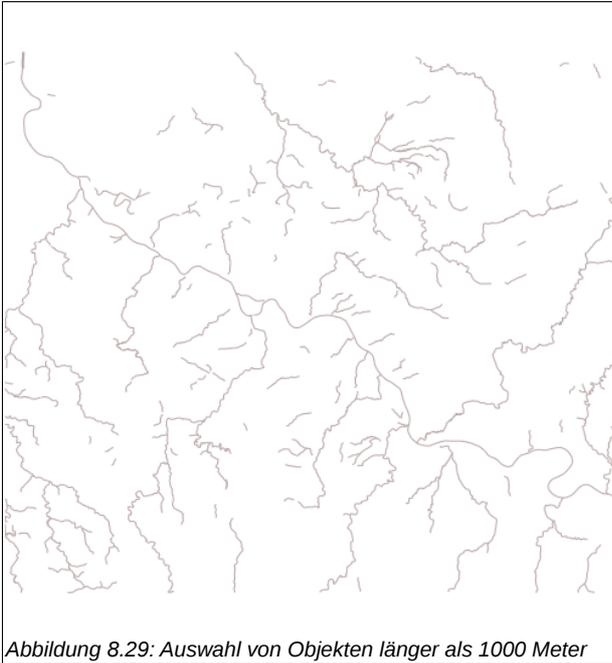


Abbildung 8.29: Auswahl von Objekten länger als 1000 Meter

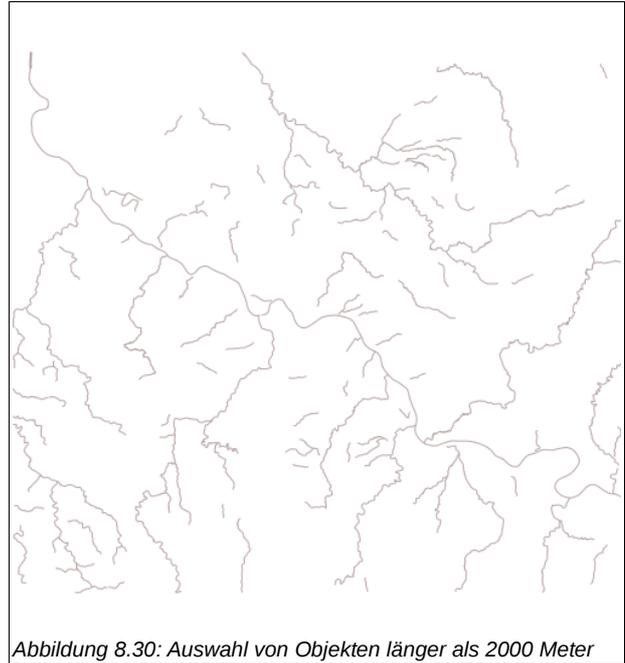


Abbildung 8.30: Auswahl von Objekten länger als 2000 Meter

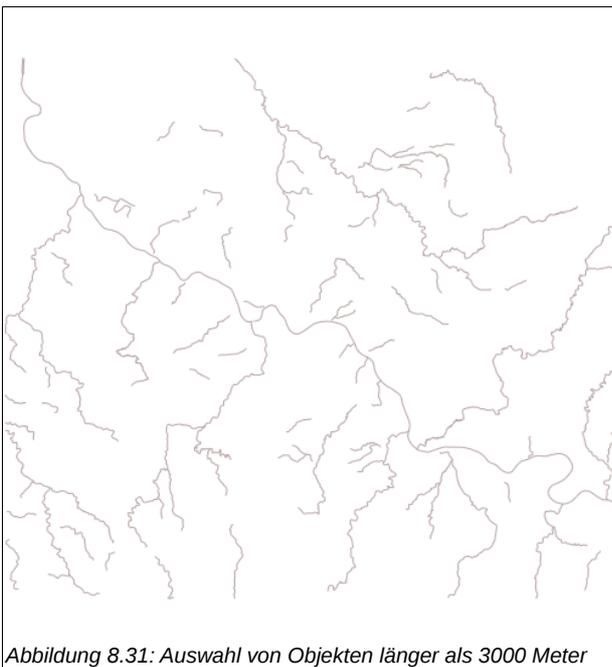


Abbildung 8.31: Auswahl von Objekten länger als 3000 Meter

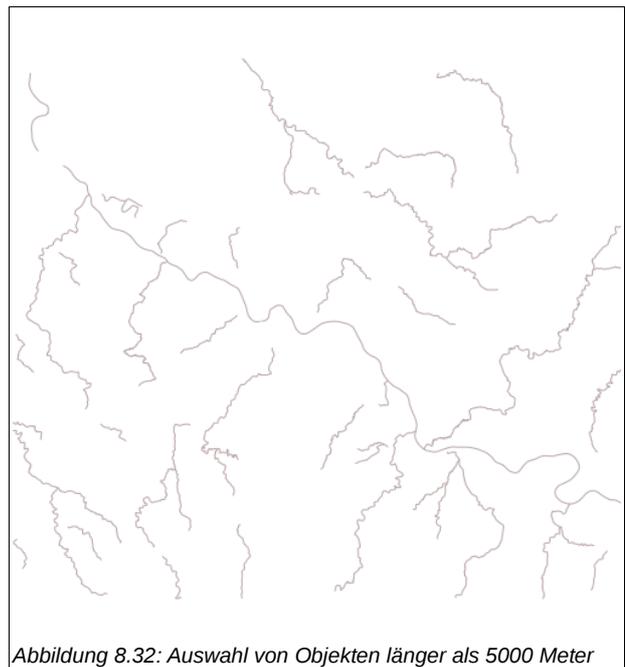


Abbildung 8.32: Auswahl von Objekten länger als 5000 Meter

Anhang I - Liniengeneralisierung mittels PostGIS-Funktion „ST_Simplify“

Abbildungen: 8.33, 8.34, 8.35, 8.36, 8.37, 8.38, 8.39, 8.40, 8.41

$B_{\text{box}} = (13.6291, 50.9965, 13.8804, 51.1436)$

$M_B = 55,95$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 8 cm

$M_A \approx 1:350.000$

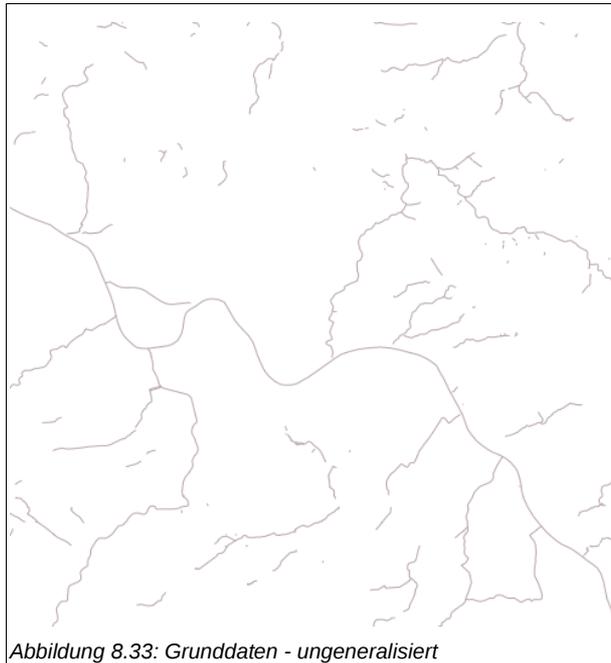


Abbildung 8.33: Grunddaten - ungeneralisiert



Abbildung 8.34: Generalisierte Daten - Toleranzwert 50



Abbildung 8.35: Differenzbild - Toleranzwert 50



Abbildung 8.36: Generalisierte Daten - Toleranzwert 100

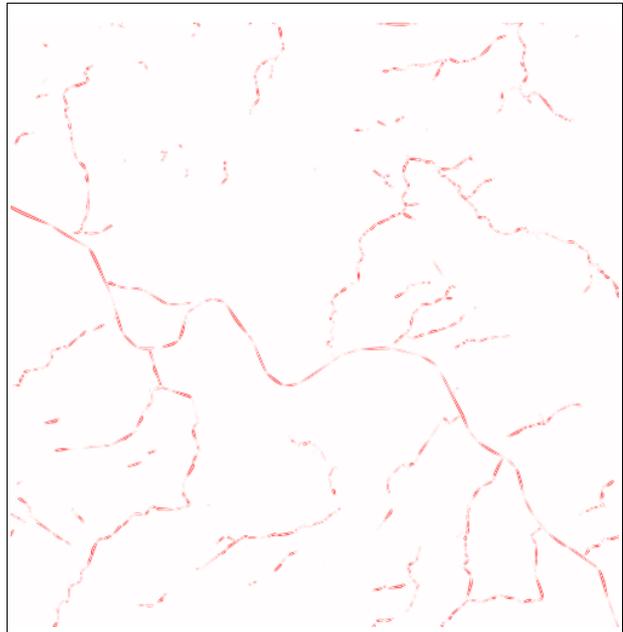


Abbildung 8.37: Differenzbild - Toleranzwert 100



Abbildung 8.38: Generalisierte Daten - Toleranzwert 200

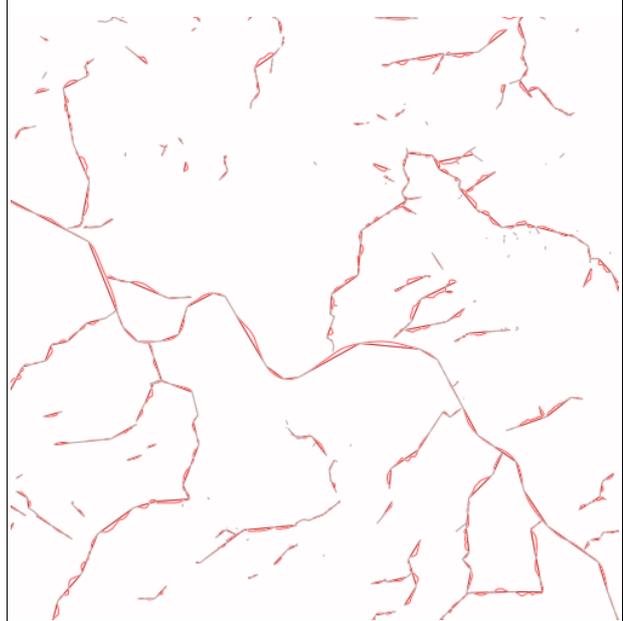


Abbildung 8.39: Differenzbild - Toleranzwert 200



Abbildung 8.40: Generalisierte Daten - Toleranzwert 500

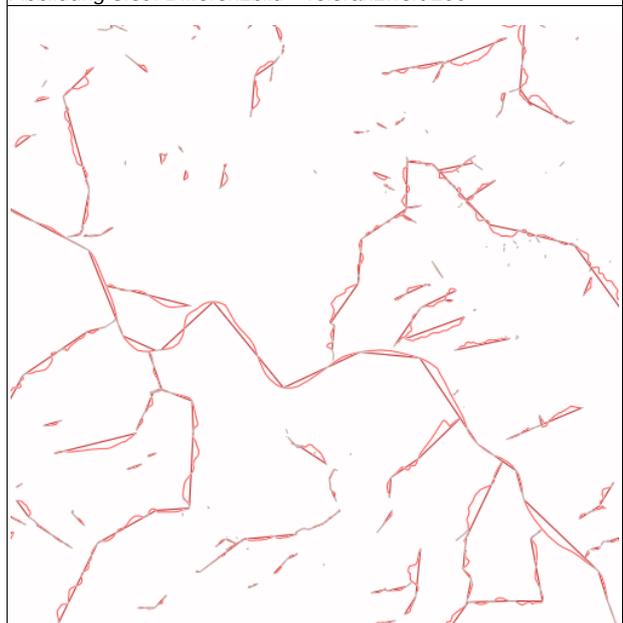


Abbildung 8.41: Differenzbild - Toleranzwert 500

Anhang J - Vergleich „ST_Simplify“ vs. „ST_SimplifyPreserveTopology“

Vergleich von Resultaten der Ableitung vereinfachter Gebäudegeometrien mittels PostGIS-Funktionen „ST_Simplify“ und „ST_SimplifyPreserveTopology“

Abbildungen: 8.42, 8.43, 8.44, 8.45

$B_{\text{box}} = (13.70772, 51.0189, 13.73914, 51.03825)$

$M_B = 7,00$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 8 cm

$M_A \approx 1:43.000$

Abbildungen: 8.46, 8.47, 8.48

$B_{\text{box}} = (13.72088, 51.02682, 13.72873, 51.03166)$

$M_B = 1,75$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 8 cm

$M_A \approx 1:10.000$



Abbildung 8.42: Grunddaten – ungeneralisiert

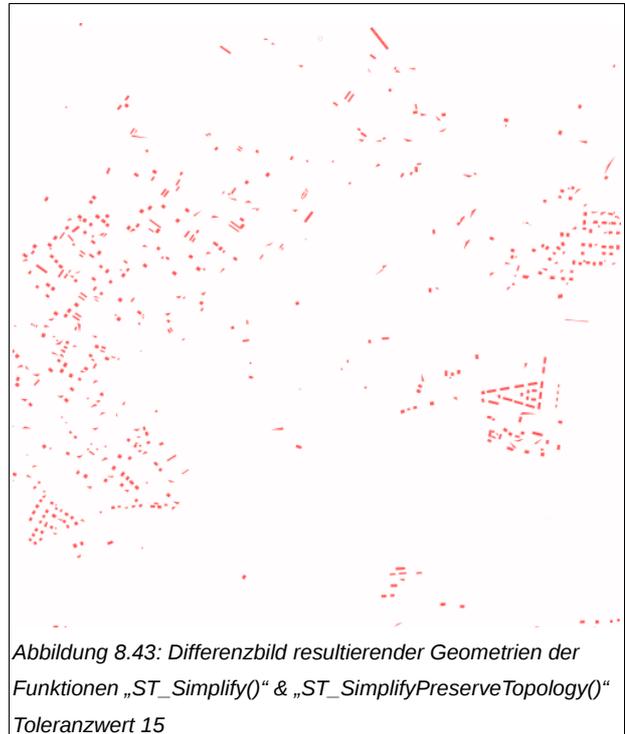


Abbildung 8.43: Differenzbild resultierender Geometrien der Funktionen „ST_Simplify()“ & „ST_SimplifyPreserveTopology()“
Toleranzwert 15





Abbildung 8.46: Grunddaten - ungeneralisiert



Abbildung 8.47: Generalisierte Daten mittels „ST_Simplify“ - Toleranzwert 10



Abbildung 8.48: Generalisierte Daten mittels „ST_SimplifyPreserveTopology“ - Toleranzwert 10

Anhang K - Gebäudegeneralisierung mittels PostGIS

Anwendung der PostGIS-Funktion „ST_SimplifyPreserveTopology“

Abbildungen: 8.49, 8.50

$B_{\text{box}} = (13.70772, 51.0189, 13.73914, 51.03825)$

$M_B = 7,00$

Bildgröße (Bildschirm) = 500 x 500 Pixel

Bildgröße (Druck) = 14 cm

$M_A \approx 1:25.000$



Abbildung 8.49: Reine, unbearbeitete Gebäudegeometrien



Abbildung 8.50: Generalisierte Gebäudegeometrien - Toleranzwert 10

Anhang L - CD mit Quellcodes und Abbildungen

Die beiliegende CD enthält eine digitale Fassung der vorliegenden Arbeit, sowie alle verwendeten Abbildungen.

- /Diplomarbeit.pdf
- /Abbildungen

Zusätzlich sind darauf die verwendeten Quellcodes in digitaler Form zu finden.

- /Anhang/Anhang A
- /Anhang/Anhang B
- /Anhang/Anhang C

Darüber hinaus sind innerhalb des Ordners „Anhang A“ sämtliche Style-Dateien hinterlegt, die für die erläuterten Implementierungen verwendet wurden.

- /Anhang/Anhang A/Style-Dateien

Des Weiteren sind alle erstellten Visualisierungen als digitale Bilddateien auf der beiliegenden CD vorhanden.

- /Anhang/Anhang D - Ausschnitte normal 1:25000
- /Anhang/Anhang E - Ausschnitte nur Gebäude 1:25000
- /Anhang/Anhang F - verkleinerter Ausschnitt 1:10000/Ausschnitte normal
- /Anhang/Anhang F - verkleinerter Ausschnitt 1:10000/Ausschnitte nur Gebäude
- /Anhang/Anhang G - nur Flüsse WPS
- /Anhang/Anhang G - nur Flüsse WPS/Karten
- /Anhang/Anhang H - Selection
- /Anhang/Anhang I - Simplify
- /Anhang/Anhang J - Simplify vs. SimplifyPreservetopology
- /Anhang/Anhang K - SimplifyPreserveTopology_Buildings

Muster

Hiermit erkläre ich, dass ich die von mir am heutigen Tage der Diplomkommission der Fachrichtung Geowissenschaften eingereichte Diplomarbeit zum Thema

.....
.....

vollkommen selbständig und nur unter Benutzung der in der Arbeit angegebenen Literatur angefertigt habe.

Dresden, den

.....
Unterschrift