Master Thesis

im Rahmen des Universitätslehrganges
„Geographical Information Science & Systems“
(UNIGIS MSc) am Zentrum für GeoInformatik (Z_GIS)
der Paris Lodron-Universität Salzburg

zum Thema

AN APPROACH TO GEOCODING BASED ON VOLUNTEERED SPATIAL DATA

vorgelegt von

Dipl. Wirt.-Inf. (FH) Christof Amelunxen
U1336, UNIGIS MSc Jahrgang 2007

Zur Erlangung des Grades
"Master of Science (Geographical Information Science & Systems) – MSc (GIS)"

Gutachter:

Ao. Univ. Prof. Dr. Josef Strobl

Paderborn, 19. Juni 2009

Acknowledgements

First of all I would like to thank Prof. Dr. Alexander Zipf, head of the Research Group Cartography, University of Bonn, and Pascal Neis, head developer of OpenRouteService and member of the research group, for giving me the opportunity to work in this interesting field of research. Their constructive feedback and constant support was of invaluable importance to the success of the work presented with this thesis.

Next I would like to thank Prof. Dr. Strobl, head of the Centre for Geomatics (Z_GIS), University of Salzburg, and all members of the UNIGIS team for their support during the time of the masters program of which the completion of this thesis represents the final step.

Last but not least I would like to thank my family and friends, who had to tolerate two and a half years of permant shortage of my free time, reaching its climax with the work on this thesis.

Statement of Originality

I hereby certify that the content of this thesis is the result of my own work. This thesis has not been submitted for any degree or other purposes, neither in whole nor in part. To the best of my knowledge and belief, it contains no ideas, techniques, quotations or any other material from the work of other people unless acknowledged in accordance with standard referencing practices.

Paderborn, June 19, 2009

Christof Amelunxen

Abstract

Background: The automated process of assigning geographic coordinates to textual descriptions of a place, generally referred to as *geocoding*, plays an important role in various fields of geographic information technologies. It is a fundamental requirement for spatial analysis of address datasets and has e.g. been used for analyses of health records or crime incidents for a long time already. The recent progress of location based services like route planning applications has further augmented the demand for geocoding services. However, the process of geocoding relies on the availability of a reference dataset against which addresses can be matched, but the collection and maintenance of appropriate spatial data is the traditional domain of official surveying offices or commercial companies. Hence there are only very few publicly available geocoding services which can be used free of charge, and those which exist are usually limited to a specific country or even smaller units. Furthermore, no freely available geocoding service offering house number-level precision has yet been implemented based on volunteered geographic data. The objective of this thesis was thus to explore the suitability of freely available spatial data, collected through collaborative volunteered effort, for its use as a reference dataset for a geocoding service.

Approach: The *OpenStreetMap* project has been selected as the data source for this research as it provides an impressively extensive database originating from collaborative volunteered effort and the exponential growth of the project data since its start in 2004 is very promising. The first task of the work presented was thus to analyze the data provided by the project and to develop an appropriate process to transform the data in a format usable for geocoding purposes. The next task has then been the actual design and implementation of the geocoding application. The geocoder has been integrated into the *OpenRouteService* project, providing a framework compliant to the *OpenGIS Location Service (OpenLS)* specifications. A major objective of the work was further to evaluate the possibilities to compensate incomplete data (namely house number positions) by probability based approaches to locate house number positions, supported by official house numbering guidelines for the study area of *Northrhine-Westfalia, Germany*.

Results: The success of the implementation has been evaluated using the standard geocoding quality key figures *match rate* and *positional accuracy*. The match rate, defined as the percentage of requests returning a correct match, has been found to be 86% on municipal level requests (sample size $n = 334$), 60% on street level requests ($n = 1000$) and 1% ($n = 1000$) on house number level requests for randomly chosen addresses within the study area. The average positional error for house number level requests (determined by comparing the results to the real positions of the buildings as provided by the surveying office for the study area) has been found to be 175 meters, with a 90th percentile of 370m. The positional accuracy nevertheless strongly depends on the source data, as whenever exact house number positions were available in the OpenStreetMap data, the average positional error was reduced to merely 13 meters. A comparison with the accuracy provided by the geocoding service offered by *Google*^{TM} showed that whenever house number data was available, the positional error was significantly lower than Google's and about equal when interpolation between two known house numbers was possible. Additionally, the probability based approaches, developed to approximate unknown house number positions, were indeed found to be capable of significantly improving geocoding accuracy.

Conclusion: The findings of the conducted research are able to serve as a proof of concept for the usage of volunteered spatial data as a reference dataset for geocoding services. The inherent inconsistencies present in the OpenStreetMap data however required substantial concessions in terms of referential integrity and the positional accuracy to be expected is strongly dependend on the availability of house number data, although means to partially compensate incomplete data have been successfully developed. The result of this work is already used in production as the geocoding engine for various research projects, of which *OpenRouteService*[1] and *OSM-3D*[2] presumably are the most prominent.

[1]available at: http://www.openrouteservice.org/
[2]available at: http://www.osm-3d.org/

Contents

List of Figures

List of Tables

List of Listings

Glossary

AJP	**A**pache **J**ava **P**rotocol
API	**A**pplication **P**rogramming **I**nterface
DBMS	**D**ata**b**ase **M**anagement **S**ystem
DML	**D**ata **M**anipulation **L**anguage
ERD	**E**ntity **R**elationship **D**iagram
GIS	**G**eographical **I**nformation **S**ystem
GPS	**G**lobal **P**ositioning **S**ystem
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol
LBS	**L**ocation **B**ased **S**ervices
MLP	**M**obile **L**ocation **P**rotocol
OGC	**O**pen **G**eospatial **C**onsortium
ORS	**O**pen**R**oute**S**ervice
OSM	**O**pen**S**treet**M**ap
RDBMS	**R**elational **D**atabase **M**anagement **S**ystem
SQL	**S**tructured **Q**uery **L**anguage
TIGER	US census **T**opographically **I**ntegrated **G**eographic **E**ncoding and **R**eference System
URL	**U**niform **R**esource **L**ocator
VGI	**V**olunteered **G**eographic **I**nformation
WFS	**W**eb **F**eature **S**ervice
WMS	**W**eb **M**ap **S**ervice
XML	e**X**tensible **M**arkup **L**anguage

1 Introduction

"Volunteered Geographic Information (VGI) has the potential to be a significant source of geographers' understanding of the surface of the Earth."

Goodchild (2007*a*, p.14)

1.1 Motivation

Goodchild (2007*a*) proposed the term *Volunteered Geographic Information (VGI)* for a phenomena which has significantly altered the world of geographic information science: geographic information generated by collaborative volunteered effort.

Until recently, the generation, maintenance and distribution of geographic information has been solely the domain of either official land surveying offices or commercial companies, but, with only very few exceptions, not of volunteers. This was presumably mainly due to the immense costs related to the actual surveying and maintenance and the lack of possibilities to effectively share and distribute the collected spatial data.

However, this has recently changed, for the two following reasons[3]:

1. The dramatically reduced costs along with the enhanced usability of modern satellite navigation handheld devices have enabled a mass of people to collect geographic data with ease of use and in precision levels which had formerly been simply beyond reach for private persons.
2. The progress of the *internet* from a formerly "read-only media" to the "web 2.0" participatory approach has made collaborative efforts to generate and share content of various kinds very common.

[3]based on suggestions by Goodchild (2007*b*)

Among a broad list of projects dealing with user generated geographic information, *OpenStreetMap* is one of the most promising. Its primary goal is to generate a free map of the world (OpenStreetMap, 2009) through volunteered effort. Nevertheless, although the generation of maps still is the focus of the project, the collected spatial data is made publicly available and may be used for other purposes as well. *OpenRouteService*[4] e.g. is an example of a project which has successfully implemented a routing service based on OpenStreetMap data.

The focus of *this* work however is to explore the suitability of OpenStreetMap data for the purpose of *geocoding*, simplified as the conversion of textual address information into point coordinates and vice versa[5]. *Geocoding* forms an essential requirement of various spatial applications[6], yet there are only very few publicly available geocoding services which can be used free of charge, because they mostly depend on proprietary spatial data and besides are usually limited to a specific country or even smaller units, too. Furthermore, no freely available geocoding service offering house number-level precision has ever been implemented based on volunteered geographic data.

If a working geocoding service could successfully be built based on OpenStreetMap data, this would be a substantial advance in the improvement and progression of a wide range of projects, based in the field of volunteered geographic information.

1.2 Objectives

The main objective of this thesis is to design, implement and evaluate a geocoding service based on volunteered spatial data. This includes the following challenges:

- Analysis of the base data concerning its suitability for geocoding purposes.
- Design and development of data models and transformation algorithms.
- Design and implementation of the geocoding process.
- Development of means to compensate incomplete base data using probability based approximation approaches.
- Evaluation of the geocoder in terms of accuracy and completeness.

[4]http://www.openrouteservice.org/
[5]a detailed definition of the term geocoding will be given in chapter 2
[6]see section 2.2 for examples

1.3 Scope

Although the methodological part of this work is supposed to generate generally valid methods and concepts, the study area for the implementational part will be focused on the federal territory of Germany. The methods and concepts presented may nevertheless just as well be adopted by subsequent research projects investigating different study areas.

1.4 Approach and Methodology

The approach and methodology chosen to accomplish the objectives stated can be described as follows:

At first, the general suitability of the OpenStreetMap data for geocoding purposes will be evaluated with respect to its data model, relational integrity and completeness. Based on this analysis the proposed data model for the geocoder's reference dataset will be designed and an appropriate data transformation and integration process will be developed following the concepts presented by Han and Kamber (2006) and Rahm and Do (2000).

This will be followed by the definition and analysis of use cases to be provided by the geocoding service. The actual processing of the geocoding use cases will be designed following standard geocoding practices as described by Goldberg (2008), Davis et al. (2003), Borkar et al. (2001) and Christen and Churches (2005).

The treatment of incomplete house number data will receive special attention. In order to compensate missing house number data in OpenStreetMap, different probability based approaches will be developed in order to effectively approximate house number locations. This includes the analysis of house numbering systems in general and research concerning habits and regulations for house number assignment within the study area in order to construct and evaluate hypotheses for the approximation of house number locations. This part will mainly be based on the work of Goldberg (2008), Ratcliffe (2001) and Bakshi et al. (2004).

The quality of the geocoder, implemented according to the concepts and guidelines developed before, will finally be measured using the standard key figures *match rate* and *positional accuracy* as described by Cayo and Talbot (2003) and additionally by comparing the positional accuracy measured to a commercial geocoding service provided by *Google^{TM}*.

1.5 Expected Results

The work presented should give answers to the following questions:

1. Is it possible to build a working geocoding service based on the volunteered spatial data provided by the OpenStreetMap project?
2. Is it possible to effectively compensate incomplete spatial data (particularly house number locations) using probability based approaches?
3. Which completeness and accuracy level can be achieved by a geocoder based on volunteered spatial data and how do these figures compare to commercial geocoding services?

1.6 Target Audience

The target audience addressed with this thesis is everybody interested in…

- the general potential of volunteered geographic information
- the specific potential of the OpenStreetMap project as a basis for geocoding purposes
- general solutions to the challenges and tasks faced when implementing geocoding services
- probability based approaches to approximate house number positions
- the research field of geocoding in general

1.7 Structure of the Thesis

The thesis is divided into the following chapters:

Geocoding Basics	Introducing the term geocoding and giving an overview of common use cases for geocoding services.

Research Basis	Providing an overview of the basic technologies and fields of research which the work presented in this thesis is based on.
Analysis and Methodology	Analysing tasks and challenges involved in implementing the geocoder and developing methodological approaches to solve them in order to provide a conceptual guideline for the implementation phase.
Implementation	Describing the implementation of the geocoder, based on the fundamental concepts and technologies presented and according to the guidelines developed in the analysis phase.
Evaluation	Measuring the success of the implementation using standard geocoding quality key figures and by comparison against a commercial geocoding service.
Summary	Summarizing the conclusions to be drawn from the results of the work presented in this thesis and proposing an outlook for further research.

2 Geocoding Basics

This chapter provides a definition of the term geocoding and gives an overview of common use cases for geocoding services.

2.1 What is Geocoding?

People have always been used to associate real world objects with geographical places by using verbal descriptions of locations. These descriptions are either expressed as a spatial relation to some other, already known location ("the Center for Geoinformatics at Salzburg University is located south of the city center") or defined by a postal address referencing an entity of a publicly known street address database ("the Center for Geoinformatics at Salzburg University is located at Hellbrunnerstraße 34").

Verbal geographical references are well understood and interpreted by people but are useless for any type of spatial analyses, which require the object's coordinates in a spatial reference system instead (i.e. it needs to be *georeferenced*). Furthermore, verbal descriptions of a place are not deterministic and prown to errors, require secondary knowledge to be interpreted correctly and can change over time as pointed out by Wiezoreck et al. (2004).

The process of transforming a textual description of a place to coordinates in a given spatial reference system is generally called "geocoding", although the definition and usage of the term varies in scientific literature. Some authors limit the scope of input data to postal addresses (Bakshi et al., 2004; Behr et al., 2008; Cayo and Talbot, 2003), whereas others widen the scope to include named places (Davis et al., 2003) or even arbitrary textual representations of a place (Pouliquen et al., 2004; Goldberg, 2008).

The output of the geocoding process can have multiple formats, too. In most cases the output is a coordinate pair (Zandbergen, 2007*b*; Ratcliffe, 2004; Davis et al.,

2003) but it may as well be of a more complex shape like lines or polygons, depending on the type of object to be georeferenced and the intended use (Wiezoreck et al., 2004).

Considering the ambiguous use in literature, Goldberg (2008, p.5) arguably gives the most generic definition of the term *geocoding* by describing it as being. . .

> "the act of transforming aspatial locationally descriptive text into a valid spatial representation using a predefined process"

Additionally, the term **reverse geocoding** is commonly used to describe the opposing process of transforming a coordinate pair into a locally descriptive text (Brownstein et al., 2005).

The service which actually *does* the geocoding is called a *geocoder*. The geocoder requires a *reference dataset*[7] containing known geographic features to determine the geographic reference matching the textual reference of the place. The comprehensiveness and accuracy of this reference dataset is one of the most important factors determining the overall quality of the geocoding results (e.g. shown by Ratcliffe (2001), Grubesic and Murray (2004) and Waldner et al. (2005)).

2.2 The Use of Geocoding

A literature research showed that the concept of geocoding as an automated process of transforming textual address data into geographical coordinates, in order to spatially analyze the collected data, has already been addressed in 1970 by the U.S. Census Use Study introducing the *DIME Geocoding System* (Farnsworth, 1970).

Among the first uses of geocoding has been the spatial analysis of health records, e.g. analyzing cancer registry data (Rushton et al., 2006, p.16) in order to detect spatial patterns in cancer distributions. The analysis of health data is still one of the main uses of geocoding (Cayo and Talbot, 2003; Krieger et al., 2005) and a great deal of the scientific literature addressing the field of geocoding has been published in the context of health science.

Another field where geocoding has been used for a long time already with increasing importance is crime analysis, as pointed out by Harris (1999) and

[7]also known as "geographic base file" (Grubesic and Murray, 2004)

Ratcliffe (2004). Harris (1999, p.98) states that "Geocoding is vitally important for crime mapping since it is the most commonly used way of getting crime or crime-related data into a GIS[8]".

Other examples of geocoding usage scenarios include the spatial analysis of customer databases, urban planning and development, market research or emergency services.

Nevertheless, in the last couple of years arguably the most driving factor in the field of geocoding has been the constant advance of *Location Based Services (LBS)* like online route planning services e.g. This has lead to a significant increase in the amount of services requiring a geocoding component serving as an interface between the user, who is used to verbal descriptions of a place, and the spatial application, which needs geographically valid references instead.

[8]GIS: Geographical Information System

3 Research Basis

This chapter provides an overview of the basic technologies and fields of research which the work presented in this thesis is based on.

3.1 OpenStreetMap

3.1.1 Project Description

The OpenStreetMap project (OSM) was founded in August 2004 at the University College London (UCL) by Steve Coast as a collaborative effort of volunteers, who contribute spatial data to a common database. The following quote from the OpenStreetMap project website[9] provides a summary of the project's intention (OpenStreetMap, 2009):

> "OpenStreetMap is a project aimed squarely at creating and providing free geographic data such as street maps to anyone who wants them. The project was started because most maps you think of as free actually have legal or technical restrictions on their use, holding back people from using them in creative, productive or unexpected ways."

The main difference to other publicly available online mapping services like Google Maps™[10] is that both the collected spatial data and the maps derivated from this data are free to use for everybody. Furthermore, everybody can use the spatial data to build their own maps from it and use those maps for their own projects, websites, illustrations etc. Google Maps™ has severe restrictions on the usage of the service instead (see Google Maps (2009)) and the raw spatial data

[9] http://www.openstreetmap.org/
[10] http://maps.google.com/

itself is not available to the public at all because it is the property of commercial mapping companies like *Navteq*[11] or *TeleAtlas*[12] as clearly stated in the terms and conditions of Google Maps (2009). Hence it is also impossible to build own elaborate services like route planners or geocoders upon these services.

OpenStreetMap follows the "wiki-concept" instead, which most people are familiar with due to the success of the online encyclopedia Wikipedia[13]. The concept basically means that every user is allowed to add, modify and delete content as he or she likes. All users together serve as a regulatory instance in this system as they are able to review and correctify or even undo the operations committed by others.

The spatial data itself is gathered from different sources but mainly contributed by users tracking points using GPS[14] handheld devices and later uploading the collected data to the OpenStreetMap database. Once uploaded the data can be edited and labeled via a set of specially developed editing tools. Figure 3.1 visualizes the single steps of the whole process.

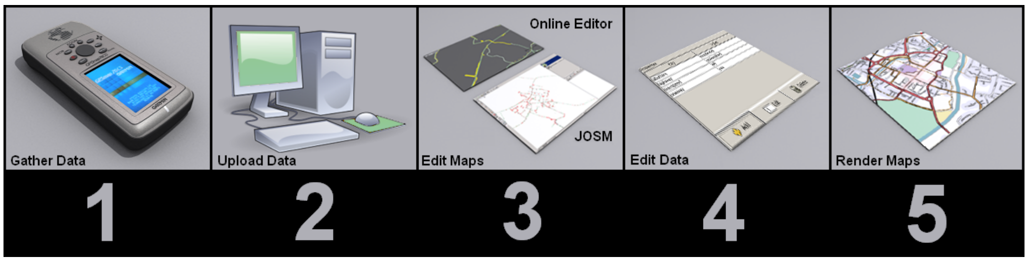


Figure 3.1: The five steps to making a map
Source: OpenStreetMap Beginner's Guide:
http://wiki.openstreetmap.org/wiki/Beginners'_Guide
last accessed: March 22nd, 2009

Most of the mapping is done by single persons, collecting the data on their own, but in 2006 a social event called "mapping party" had been introduced, where a group of people met to collaboratively map a predefined area. Perkins and Dodge (2008) have shown the potentials and weaknesses of this approach by joining a mapping party held in Manchester, UK in May 2006 and analyzing the results afterwards.

[11]http://www.navteq.com/
[12]http://www.teleatlas.com
[13]http://www.wikipedia.de/
[14]GPS: Global Positioning System

Other sources of spatial data include public-domain datasets like the *TIGER* street data offered by the US Census Bureau, which is converted as needed and integrated into the OpenStreetMap database, as well as available copyright-less or donated satellite images[15] and maps.

3.1.2 Software Components

The OpenStreetMap project consists of five main software components:

Database	used to store the data objects.
Interfaces	provide access to the database's contents.
Editors	are used by the project members to upload and edit data.
Renderers	generate maps from the database's contents.
Frontends	provide access to the maps generated by the renderers.

The **database** system used to store the data is a *MySQL*[16] relational database management system[17]. Several **interfaces** provide access to this database, which will be discussed in section 3.1.4. The data itself is uploaded and edited by the project members via a set of specially developed **editors**[18].

Renderers[19] are software components which generate map images using objects retrieved from the database and those maps are then presented to the public by appropriate **frontends**. For performance reasons the maps are normally pregenerated as images in different scales, called *tiles*, which are made available via HTTP[20] from a set of *tile servers*. Theses tiles are again retrieved and assembled to a complete map of the area which is to be shown by the frontends.

The main **frontend** is the project's website[21] which is not only used to present the maps but additionally offers an online editor[22] as well as user interaction features like registering new users or login and logout.

[15]Yahoo^{TM} e.g. granted OSM the right to use their satellite images
[16]http://www.mysql.com/
[17]status as of March 28th, 2009
[18]a list of available editors is available at http://wiki.openstreetmap.org/wiki/Editor (last accessed: March28th, 2009)
[19]a list of available renderers is available at http://wiki.openstreetmap.org/wiki/Rendering (last accessed: March28th, 2009)
[20]HTTP: Hypertext Transfer Protocol
[21]http://www.openstreetmap.org/
[22]the *Potlatch* editor, for a description see http://wiki.openstreetmap.org/wiki/Potlatch (last accessed: March28th, 2009

Figure 3.2 shows an overview of the project's components published on the OpenStreetMap website.

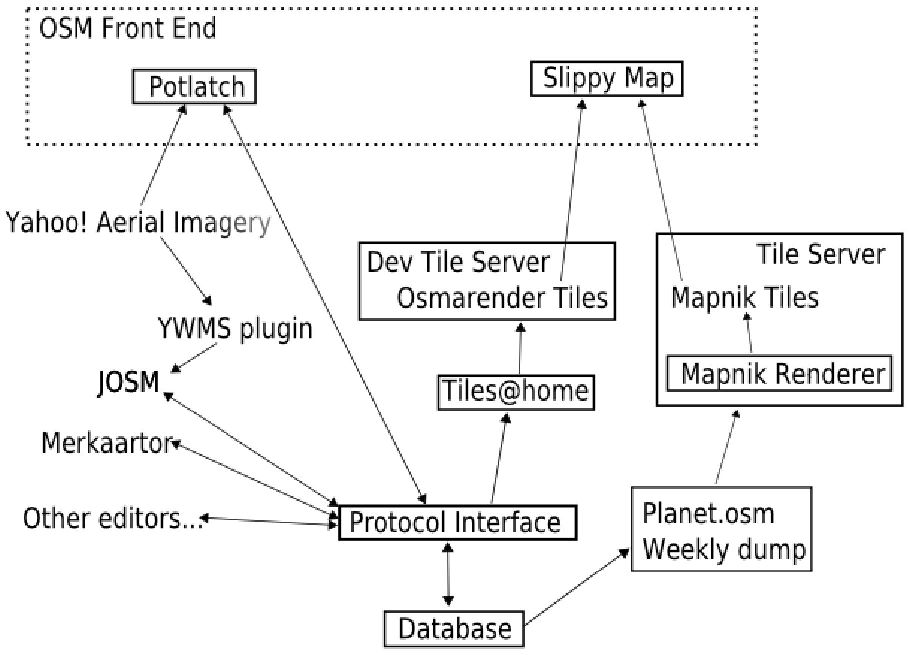


Figure 3.2: OpenStreetMap components overview
Source: http://wiki.openstreetmap.org/wiki/Image:OSM_Components.png
last accessed: March 28th, 2009

3.1.3 Data Model

The data model of OpenStreetMap is kept very simple. It consists of only three different object classes called *data primitives* (OpenStreetMap, 2009):

Node	A point feature represented by a latitude/longitude coordinate pair.
Way	A line feature represented by two or more connected **nodes**.
Relation	A set of primitives which are grouped together to form an abstract data structure.

Figure 3.3 on the following page gives an overview of the data primitives and their relations as defined in the OpenStreetMap data model.

Each of the data primitives is given a set of attributes. The following attributes are used by each of the three primitives to store internal data:

id	The internal identifier for an object, generated automatically when an object is created, and which must be unique within its object class (node, way or relation).
user	The user who contributed the object.
timestamp	Time of last modification.
visibility	If set to 'false' the object is logically deleted but still accessibly when specifically addressed.

In addition to these attributes the objects may contain a set of **tags**. These are key/value pairs describing "what the object is". There are no technical restrictions on the selection of tags added to a given object but there is an increasing list of well-known and recommended tags and values which the users are strongly encouraged (but not forced) to use[23]. The whole set of valid objects, their attributes and relations is defined in a Document Type Definition (DTD).[24]

[23]a description of the most commonly used tags and their intended meaning is available at http://wiki.openstreetmap.org/wiki/Map_Features (last accessed: March 25th, 2009)
[24]available at http://wiki.openstreetmap.org/index.php/OSM_Protocol_Version_0.5/DTD (last accessed: March 26th, 2009)

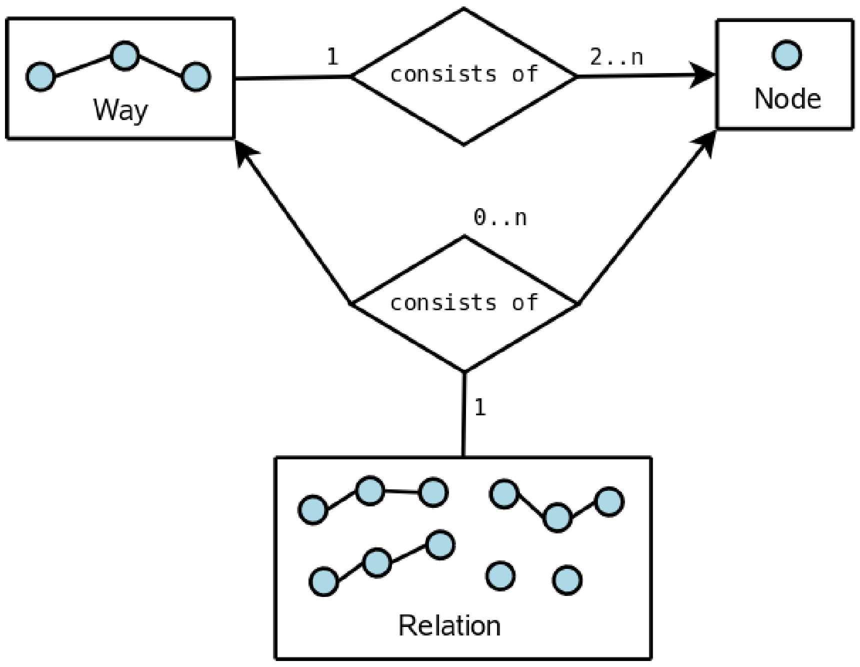


Figure 3.3: OpenStreetMap data primitives and their relations
Source: own assembly based on OpenStreetMap (2009)

At time of writing the OpenStreetMap database consists of approx. 325 million nodes, 26 million ways and 88,000 relations (OpenStreetMapStats, 2009).

3.1.3.1 Nodes

Nodes are the most basic data type in the OpenStreetMap database representing a single point on the earth's surface referenced by a lat/lon coordinate pair. Nodes are either used as vertices for line objects or have a meaning on their own, defined by appropriate tags. Listing 3.1 shows a point feature extracted from the OpenStreetMap database in XML[25] format which is defining the center of the city of Salzburg[26]. This point is given a meaning by its tags *place* and *name*.

```
<node id="34964314" lat="47.8001948" lon="13.0410636">
  <tag k="name" v="Salzburg"/>
  <tag k="place" v="city"/>
</node>
```

Listing 3.1: XML example of a point object in the OpenStreetMap database

3.1.3.2 Ways

A **way** is a collection of connected **nodes** forming a linestring. The most common use for this data type is the definition of street segments but it is as well used to store any other real world entity which can be represented as a line, like railways, rivers, borders, etc. Listing 3.2 on the following page shows an example of a segment of the street "Hellbrunner Straße" in Salzburg[27]. The vertices of the line segment are defined by nested <*nd*>-elements referencing the nodes storing the actual coordinates.

If a linestring forms a closed ring (i.e. a line of at least three nodes of which the first and the last are identical) the object is considered a polygon and thus representing an area. This is used to model landuse or buildings e.g. Listing 3.3 on the next page shows an example of a building object extracted from the OpenStreetMap database.

[25] XML: Extensible Markup Language
[26] irrelevant attributes removed for clarity
[27] irrelevant attributes removed for clarity

3.1.3.3 Relations

Relations define logical connections between two or more objects (nodes, ways or other relations) which share a common role. It is most commonly used to combine line segments (ways) in order to model a route object such as a special bicycle path but it may as well be used to combine a set of buildings which represent a logical unit, e.g. a university campus. The members of a relation are referenced by a list of <*member*>-elements nested inside. Listing 3.4 on the following page shows an example of a relation object extracted from the OpenStreetMap database, representing the route of a local train service and thus containing line segments as members.

```
<way id="23253405">
    <nd ref="20967534"/>
    <nd ref="245562416"/>
    <nd ref="245562417"/>
    <nd ref="248144729"/>
    <nd ref="276158449"/>
    <nd ref="245562418"/>
    <nd ref="245562419"/>
    <nd ref="245562420"/>
    <nd ref="251625231"/>
    <nd ref="251625234"/>
    <nd ref="251625235"/>
    <tag k="highway" v="residential"/>
    <tag k="name" v="Hellbrunner Straße"/>
</way>
```

Listing 3.2: XML example of a line object in the OpenStreetMap database

```
<way id="30514144">
  <nd ref="336827617"/>
  <nd ref="336827620"/>
  <nd ref="336827623"/>
  <nd ref="336827626"/>
  <nd ref="336827617"/>
  <tag k="name" v="Universität Salzburg Naturwissenschaftliche
     Fakultät"/>
  <tag k="area" v="yes"/>
  <tag k="building" v="yes"/>
</way>
```

Listing 3.3: XML example of a polygon object in the OpenStreetMap database

3.1.4 Data Interfaces

There are three different types of interfaces used to access and work with the data collected by the OpenStreetMap project.

- directly accessing the database
- using XML export files extracted from the database
- using the maps generated by the renderers

3.1.4.1 Direct Access

The OpenStreetMap database ist directly accessible via an API[28] built upon HTTP[29]. To access the interface, clients establish an HTTP connection to the server and then send regular HTTP commands (`GET`, `PUT`, `POST` and `DELETE`) to retrieve or modify database objects. The actual payload ("the data") is exchanged in XML format via HTTP request and response bodies. This whole process is defined in the *OSM Protocol*[30] which will not be discussed in detail here.

3.1.4.2 Database Exports

OpenStreetMap offers a weekly updated export of all objects stored in the database as a single file in XML format called *planet file*[31]. At time of writing

[28]API: Application Programming Interface
[29]the specification of HTTP is published in IETF: The Internet Engineering Task Force (1999)
[30]current version of the protocol is 0.5 (as of March 28th, 2009), see OSMProtocol (2009)
[31]available at: http://planet.openstreetmap.org/
(last accessed: March 27th, 2009)

```
<relation id="67041">
   <member type="way" ref="22954540" role=""/>
   <member type="way" ref="30039116" role=""/>
   <member type="way" ref="30103534" role=""/>
   <tag k="name" v="S1"/>
   <tag k="route" v="rail"/>
   <tag k="type" v="route"/>
   <tag k="operator" v="Salzburger Lokalbahn"/>
</relation>
```

Listing 3.4: XML example of a relation object in the OpenStreetMap database

this file contained more than 150 gigabytes of raw XML data. It is mainly used to import the data (either in whole or in part) into another database used by projects based on OpenStreetMap.

In addition to the *planet file* OpenStreetMap offers daily updated differential files containing only the updates made since a given point in time. These can be used to update derivated databases which had initially been created from a *planet file*.

Apart from that, some external organisations and companies offer local extracts from the *planet file*. The German company *Geofabrik* e.g. offers extracts down to first level administrative boundaries for the scope of Europe which can be downloaded from their website for free[32].

3.1.4.3 Rendered Maps

If the raw data itself is not needed, the content of the database can be accessed by retrieving the maps pregenerated by the renderers. The maps generated by the most commonly used *Mapnik renderer* e.g. are available via specifically constructed HTTP requests including a lat/lon coordinate pair and a zoom level[33]. Listing 3.5 shows an example URL used to retrieve a tile of the city center of Salzburg at zoom level 15[34].

```
http://tile.openstreetmap.org/15/17571/11417.png
```

Listing 3.5: Example URL used to retrieve a tile generated by mapnik renderer

3.1.5 Usage Examples

Apart from the OpenStreetMap website itself, which primarily offers an online mapping service, there is a constantly increasing number of projects incorporating components of the OpenStreetMap project for various kinds of applications

[32]available at: http://download.geofabrik.de/osm/
(last accessed: March 27th, 2009)
[33]Mapnik offers zoom levels ranging from 0 (whole world) to 18
[34]a detailed description of the contruction of appropriate URLs is available at: http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames
(last accessed: March 28th, 2009)

and services, ranging from merely including generated maps into their website to complex spatial applications like online route planning services.

An example for a service using OpenStreetMap data to generate special purpose maps is *OpenCyleMap*[35], providing maps specifically designed for bicyclists. Others even use the data to create printed maps, like a map of the city of Freiburg which has been generated to provide a handout for a conference[36].

Other projects use OpenStreetMap data to create elaborate spatial applications way beyond map making, like routing services (e.g. *OpenRouteService*[37] and *YourNavigation*[38]) or even 3D visualization services as developed by *OSM-3D Germany*[39].

3.1.6 Licensing

The data generated by the OpenStreetMap project is licensed under the terms of the *Creative Commons Attribution-Share Alike License*[40] as of version *2.0*. This license allows to freely *share, copy, distribute* and *transmit* the data as well as *remixing*[41] and using it for other projects under the following conditions (Creative Commons, 2009):

- the originator of the data has to be clearly named
- the redistribution of data derivated from the original data has to be licensed under terms conforming to this license
- the license itself has to be provided and notices of the license included in the original data must not be removed

[35]service available at: http://www.opencyclemap.org/
(last accessed: March 28th, 2009)
[36]available at: http://www.remote.org/frederik/tmp/freiburg-stadtplan.zip
(last accessed: March 29th, 2009)
[37]http://www.openrouteservice.org/
[38]http://www.yournavigation.org/
[39]http://www.osm-3d.org/
[40]available at: http://creativecommons.org/licenses/by-sa/2.0/
(last accessed: March 27th, 2009)
[41]modifying the data or transforming it into a different format

3.2 OpenGIS Location Services (OpenLS)

The *OpenGIS Location Services (OpenLS)* is a framework published by the *Open Geospatial Consortium (OGC)* proposing a standardized set of interfaces, protocols and data types for implementations of *Location Based Services (LBS)*. The main goal of this initiative, which was founded in 2000, is to improve interoperability between Location Based Services by providing developers with a common specification. The most current version at time of writing (1.2) has been published in September 2008 (OGC, 2008). Contributors to this standard include leading companies on the GIS market, e.g. *Autodesk*, *ESRI*[42], *MapInfo* and *Intergraph*, which demonstrates their interest in this standard.

3.2.1 Core Compontents

The framework defines five core components (OGC, 2008):

Part 1: Directory Service	Provides an online directory to find a specific place, product or service.
Part 2: Gateway Service	Provides an interface through which the position of a mobile device can be determined from the network provider following the *Mobile Location Protocol Specification* (MLP, 2001).
Part 3: Location Utility Service	Provides a geocoding and reverse geocoding service.
Part 4: Presentation Service	Rendering geographic information for display on a mobile terminal.
Part 5: Route Service	Providing route planning functions.

3.2.2 Location Utility Service

The *Location Utility Service* defines the geocoder component of the OpenLS specification and will thus be given a closer look as it will form the basis for the geocoder

[42]ESRI: Environmental Systems Research Institute, Inc
http://www.esri.com

to be implemented within the scope of this thesis. The specification defines the following main tasks of the component as (OGC, 2008, pp.19):

- to determine a geographic position given a place name, street address or postal code (geocoding)
- to return a complete, normalized description of the place (normalization)
- to determine a complete, normalized decription of a place oder address given a geographic position (reverse geocoding)

3.2.2.1 Use Cases

Figure 3.4 shows a use case diagram explaining the role of the *Location Utility Service* component in different kinds of spatial requests to a service incorporating the *OpenGIS Location Services* specifications.

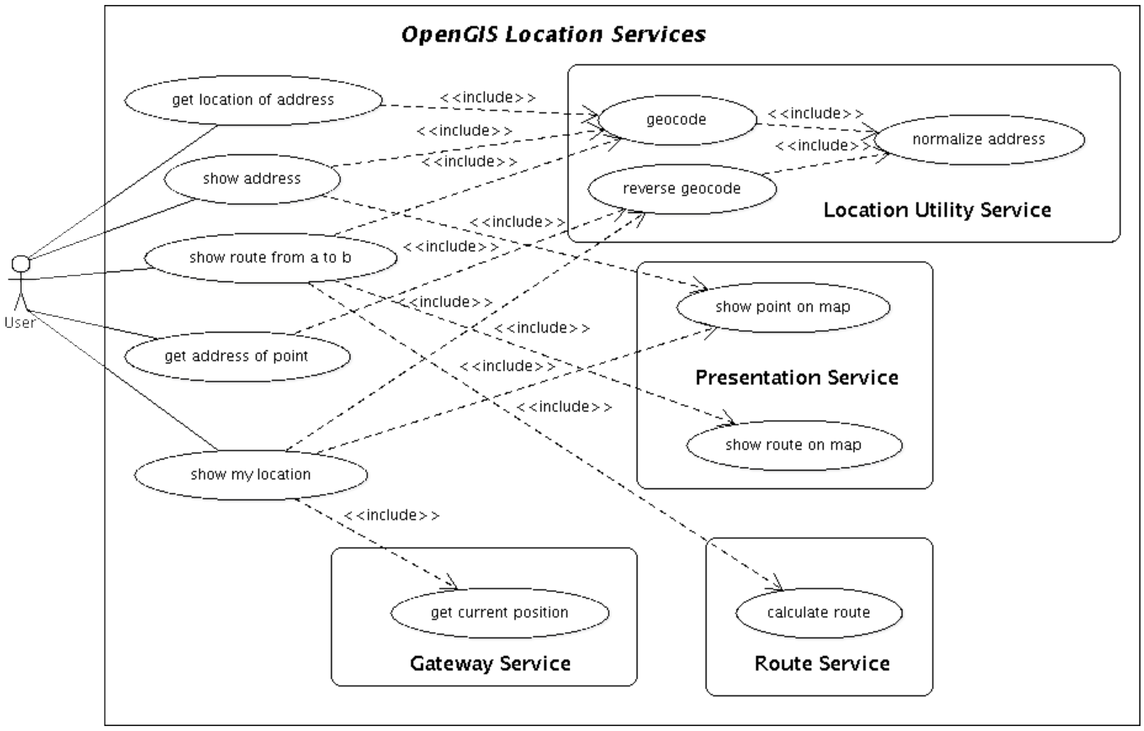


Figure 3.4: Example use case diagram for OpenGIS Location Services
Source: own assembly based on OGC (2008)

3.2.2.2 Abstract Data Types

Requests and responses within a *OpenGIS Location Service* are exchanged via XML over HTTP following an XML schema defined for that purpose[43].The actual spatial data is encapsulated in *Abstract Data Types (ADTs)* nested inside the XML requests and responses. These are well-known data types defined in a separate XML schema (see ADT.xsd on pp.49 in OGC (2008)[44]). The following ADTs are important to know about in the context of the *Location Utility Component*:

Address ADT	Contains an address information for a geographic place, normally consisting of a street address, a municipal, country and a postal code.
Area of Interest ADT	Defines an area of interest for a location request as a circle, bounding box or polygon. No matches outside of this area will be returned.
Position ADT	The description of a geographic position (a point geometry).

3.2.2.3 Request and Response Parameters

The *OpenGIS Location Services* specification defines a set of parameters, which each request and response to and from the *Location Service* component may or must include (OGC, 2008, pp.20):

A **geocode request** must include a list of one or more addresses as *Address ADTs* which may be unstructured free form, partial or complete addresses.

A **geocode response** must include a list of normalized addresses as *Address ADTs* including point geometries matching the requested address. It may as well provide a *GeocodeMatchCode* providing information about the accuracy of the match defined by a float value ranging from 0 to 1 and a *MatchCode* describing the algorithm used to find the match.

A **reverse geocode request** must include the point to be transformed into an address as a *Position ADT* and may as well include information about the area of

[43]schema available at: http://schemas.opengis.net/ols/1.2.0/LocationUtilityService.xsd
(last accessed: March 29th, 2009)
[44]also available at: http://schemas.opengis.net/ols/1.2.0/ADT.xsd
(last accessed: March 29th, 2009)

interest as an *Area of Interest ADT* and the preferred type of address returned (street, postalcode, point of interest, etc.). If no preference is given the default is to return the nearest street address.

A **reverse geocode response** must include a list of normalized addresses and the exact location of each address as a point geometry. It may as well include the distance from the requested point to the point found. The list of addresses must be sorted by their distance to the requested point.

3.2.2.4 Example Requests and Responses

For a more practial explanation of the *Location Utility Service* some examples of valid responses and requests will be given. Listing B.1 on page 92 shows a valid geocode request containing a freeform address search. The matching response containing the geocode result is shown in listing B.2 on page 93. Listing B.3 on page 94 shows a valid reverse geocode request and listing B.4 on page 95 shows its result.

3.3 OpenRouteService

3.3.1 Overview

OpenRouteService is an online route planning service conforming to the *Route Service* component of the *OpenGIS Location Services* specification. It not only provides route planning features but implements the *Locating Utility Service* component used for geocoding purposes, as well as the *Directory Service* for point-of-interest directed searches. The service has been designed and implemented within the scope of a master thesis[45] written at the University of Applied Sciences Mainz by Pascal Neis[46] and is now maintained and further developed by the Research Group Cartography, University of Bonn, lead by Prof. Dr. Alexander Zipf[47].

Apart from the HTTP/XML-based interfaces defined in the *OpenGIS Location Services* it provides a web-based user interface[48], serving as an easy to use frontend

[45]"Location Based Services mit OpenStreetMap Daten", see Neis (2008)
[46]preliminary work has been done in the scope of a diploma thesis (Neis, 2006)
[47]Pascal Neis is a member of the research group and still the main developer of the project
[48]http://www.openrouteservice.org/

for users to access the route planning and geocoding functions, e.g. online mapping of geocode results and routes. As shown in figure 3.5 ORS makes as well use of WFS/WMS[49] Geoservers and includes existing rendering engines from the OpenStreetMap project for the actual presentation of the maps.

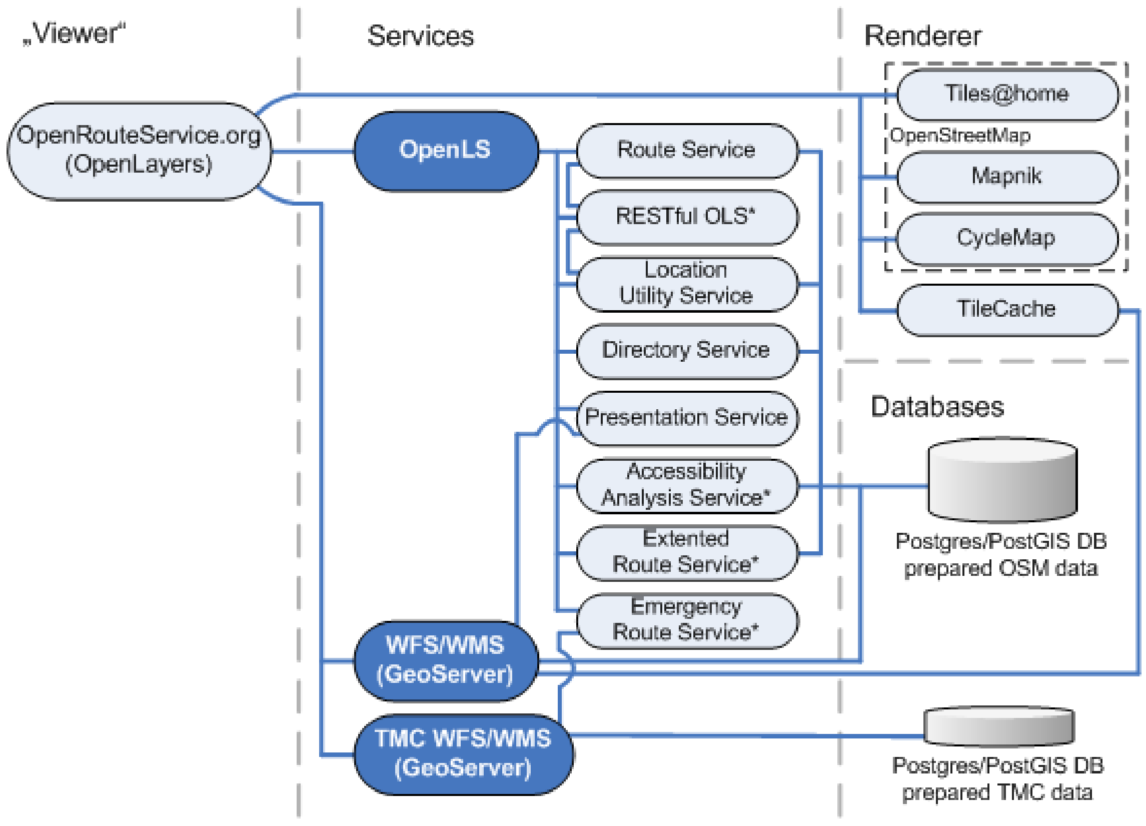


Figure 3.5: Components of OpenRouteService
Source: http://wiki.openstreetmap.org/images/6/66/ORS_Components.png
(last accessed: April 22nd, 2009)

The capabilities of the routing component include:

- car navigation by fastest or shortest route
- bicycle routing
- pedestrian routing
- inclusion of intermediate stops
- definition of *avoid areas*[50] for route calculations, which are a prerequisite for emergency routing purposes (define flood areas, etc.)

[49]WFS/WMS: Web Feature Service / Web Map Service
[50]areas which must not be intersected by the route

- accessibility analysis[51]
- realtime integration of traffic news (limited to specific areas)
- generation of driving instructions ("turn left at...")

The specialty of *OpenRouteService* compared to other publicly available services with similar capabilities is that it can be called "three times open" (Neis and Zipf, 2008), as it is based on…

- *open standards* as defined in the *OpenGIS Location Services*
- *open source software*, e.g. *PostgreSQL Database Management System*
- *open geographical data* derivated from the *OpenStreetMap* project

3.3.2 OpenStreetMap Data Integration

As mentioned before, OpenRouteService is primarily based on spatial data generated by the OpenStreetMap project. For this purpose, weekly XML exports of OSM data (as shown in section 3.1.4.2) are preprocessed by custom Java programs which parse and filter the export datafiles, extract the relevant geographic data and finally insert them into appropriate tables in a PostgreSQL/PostGIS database (see figure 3.5 on the previous page). The whole process has been developed and described by Neis (2008) and will not be discussed in detail here.

3.3.3 The Geocoder Component

OpenRouteService already included a basic geocoder component before the work presented in this thesis had begun. It was already conforming to the *Location Utility Service* as described in section 3.2.2 and primarily used by the web frontend to locate start and end points for routing calculations but also directly accessible via an HTTP/XML interface.

The major drawbacks of this early implementation were the following:

- no possibility to search for house numbers
- no approximate string matching features, hence only "perfect equality match or nothing"

[51]giving answers to the question: "what can be reached from a specific point within a given amount of time?"

- no attribute relaxation techniques included, hence "either all attributes match or no result is returned"
- randomly selected points for a street are given as result instead of center-point
- unfavourable modeling of the reference dataset
- poor performance

3.4 PostgreSQL Database Management System

3.4.1 Overview

PostgreSQL is an open source object-relational database management system, released under the *BSD license*[52] which makes it free to use for any purpose. Its development started in 1986 under the project name *Postgres* at the University of California at Berkeley (UCB) as a followup project to the *Ingres* database management system (hence the name "Postgres", meaning "post ingres"). It has later beed renamed to *Postgres95* and finally to *PostgreSQL*[53]. Since 1996 the software is maintained and developed outside of the UCB as an open source project with volunteered distributions ranging from single programmers to large corporations from all over the world.

PostgreSQL claims to strongly conform to the standards for the SQL database language as defined by the *American National Standards Institute (ANSI)*[54] in 1992 and 1999 respectively[55] and provides a rich list of advanced database features such as[56]...

- views and triggers
- stored procedures using its own procedural language *PL/pgSQL* as well as many native programming languages such as C, Perl, Java, Ruby, Python, etc.

[52]available at: http://www.postgresql.org/about/licence
(last accessed: April 15th, 2009)
[53]historical informations taken from: http://www.postgresql.org/about/history
(last accessed: April 15th, 2009)
[54]http://www.ansi.org/
[55]also known as *SQL-92* and *SQL:1999*
[56]for detailed explanations of the features listed see PostgreSQL (2008)

- database replication
- point-in-time recovery
- online backups
- storage of binary objects like images, audio files, etc.
- advanced indexing technologies like *Generalized Search Trees (GIST)* and *Generalized Inverted Indexes (GIN)*
- query rewriting using a rules system

3.4.2 PL/pgSQL procedural database language

PostgreSQL, just like almost any commonly used database management system, features its own procedural database language called *PL/pgSQL* (PostgreSQL, 2008, pp.721). It is very similar to other widely used procedural database languages like e.g. Oracle's *PL/SQL*. *PL/pgSQL* features a wide range of standard programming language elements such as variables, control structures (loops, conditions, etc.), functions or error handling[57].

3.4.3 The PostGIS Extension

PostGIS is an extension to the *PostgreSQL* database management system, adding the possibility to store and analyze geographic objects in the database. It is developed by *Refractions Research Inc*[58] as a research project and is made freely available for any use under the terms of the *GNU Public License*[59]. PostGIS follows the *OpenGIS "Simple Features Specification for SQL"* and has been certified by the *OGC* as being compliant to the *OpenGIS Simple Features Specification for SQL, Version 1.1, Types and Functions Alternative*[60].

The PostgreSQL/PostGIS combination serves as a popular alternative to commercial spatially enabled database management systems like e.g. *Oracle Spatial*

[57]the complete list of features including example programs is included in PostgreSQL (2008)
[58]http://www.refractions.net/
[59]see Free Software Foundation (1991)
[60]certificate available at http://postgis.refractions.net/files/opengis-certification.png (last accessed: April 15th, 2009)

and is used in various kinds of projects ranging from commercial applications and services to research projects[61].

Additionaly, PostGIS is natively supported by almost all open source *Geographical Information Systems (GIS)* e.g. including *Quantum GIS*[62] and *gvSIG*[63]. The support for PostgreSQL/PostGIS databases among commercial GIS solutions however is significantly lower. They mostly require middleware components such as a *Web Feature Service* serving as an interface or separately licensed add-on products to gain access to the spatial objects stored inside the database. As an example, ESRI's popular *ArcGIS* suite offers no native support for direct PostgreSQL access although independently developed commercial interfaces like the *zigGIS* plugin by *Obtuse Software LLC*[64] are filling this gap[65].

[61] a list of case studies is available at http://postgis.refractions.net/documentation/casestudies/ (last accessed: April 16th, 2009)

[62] http://www.qgis.com/

[63] http://www.gvsig.gva.es/

[64] http://pub.obtusesoft.com/

[65] beginning with version 9.3 of ArcGIS Server, ESRI included native support for access to PostgreSQL/PostGIS spatial databases via their ArcSDE database interface (see ESRI (2008))

4 Analysis and Methodology

In this chapter the tasks and challenges involved in implementing the geocoder will be analyzed and appropriate methodical approaches to them will be developed. The result will be a methodical guideline for the implementation phase addressed in chapter 5.

4.1 Reference Dataset

As pointed out by Ratcliffe (2001), Grubesic and Murray (2004) and Waldner et al. (2005), the reference dataset is a very important (if not the most important) component of a geocoder, as it directly influences the overall quality of the geocoding results in terms of coverage and accuracy. In this section the general suitability of the OSM data for geocoding purposes will be discussed and necessary preprocessing steps to integrate the data into an appropriate reference dataset will be developed.

4.1.1 Completeness and Accuracy

The amount of spatial data collected by the project's contributors is already impressive and constantly growing in size and coverage. At the time of writing the project database consisted of 325,210,959 individual nodes and 26,181,088 linestrings collected by 101,934 registered users[66]. Figure 4.1 shows the development of GPS track points collected and the number of project members registered since August 2005.

Recent research performed by Haklay (2008), comparing the completeness and accuracy of the OpenStreetMap data to official Ordnance Survey datasets in

[66]taken from OpenStreetMapStats (2009)

the UK, showed that the average distance between two similar objects in the databases is only 6 meters (and thus close to the maximum precision level available using conventional GPS handheld devices) while it took only four years to collect about 29% of the area of England. Haklay (2008, p.24) however points out that the quality of the data is very inconsistent and can vary significantly.

Considering that commercial online mapping services such as Google Maps^{TM} depend on proprietary data licensed by commercial mapping companies like *Navteq*, OpenStreetMap often has significant advantages in terms of actuality (changes can be implemented within minutes) but especially in terms of completeness in areas which are not well covered by commercial providers. An impressive example of this point can be observed by comparing the map for a part of the city of Baghdad from *Google Maps^{TM}* to *OpenStreetMap* as shown in figure A.2 on page 87. Both maps show the same randomly selected area within the center of Baghdad at identical zoom levels, yet OpenStreetMap's version is way more

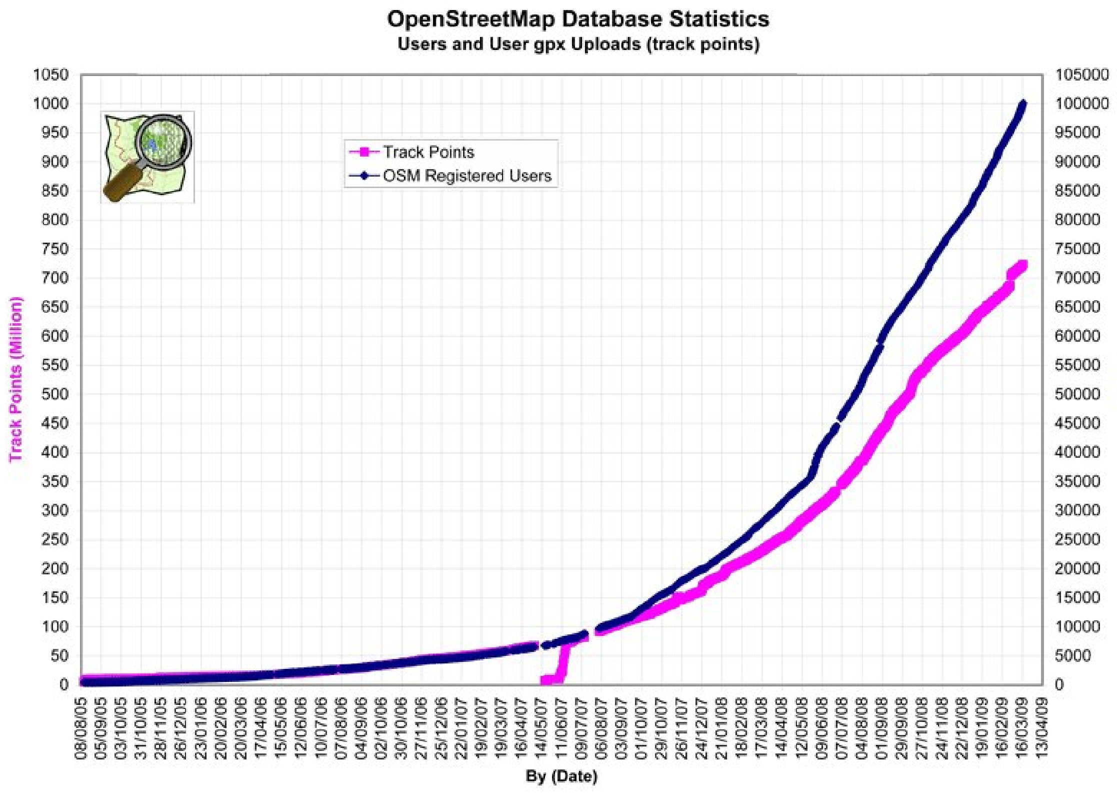


Figure 4.1: OpenStreetMap database statistics
Source: http://wiki.openstreetmap.org/wiki/Statistics
last accessed: March 22nd, 2009

detailed than the one offered by Google Maps^{TM}.

It is nevertheless difficult to measure the overall *completeness* and *coverage* of the OpenStreetMap data because there is no general definition of what "complete" means in this context as it depends heavily on the intended use of the data. As an example, the dataset for a given area lacking information about footpaths may be perfectly suitable for a car navigation system while at the same time useless for pedestrian routing.

4.1.2 Suitability of OSM Data Model for Geocoding

Looking at the data model of OpenStreetMap as described in section 3.1.3 it becomes clear that it was not designed to serve as a reference dataset for a geocoder in the first place but simply to generate maps. The following inherent structural deficits of the OpenStreetMap data model have to be considered when implementing the geocoder:

The meaning of an object is determined by its tags only

There is no distinction between a city, a building, a vertex of a street line or even a postbox in terms of its object type. All of these real world entities are represented by a *node*. Their meaning is instead determined by appropriate tags, e.g. the tags *place=city* and *name=Salzburg* mark a node as the center of a city named "Salzburg" whereas a node with the tag *amenity=postbox* refers to a postbox.

To make it even worse, similar entities of the real world may be stored using different object types, e.g. a building may be represented as a *node* (defining its centroid only) or using a *closed way* (defining the actual structure of the building as a polygon[67]).

There is no structural relation between objects belonging together

Apart from using the object type *relation* (which is not widely used) there are no structural relations between objects which should logically be considered as

[67] the building may even be represented by multiple, unrelated closed ways, each representing a part of the building

belonging together. A street, for example, may consist of multiple, not directly connected line strings, each represented by a *way* object in the database. However, the obviously strong relation of these line strings ("they are forming a street!") is *not* inherently represented in the data model.

The same problem arises when trying to determine hierarchical dependencies between entities, e.g. determining the city a street belongs to or determining the street a building belongs to, etc. This is not a problem when *mapping* those entities, because their *logical* relation can be determined by their *spatial* relation on the map (when looking at a map, it is mostly sufficient to "see" that a given building is "near" a street), but nevertheless a major problem when trying to build a geocoder which needs to reliably assign the building to a street.

Administrative boundaries

There still is an ongoing discussion among the OSM community members about how administrative borders or areas should preferrably be stored[68], and (partly because of this) the mapping of borders is still not complete. This makes it particularly hard to assign an object to an administrative unit, e.g. to assign a German city to its federal state or even to its country. Additionally, areas marking postal codes, which play an important role in the addressing system of many countries, are not included at all.

For the scope of this thesis it has thus been decided to integrate additional geographic data which may be replaced by OSM data later. The additional data used is the sample data for Europe from ESRI's product *ArcGIS Desktop 9.1* which includes vector data representing postal code areas for the scope of Germany as well as administrative borders for all European countries including 1st level administrative units such as federal states (ESRI, 2005).

4.1.3 Proposed Data Model

The design of the data model used for the reference dataset is primarily determined by the list of attributes a user should be allowed to use as search terms. The following attributes have been selected to represent a general model of how

[68]see discussion at: http://wiki.openstreetmap.org/wiki/Talk:Relation:boundary (Last accessed: April 19th, 2009)

addresses may be constructed and these should form the basis of the data model for the reference dataset to be implemented:

- Country
- Country Subdivison (e.g. states in the USA, cantons in switzerland, etc.)
- Postal Code
- Municipal
- Municipal Subdivision (e.g. districts of a city)
- Street
- House Number

Taking into account the differences between countries in their addressing schemes (e.g. some countries do not have postal codes, some have multiple or no country subdivisions, etc.) this list shall only form the basic attributes which *may* be part of an address and should therefore be represented in the reference dataset model without being strictly required for a given place oder address. It is merely a list of attributes which "may be used as a search term for the geocoding process". Figure 4.2 shows an example relation diagram for the addressing scheme used in Germany (which will be the primary study area in the implementation phase).

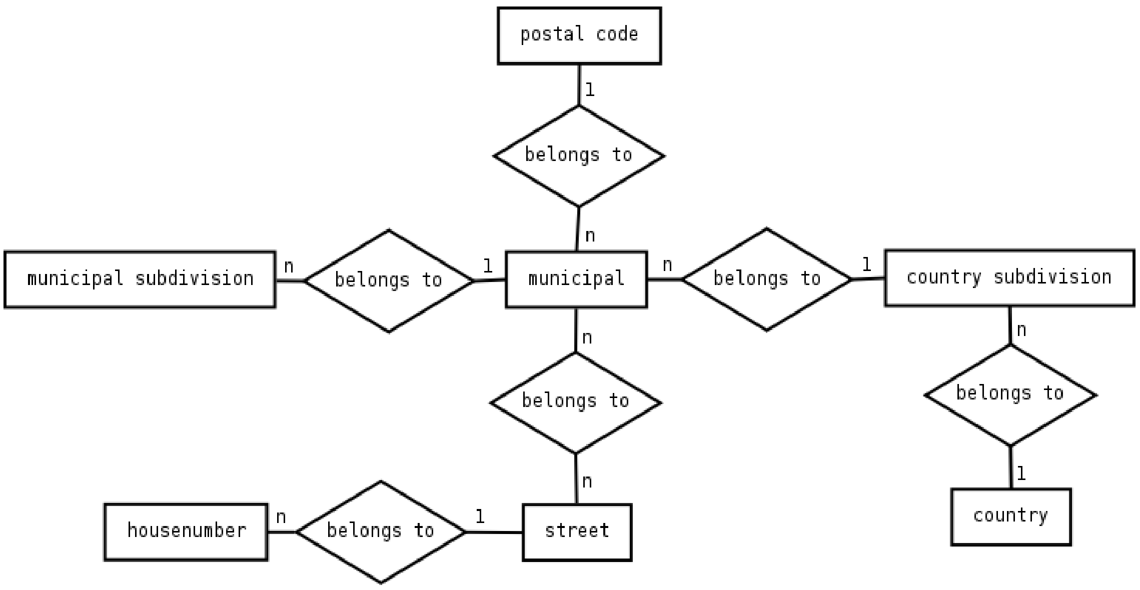


Figure 4.2: Relation between entities of the reference dataset in Germany
Source: own assenbly

4.1.4 Data Integration

As shown in the previous section, the OpenStreetMap data ist not directly usable as a reference dataset but has to be filtered and transformed for that purpose. The general steps necessary for this task are[69]:

Extraction	The data has to be extracted from its original source.
Reduction	Filtering out the relevant parts of the data.
Cleaning	Correcting errors in the source data.
Standardization	Standardization of attribute values.
Aggregation	Joining data sources (if more than one is used).
Semantic Transformation	Transforming the data to the schema in the target database.
Loading	Inserting the data into the target database.

The **extraction** step depends heavily on the format of the input data. In case of OSM weekly XML dumps, an XML-parser has to be implemented using programming languages capable of string processing like *C*, *Perl* or *Java*. **Reduction** in terms of OSM means to filter out all objects and attributes which are not necessary for geocoding purposes. If waterways e.g. are not declared as valid results of the geocoding, all line strings referencing them can be left out.

Cleaning in this case includes the correction of syntactical or typographic errors, e.g. multiple blank spaces that should be reduced to a single blank, removal of illegal characters, etc., as well as the detection of inconsistencies and logical errors as far as possible.

The data has to be **standardized** as well, e.g. multiple spellings or typings of a similar logical value have to be consolidated ("Hellbrunnerstraße" vs. "Hellbrunnerstr." etc.).

If more than one data source is used, they may have to be joined and **aggregated**, too. Before the final insertion into the reference dataset they may as well have to be **semantically transformed** to fit the target scheme of the reference dataset first.

Figure 4.3 on the following page shows the consecutive steps, which form the process of data integration.

[69]based on suggestions by Han and Kamber (2006) and Rahm and Do (2000)

4.2 Designing The Geocoding Process

In this section the geocoding process to be implemented will be designed. A geocoding process basically consists of three steps:

1. Parsing the input data.
2. Searching for matching objects in the reference dataset.
3. Generating a geographic reference in the required output format.

Davis et al. (2003) name these steps the *parsing*, *matching* and *locating* phase.

4.2.1 Parsing

In this phase the input data is transformed into a standardized form which may later be used during the matching phase. It consists of three steps:

1. Segmentation
2. Cleaning
3. Standardisation

Segmentation

If the input data contains an unstructured text like a complete address in raw format (e.g. “Hellbrunnerstraße 34 Salzburg Österreich”) it has to be **segmented** first, i.e. split into separate logical components. The segments have then to be assigned to appropriate attributes of the reference data, in the example given before this would be *street name*, *house number*, *municipal* and *country*. Depending on the input data this task can be very challenging and a great deal of literature

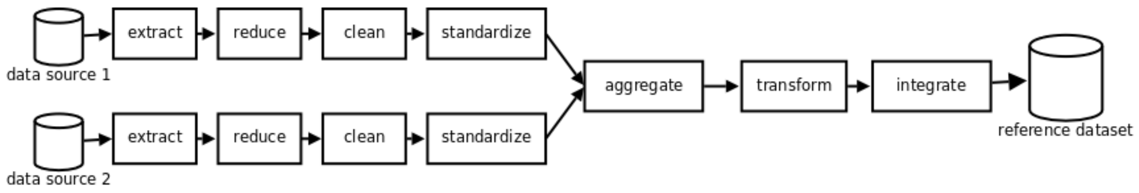


Figure 4.3: The data integration process
Source: own assembly

has been written focusing this problem by using different approaches including complex probabilistic algorithms like *Hidden Markov Models* (Christen and Zhu, 2002; Borkar et al., 2001).

Figure 4.4 illustrates the process in a simplified version.

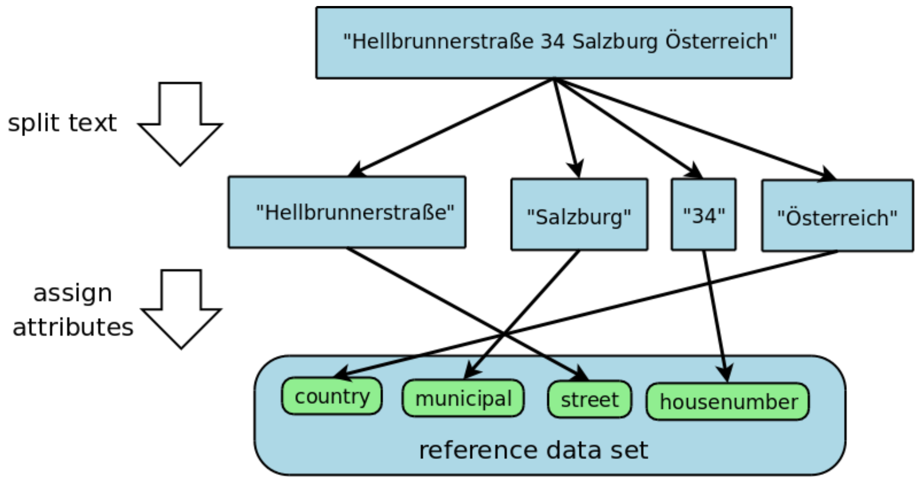


Figure 4.4: Segmenting freeform search texts
Source: own assembly

Both the *splitting* and the *assigning* contains challenges which must be solved in the implementation. For usability improvement, the user should be allowed to either use *commas* (preferred way) or *blanks* as field separators for the freetext search. The *blank* as a field separator however can lead to complicated issues if a single logical component of the search term (e.g. a street name) contains blanks itself. This problem is illustrated in figure 4.5 on the following page and must be dealt with in the implementation.

The *assignment* of the single components of the search term to matching attributes in the reference dataset should be done by a combination of the following approaches (based on Goldberg (2008)):

lexical analysis	Search text components are assigned to attributes of the reference dataset by lexical analysis of the text using string matching patterns. A 2-digit number e.g. is most certainly referring to a house number, whereas a 5-digit number (in Germany) can only be a postal code, strings ending with “str.” are most certainly referring to a street name, etc.

lookup table	All components which can not be assigned to an attribute because of their lexical structure should be tested against the reference dataset, i.e. if for a given string a matching *municipal* in the reference dataset exists, it is likely that this is the "city component" of the search term, and so on.

Cleaning

Apart from segmenting the input data a **cleaning** algorithm is used to remove invalid or unnecessary characters from the input text if necessary. This algorithm may as well incorporate logical restrictions to eliminate invalid data such as invalid postal zipcodes, invalid combinations of characters etc. as addressed by Christen and Zhu (2002).

Standardization

The last step is the **standardization** phase which is mainly used to convert multiple textual representations of the same logical content into the format which is used in the reference data, e.g. abbreviations for street types ("Str." vs. "Straße" or "Ave" vs. "Avenue").

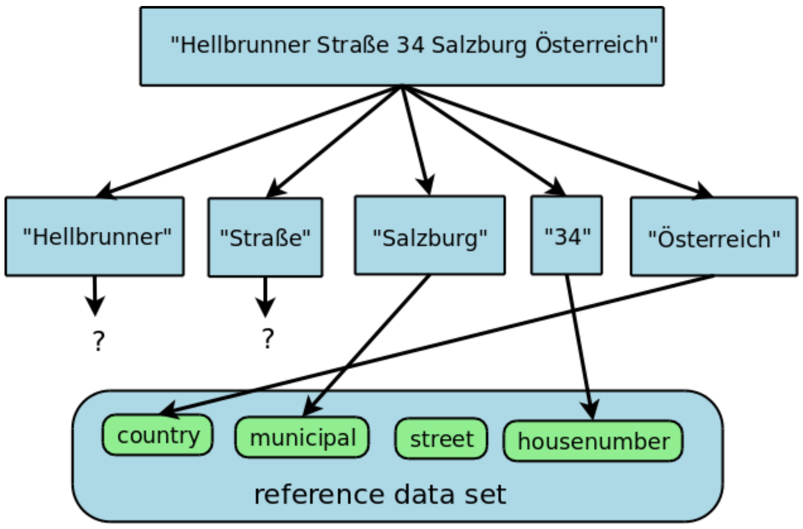


Figure 4.5: Splitting freeform search text by blanks problem
Source: own assembly

4.2.2 Matching

In this phase the best matching geographic feature in the reference dataset is retrieved. Goldberg (2008, p.71) as well as Christen and Churches (2005) classify the algorithms to match geographic features into two main categories:

deterministic:	a perfect matching feature in the reference dataset is found
probabilistic:	no perfect matching feature is found but one or more features match with a certain probability

The **deterministic** approach minimizes the geocoding errors as only perfect matching features are considered but it considerably limits the *match rate*[70] of the geocoding process and is sensitive to misspellings or typing errors in the input text or reference dataset.

In practice this often leads to an iterative approach in which the requirements for a match are constantly relaxed until a matching feature is found, e.g. if the search for a given city, street and house number returns no match in the next step only city and street is used and so on. Goldberg (2008, p.71) calls this method *Attribute Relaxation*.

The **probabilistic** approaches use statistical models instead to find matching features in the reference dataset and rely on rather complex mathematical algorithms which will not all be discussed in detail here.

A widely used example of probabilitic feature matching is *approximate string matching*. In this case string matching functions are used which determine the probability that two strings "mean" the same, although they do not match perfectly on a per-character basis. Goldberg (2008, p.80) calls this *essence-level equivalence*.

An introduction to basic principles and techniques used in the field of approximate string matching as well as examples of different implementations have been presented by Navarro (2001). Snae (2007) explains the most common matching algorithms and compares them concerning their accuracy under different conditions. Christen (2006) and Lait and Randell (1996) have compared common matching algorithms with emphasis on name matching.

For the geocoder to be implemented, matching should preferrably be done by a deterministic approach followed by probabilistic approaches (if no match found).

[70]the percentage of geocoding requests which return a result (Cayo and Talbot, 2003)

Futhermore, *attribute relaxation* techniques should be integrated to improve the match rate.

4.2.3 Locating

If the matching phase returned an appropriate object of the reference dataset, this result has then to be turned into a valid geocoding result.

This may be trivial, e.g. if the point coordinates of a specific address have been searched for ("Hellbrunnerstraße 34 Salzburg") and a point object referencing exactly this address has been found. In this case the coordinates of the reference object could simply be used as the geocoding result.

If the matching object in the reference dataset is a polygon instead, the point coordinates have to be determined first, e.g. by calculating the centroid of the polygon (Dearwent et al., 2001).

If no exact match has been found in the reference dataset, the whole process becomes considerably more complex. In these cases interpolation techniques may be used to determine the approximate location of the searched object. A great deal of research has already been conducted, addressing various questions in this complex field (Davis et al., 2003; Zandbergen, 2007*b*; Wiezoreck et al., 2004; Bakshi et al., 2004) and solving this particular problem will be one of the major challenges in the implementational part of this thesis.

For the geocoder to be implemented, whenever a point object matching the searched address could be found in the reference dataset (e.g. a point reference to a city center), the geocoder should simply return the exact coordinates of that object. If a street is searched for, the point coordinates to be returned have to be calculated by determining the middle of the street (which may turn out to be rather complicated if the street consists of multiple unconnected segments). The same holds true if a request for a particular house number matches a building stored as a polygon.

4.3 Use Cases

Three different interfaces, each representing a particular use case of the geocoder, have to be implemented:

Freeform Text	An arbitrary textual description of an address is sent to the geocoder and a normalized address including an X/Y coordinate pair is returned.
Structured Text	A list of of search texts, each representing a specific attribute of an entity in the reference dataset, is sent to the geocoder and a normalized address including an X/Y coordinate pair is returned.
Reverse Geocoding	An X/Y coordinate pair is sent to the geocoder and a normalized address is returned.

Figure 4.6 visualizes theses use cases and shows their connection to the standard steps involved in the geocoding process as discussed in section 4.2.

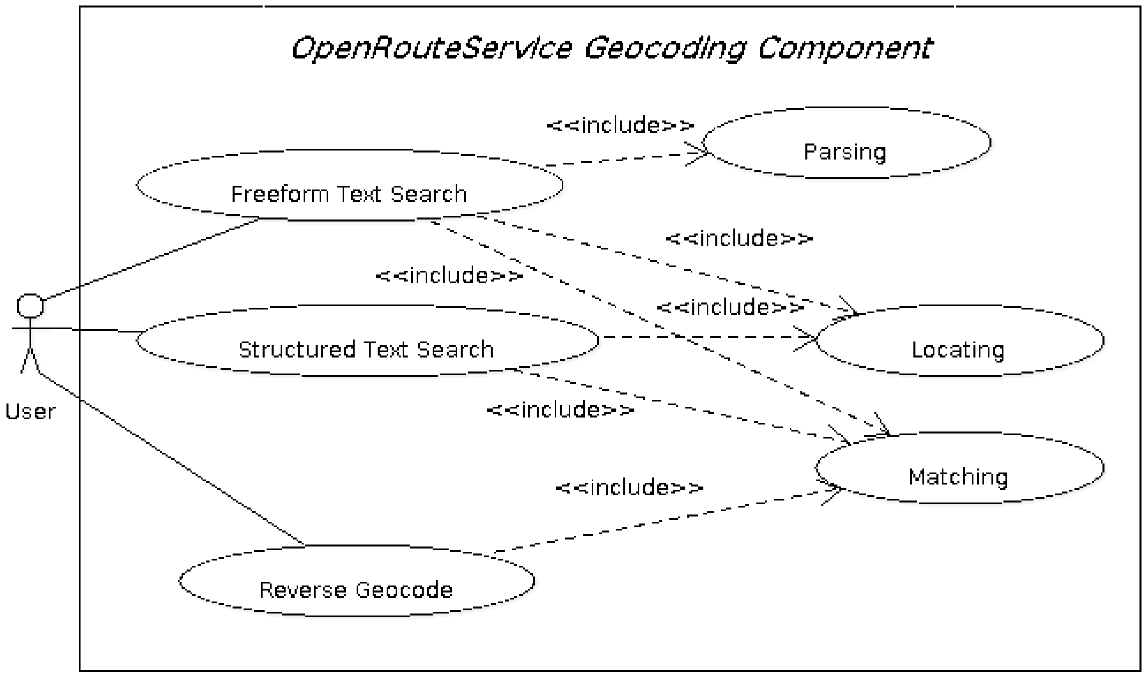


Figure 4.6: Use Case diagram for the ORS geocoding component
Source: own assembly

Each of the three use cases should return a geocode result consisting of. . .

- a normalized address of the matched object in the reference dataset
- a coordinate pair defining the match location
- the complete geometry of the match
- an indicator defining the quality of the match

4.4 House Numbering

The geocoding of addresses including house numbers includes specific challenges, which will be separately addressed in this section.

4.4.1 Historical Overview

As a European citizen living in an urban area, it is hard to imagine everyday life without street names and house numbers. The *street name / house number* referencing system to identify buildings and places is vital to many different applications such as sending mail or urban navigation but it is also the reference system used to define people's home locations. In Germany e.g. the home address is vital to administrative services ranging from tax collection to welfare. Thus it is not solely used to identify places and buildings but as a means of *personal identification*[71].

The main driving force for the development of a *street name / house number* addressing scheme is *urbanization*. For people living in rural communities below a certain size, it was just not necessary to assign a specific address to a place, because everybody living in the village just "knew" where everbody else's houses and public places were. Another driving force are administrative purposes, generating the need for non-local people to identify houses via an address such as tax collection, census or military service draft.

The first general street naming and house numbering systems in Europe have thus been established in the developing urban areas of centrally managed European civilizations. According to Tantner (2006) and Farvacque-Vitkovic et al. (2005) the first efforts to systematic addressing (although initially limited to specific areas) in Europe have been made in the 18th century, e.g. in Madrid in 1750, Trieste in 1754 or in London 1762. Since 1768 provincial cities were also included in the addressing system in France. In 1799 the Prussian king Friedrich Wilhelm III ordered the systematic house numbering of Berlin and in 1857 the kingdom of Hannover established the law that every building in its dominion had to be given a house number (Königlich Hannoversches Ministerium des Inneren, 1857).

[71] this is further demonstrated by the fact that the German identification card (the "Personalausweis") includes the person's home address)

4.4.2 House Numbering Systems

Although most developed regions in Europe and elsewhere in the world developed some sort of addressing scheme for buildings, the schemes themselves or more specifically their underlying algorithms and ordering systems can be significantly different even throughout a single country. However, a basic principle shared between any of them can be found:

- a building is given an identifier, mostly an (alpha-)numerical value
- each building is further assigned to a higher level administrative unit
- the identifier is unique among all buildings assigned to the same administrative unit

In most cases, especially in Europe, the *administrative units* are streets, but it may as well be a block (e.g. used in the German city of Mannheim[72]) or an informal rural settlement. The schemes defining how the identifiers are generated and assigned are even more diverse. In most European areas buildings are assigned to a street and numbered in ascending order from the defined starting point of the street, where one side of the street has odd numbers and the other side has even numbers[73].

In other areas, e.g. parts of Berlin, a system was established which numbers the buildings along a street sequentially from a starting point on one side of the street and continuing the numbering on the other side back to the beginning of the street (the "horseshoe" or "U-shaped" approach). In more planned cities, like most of the cities in the USA e.g., buildings are often not just sequentially numbered, but the house number itself contains information about the distance from a reference point[74] (either a defined starting point of the street or a central point from which the street is divided into north/south or east/west sections). This system is called *metric numbering*[75] and may be combined with an odd/even rule to determine the side of the street. Figure 4.7 on the following page visualizes examples of these numbering systems.

[72]see Tantner (2006)

[73]Farvacque-Vitkovic et al. (2005, p.149) calls these schemes *sequential alternating numbering systems*

[74]for an example of this scheme see Fonda-Bonardi (1994), describing the numbering system used in Los Angeles

[75]see Farvacque-Vitkovic et al. (2005, p.148)

Apart from these approaches to house numbering there are schemes in use which (from a systematical point of view) seem very odd. In most parts of Japan e.g., houses are not assigned to streets but to a "neighborhood" or block, and house numbers are sequentially assigned according to the *construction date* of the building (Farvacque-Vitkovic et al., 2005, p.9). As a result, house numbers do not contain any spatial reference at all, hence they are useless for navigational purposes or spatial analyses. Farvacque-Vitkovic et al. (2005) satirically states that the easiest way to find a particular house number in Japan "is thus to go to the nearest police station, as the Japanese themselves do" (Farvacque-Vitkovic et al., 2005, p.9). A similar system, assigning buildings to parcels rather than streets, is used in Korea (see Kim (2001) for an explanation).

4.4.3 Using Existing Data

4.4.3.1 House Number Data in OpenStreetMap

Until the beginning of 2008 no house number data had been integrated to OpenStreetMap at all. This was partly due to the lack of a commonly agreed scheme defining how to map house number information. Several propopals had been suggested by OpenStreetMap community members[76] but neither of them has officially been approved. However, one particular proposal has now become a de-

[76]a list of proposals and arguments is available at: `http://wiki.openstreetmap.org/index.php?title=Proposed_features/House_numbers` (last accessed: April 19th, 2009)

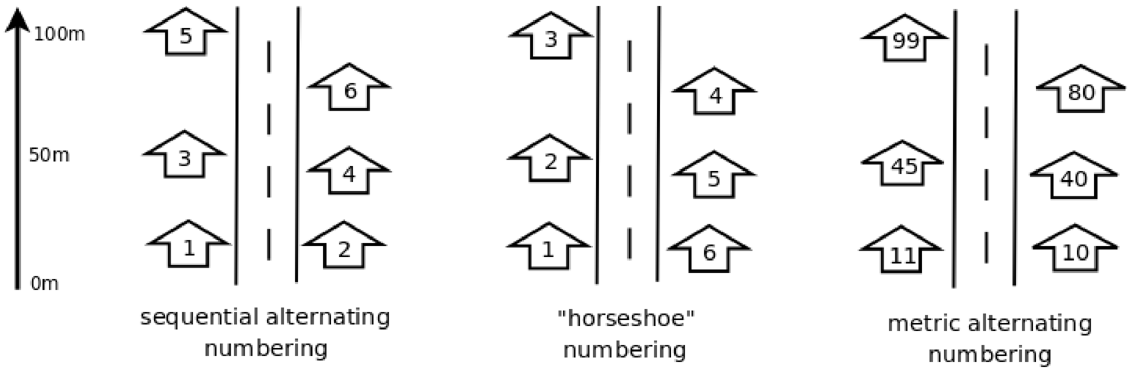

100m
50m
0m
5
3
1
6
4
2
sequential alternating numbering
3
2
1
4
5
6
"horseshoe" numbering
99
45
11
80
40
10
metric alternating numbering

Figure 4.7: Examples of street numbering systems
Source: own assembly

facto standard because a group of mappers started using it to map house numbers and over time other users just adopted it[77]. The proposal is called the "Karlsruhe Schema" and has first been published as a wiki page in April 2008[78].

The basic working principles of this scheme are:

- Single house numbers can be attached to nodes or buildings (represented by closed ways).
- A range of house numbers along a street can be defined via ways called "interpolation lines" starting and ending at a node which includes a house number.

The house number itself is assigned using the tag `addr:house number`.

Single House Numbers

Listing 4.1 shows a node containing a house number conforming to the *Karlsruhe Schema* whereas listing 4.2 on the following page is an example of a house number assigned to a building.

```
<node id="123" lat="47.7659168508311" lon="13.058013414565503">
  <tag k="addr:housenumber" v="34" />
</node>
```

Listing 4.1: Assigning house numbers to nodes using the Karlsruhe Schema

Range of House Numbers

Interpolation lines are used to construct an imaginary way along which interpolation of intermediate house numbers between two known house numbers should take place. It is defined by the tag `addr:interpolation`. Its value is one of *odd*, *even*, *all* or *alphabetic*, defining the type of house numbers to be interpolated. Listing 4.3 on the next page shows an example of an interpolation line.

[77] this process of "standardization by acceptance" is quite common for OSM (and many other community-driven projects as well)

[78] the current version of the wiki page is available at: `http://wiki.openstreetmap.org/wiki/Proposed_features/House_numbers/Karlsruhe_Schema` (last accessed: April 16th, 2009)

Which street does a house number belong to?

The examples shown before did not include any information about which street the named house numbers actually belong to. In these cases the street is determined by selecting the nearest street relative to the given node oder building but the corresponding street may as well be assigned using the tag `addr:street`. Listing 4.4 on the following page shows an example of a node including house number and street information. It is also possible to assign a house number node or building to a street using a dedicated relation object but this practice has not

```
<way id="30514144">
  <nd ref="336827617"/>
  <nd ref="336827620"/>
  <nd ref="336827623"/>
  <nd ref="336827626"/>
  <nd ref="336827617"/>
  <tag k="name" v="Universität Salzburg Naturwissenschaftliche
     Fakultät"/>
  <tag k="area" v="yes"/>
  <tag k="building" v="yes"/>
  <tag k="addr:housenumber" value="34"/>
</way>
```

Listing 4.2: Assigning house numbers to buildings using the Karlsruhe Schema

```
<node id="1" lat="47.7971145" lon="13.0535358">
  <tag k="addr:housenumber" v="30" />
</node>

<node id="2" lat="47.7969243" lon="13.0539049"/>

<node id="3" lat="47.7967801" lon="13.0541624">
  <tag k="addr:housenumber" v="40" />
</node>

<way id="123">
    <nd ref="1"/>
    <nd ref="2"/>
    <nd ref="3"/>
    <tag k="addr:interpolation" v="even"/>
</way>
```

Listing 4.3: Defining interpolation lines using the Karlsruhe Schema

been adopted by the community (despite from a data modeling point of view being the better solution).

```
<way id="30514144">
  <nd ref="336827617"/>
  <nd ref="336827620"/>
  <nd ref="336827623"/>
  <nd ref="336827626"/>
  <nd ref="336827617"/>
  <tag k="name" v="Universität Salzburg Naturwissenschaftliche
     Fakultät"/>
  <tag k="area" v="yes"/>
  <tag k="building" v="yes"/>
  <tag k="addr:housenumber" value="34"/>
  <tag k="addr:street" value="Hellbrunnerstraße"/>
</way>
```

Listing 4.4: Assigning street information to house number nodes

Usage of the Karlsruhe Schema

As stated before, the *Karlsruhe Schema* has been adopted by the majority of the community members and is nowadays by far the most commonly used way to store house number information (especially in Europe). Table 4.1 shows the usage of different parts of the Karlsruhe Schema in Europe as of April 14th 2009, according to TagWatch (2009).

Usage Type	Number of objects
nodes/buildings including addr:housenumber	2,735,368
nodes/buildings including addr:street	2,711,030
interpolation lines defined by addr:interpolation	21,936
house number/street association via relations	2,037

Table 4.1: Usage of the Karlsruhe Schema in Europe

4.4.3.2 Exact Match

If a geocode request for an address including a house number is sent to the geocoder and this particular house number has been inserted in the OpenStreetMap data using the Karlsruhe Schema, the exact position of the node should be returned as result. If the house number is attached to a building polygon, the centroid of the polygon should be calculated and returned as result instead (Dearwent et al., 2001).

4.4.3.3 Interpolation

The *interpolation lines* of the Karlsruhe Schema have to be integrated into the reference dataset so that requests for house numbers along these lines are properly interpreted by the geocoder. Additionally, the following hypothesis should be tested in order to improve the geocode results:

> "When the positions of house numbers A and B along a given street are known and a geocode request for a house number X, which lies numerically between A and B, is sent to the geocoder, is it reasonable to determine house number X via interpolation although this assumption is not explicitly supported by an interpolation line?"

4.4.4 Probability Based Approach

Assuming a geocode request (either freeform or structured text) includes a house number whereas the reference dataset contains no house number information for that street at all, most geocoders would presumably simply ignore the house number information and return the center of the corresponding street as a result. Because house number data in OpenStreetMap is still rare and inhomogeneously distributed (some areas are almost completely mapped whereas others contain no house number data at all) one specific challenge of this work was to find out, if the location of a house number along a given street can be effectively approximated by probability based approaches. In other words, the question to be answered is:

> Is it possible to effectively approximate the position of a house number along a given street in the absence of any house number data?

The term "effectively" in this case is meant in the sense of "better than simply returning the center position of the street". As shown in section 4.4.2, house numbering systems can differ significantly, hence the following hypotheses will be limited to the most commonly used system in Europe, the *sequential alternating* *system* (see figure 4.7 on page 42). Due to the availability of a dataset including house number data from its official surveying agency, the study area will be further limited to the federal state *Northrhine-Westfalia* in Germany.

4.4.4.1 Parameters needed for the Calculation

To guess the location of house numbers assigned according to the *sequential alternating system* the following parameters have to be estimated:

1. What is the average distance between two house numbers?
2. What is the offset of the first house from the beginning of the street?
3. On which end of the street does the numbering start?

The first two of these parameters should be determined by spatial analyses of known reference datasets including house numbers. Additionally, different factors which might possibly influence these values should be statistically tested for correlations in order to improve the approximation results.

If the direction of a street has been determined and the average distance d between house numbers is known, the position p of house number x could be calculated as…

$p = d + (x + x \pmod 2 - 2) * d$

…assuming that the distance from the starting point of the street to the position of the first house equals half the distance between two houses, i.e the distance between two house numbers given an alternating sequential house numbering schema.

But this is not adequate, because of what Goldberg (2008, p.88) calls the "corner lot problem"[79]. The calculation above assumes that each building located next to a street is assigned to this street but this assumption may not hold true when investigating the first building along a street. The problem is visualized in figure 4.8 on the next page. House "B" may be assigned to street "Y", which would imply that it is the first house on this street, but it may as well be assigned to street "X", in which case house "A" would be the first house on street "Y".

The average offset from the street's starting point has thus to be determined separately by statistical analysis of a given reference dataset for the study area. If

[79]It has likewise been address by Bakshi et al. (2004)

the average distance between house numbers d and the offset o is known, the approximate position p of the building with house number x along a given street can then be calculated as:

$p = o + (x + x \pmod 2 - 2) * d$

Figure 4.9 visualizes the calculation of a house number position along a street.

It is important to mention that this only holds true for streets forming a simple linestring, i.e. the street does not consist of multiple non-connected segments and each segment apart from the first and the last segment is connected on both sides to exactly one other segment. As stated before, it further assumes an alternating sequential numbering system.

In order to guess the *direction* of a street, official guidelines on house numbering available for the study area (Northrhine-Westfalia) have been consulted. In 1979, the *North Rhine-Westfalian Association of Cities* ("Städtetag NRW") has published

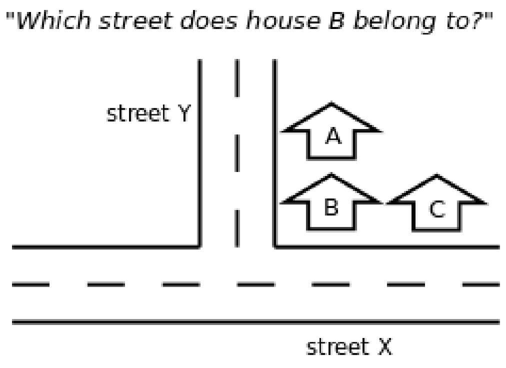


Figure 4.8: The corner lot problem
Source: own assembly based on Bakshi et al. (2004)

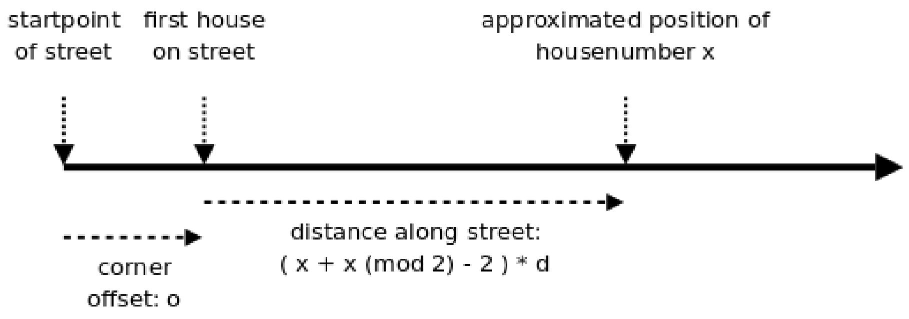

startpoint of street
first house on street
approximated position of housenumber x
distance along street: (x + x (mod 2) - 2) * d
corner offset: o

Figure 4.9: Calculating appromaximate house number positions along a street
Source: own assembly

a guideline on how to assign house numbers to buildings (see Städtetag NRW (1979)) containing the following passages:

1. The numbering system should be *sequential alternating* with odd numbers on the left and even numbers on the right side of the street (see section 5.1.1 of Städtetag NRW (1979)).
2. Numbering should normally start at the end of the street, which is closer to the city center (see section 5.1.3 of Städtetag NRW (1979)).
3. In dead-end streets, numbering should start at the end connected to another street (see section 5.1.3 of Städtetag NRW (1979)).
4. In developing areas, numbering should start at the connection to the main road (see section 5.1.3 of Städtetag NRW (1979)).

If the direction of the street, and the side, on which the searched building is located, are known, its approximate position may be considered not exactly on the street centerline, but with a certain offset[80] left or right to the street. Ratcliffe (2001) has shown that the value of this offset is hard to guess due to varying street widths and building positions relative to the street, so it has been decided to initially just calculate a point along the street centerline.

4.4.4.2 Hypotheses for Educated Guesses

Based on the guidelines for the order of numbering, the following hypotheses have been constructed and should therefore be tested using statistical methods:

1. If only one of the end points of a street is connected to another street, the street starts at this point.
2. If a street proceeds away from the city center in a radial direction, the street starts at the end closest to the city center.
3. If a street is on one end connected to a street of a higher rank as it is on the other end, the street starts at the end connected to the higher level street.

Additionally, the following hypotheses have been constructed to test influences on the calculation's parameters by analysis of correlations:

[80]Goldberg (2008, p.84) calls this a "dropback"

1. The average distance of houses along a street correlates with the street's distance to the city center (built-up areas may become less dense as they proceed away from the city center).
2. The average distance between houses along a street correlates with the length of the street (longer streets are more likely to contain "gaps" with no buildings).
3. The average distance between houses along a street correlates with the number of surrounding streets (bigger lots in less urban areas).
4. The average distance between houses along a street depends on the surrounding landuse (houses in residential areas are closer than in industrial areas).
5. The average offset of the first house along a street to the start point of the street correlates with the street's length.

5 Implementation

This chapter describes the implementation of the geocoder, based on the fundamental concepts and technologies presented in chapters 2 and 3 and according to the guidelines developed in chapter 4.

5.1 Development Environment

The geocoder is an essential part of *OpenRouteService* and the preliminary implementation described in section 3.3.3 had already been in production when the work presented in this thesis had begun. The development has thus been shifted to a completely separated development environment consisting of...

- A SuSE Novell™ Linux Enterprise Server 10 running on a VMware™ Virtual Machine platform (2.26 GHz CPU / 2 GB RAM)
- A PostgreSQL 8.3.3 database management system including PostGIS 1.3.3 spatial extension

The outcomes (software and data) of the development have been periodically integrated into OpenRouteService and tested for both function and performance.

5.2 Building the Reference Dataset

5.2.1 Database

A PostgreSQL DBMS[81] (version 8.3.3) has been installed on a SuSE Novell™ Linux Enterprise Server 10 operating system and the PostGIS extension as of version 1.3.3 has been added. Additionally, a database user `osm` and a database named

[81] see section 3.4

`osm` has been created and support for the PL/pgSQL language and the module `pg_trgm` (providing approximate string matching features) has been added to the database. Listing B.7 on page 102 shows the commands called[82].

5.2.2 Data Model

The analysis of the OSM data model with respect to the suitability for geocoding purposes revealed significant disadvantages. For the reasons shown in section 4.1.2 it is, in its current form, not possible to derive a consistent relational structure from it, which would however be necessary to construct a data model as proposed in section 4.1.3. It has thus been decided to implement a data model, which is, from a scientific data modeling point of view, inconsistent and inherently redundant, but nevertheless suitable for the practical purpose of geocoding.

It was decided to initially implement only the three following data types:

- Municipals (Points)
- Streets (Linestrings)
- House Numbers (Points)

The data types are represented in the PostgreSQL database as tables with each row storing an entity of the corresponding data type. The linkage between those data objects is not enforced in the data model, just as it is not enforced in the OSM data model, but has to be generated dynamically in the matching phase of the geocoding process instead.

The additional data such as administrative borders and postal codes are stored as attributes of the three objects, e.g. the `street` table contains the columns `municipal`, `postcode`, etc. In order to provide referential integrity and to avoid redundancies it would normally be necessary to construct separate tables `postcode` etc. and proper relations between objects using foreign key constructs, but it has been decided to clone the data model used in the OSM data instead, because there *simply are no such relations* in the OSM data, hence they can not be implemented in the derivated database.

Whenever a street or a city, which does not contain information about its assignment to administrative units, is transferred to the reference dataset, administrative units are added from ESRI's data (see section 4.1.2). It is important to mention

[82]irrelevant details of the installation left out for clarity

that this is only supposed to serve as a *temporary workaround* until appropriate data is available from OSM and that the data is *not necessary* for the purpose of geocoding cities, streets and house numbers but solely for the purpose of geocoding using postal codes and administrative borders.

5.2.3 Data Integration

Data integration is the process of generating the reference dataset. As described in section 4.1.4 the process consists of different steps, whose implementation will now be described:

Extraction

The data is extracted from a *planet file* OSM dump in XML format using scripts written in the *Perl* programming language.

Reduction

Only those objects referring to municipals, streets or house numbers are filtered out.

Cleaning

The data is cleaned in different ways, e.g. multiple blanks are trimmed to a single blank, punctuation characters are removed, etc.

Standardization

The data is standardized, i.e. multiple spellings of the same logical meaning are standardized to a common variant. As an example, every occurance of the string "str." at the end of a word is expanded to "straße" and so on.

Aggregation

In order to assign cities and streets to their administrative units such as postal codes, country subdivisions etc., data from the ESRI dataset is being added to the objects.

Semantic transformation

The data is semantically transformed in order to fit to the logical database schema embodied in the reference dataset, i.e. a valid database objects in DML format are constructed. This may as well include geometric transformations, e.g. the transformation of polygons or linestrings (buildings, interpolation lines, etc.) to point objects referencing house number locations.

Loading

The data is finally loaded into the reference dataset by connecting to the database and inserting the data using appropriate `INSERT`-commands.

Figure A.1 on page 86 summarizes the whole process of data integration as implemented[83].

5.3 Programming the Interfaces

The framework for the geocoder, i.e. the HTTP/XML interface for client interaction and the whole *OpenLS* implementation, had already been implemented by Neis (2008) and has not been changed. The processing of the use cases however has been created from scratch.

The three use cases for the geocoder (as described in section 4.3) have each been implemented as a database function using the PostgreSQL procedural database language *PL/pgSQL*. The decision for a database procedural language in favour of a standard programming language like *C* or *Java* was primarily based on performance advantages, since early prototypes using *Java* were incapable of providing acceptable response times.

Figure 5.1 on the next page illustrates those parts of the geocoder which had already been implemented, as opposed to the parts representing the work shown in this thesis. The three use cases are represented by the PL/pgSQL functions `freetext_search()`, `struct_search()` and `reverse_geocode()`.

5.3.1 Geocode Result Format

The three functions should return a standard geocode resultset as defined in section 4.3. To accomplish this, a special data type `geocode_result` has been created in the database, which is returned by each function. Listing 5.1 shows the SQL code used to create this data type.

[83]cleaning and standardization steps left out for clarity

5.3.2 Freetext Search

The function `freetext_search()` is given an arbitrary formatted text as a parameter and returns a list of geocode results. Figure A.3 on page 88 shows the (simplified) flow chart describing the internal processing programmed into this function.

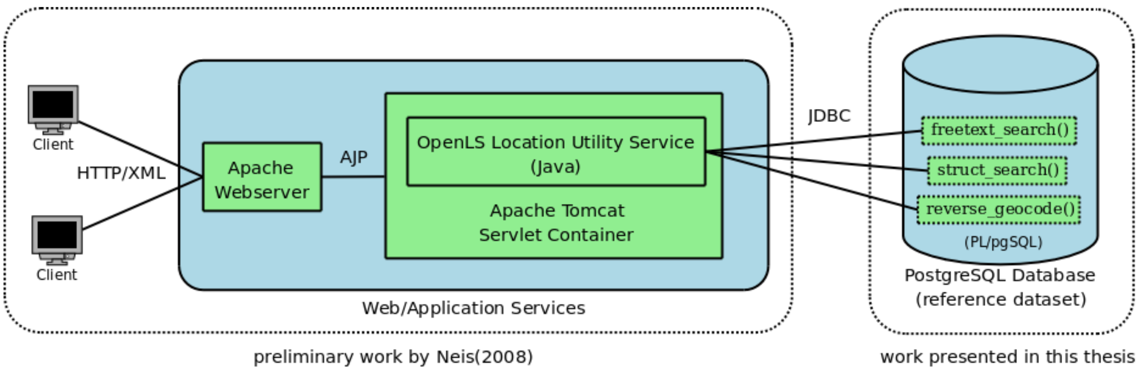

Client
HTTP/XML
Client
Apache Webserver
AJP
OpenLS Location Utility Service (Java)
Apache Tomcat Servlet Container
Web/Application Services
JDBC
freetext_search()
struct_search()
reverse_geocode()
(PL/pgSQL)
PostgreSQL Database (reference dataset)
preliminary work by Neis(2008)
work presented in this thesis

Figure 5.1: OpenRouteService Geocoder Architecture
Source: own assembly

```
CREATE TYPE geocode_result AS
(
  id int,
  countrycode varchar,
  countrys varchar,
  postalcode varchar,
  municipal varchar,
  municipals varchar,
  strname varchar,
  housenr varchar,
  point varchar,
  type int,
  the_geom geometry,
  geocode_quality decimal(3,2)
);
```

Listing 5.1: SQL: creating the geocode result data type

5.3.2.1 Cleaning and Segmenting

For the *cleaning* and *lexical analysis* multiple *regular expressions*[84] have been constructed. Listing 5.2 shows an example of this technique used to extract a house number into the variable `v_housenumber` from the search text stored in variable `v_freetext`.

```
IF v_freetext ~ E'\\m[0-9]{1,3}[a-zA-Z]?\\M'
THEN
    v_housenumber := substring(v_freetext, E'\\m[0-9]{1,3}[a-zA-
       Z]?\\M');
    v_freetext := regexp_replace(v_freetext,v_housenumber,'');
    v_housenumber := regexp_replace(v_housenumber,E'[^0-9]','','
       g');
END IF;
```

Listing 5.2: Extracting house numbers using regular expressions

5.3.2.2 Determining Street/City Relations

As discussed earlier, no static relations between streets and cities are enforced in the database schema. It was thus necessary to develop an approach to dynamically determine such relations, e.g. when searching for a street within a specific city. The following algorithm has been developed and implemented to determine these relations:

Find a street named X of a given city Y:

1. Search for streets that have the street name *"X"* in their *strname* column and the name *"Y"* in their *municipal* column.
2. If no match found in step 1: search for all cities that have the name *"Y"* in their *municipal* column
3. For each city found in step 2, search for streets matching name *"X"* within a search radius "d" of the city
4. If no street found in step 3, increase the search radius "d" and repeat step 3 until street found or maximum search radius reached

Figure A.4 on page 89 shows the flow chart of this approach.

[84] regular expressions: a search pattern language

5.3.2.3 Approximate String Matching

Approximate string matching should be used to determine possible matches for cities and streets when no perfect match could be found. In order to implement this requirement, the additional module *pg_trgm* for *PostgreSQL* has been used. This module provides several functions and operators to measure the similarity of two strings "by counting the number of trigrams they share"[85] (PostgreSQL, 2008, p.1807). The similarity is specified by a numerical value ranging from 1 (strings are completely identical) to 0 (strings have no similarity at all). Listing 5.3 shows an example usage of the `similarity()` function provided by this module.

```
select  similarity ('Salzburg','Salzburg')  as example1,
        similarity ('Salzburg','Sallzburg') as example2,
        similarity ('Salzburg','Sallzbur')  as example3,
        similarity ('Salzburg','Sallzbg')   as example4;

 example1 | example2 | example3 | example4
----------+----------+----------+----------
        1 | 0.727273 |      0.5 | 0.307692
```

Listing 5.3: Example of similarity match function

The similarity value returned by this function is used to calculate the parameter *geocode_quality* of the geocode result, i.e. its value gives a hint about the probability of what is being returned is what has been searched for (a geocode_quality of "1" indicates a perfect match). Listing 5.4 on the following page shows an example of two geocode requests returning a perfect match and an approximate match result.

5.3.2.4 Locating the Geocode Result

The result of the freeform or structured text geocode request must contain a point coordinate pair defining the location of the object returned. This is an easy task when the corresponding object in the reference dataset found during the location phase already is a point object, like a city, stored as a point object referencing its

[85]a triagram in this case means a group of three consecutive characters found in a string (PostgreSQL, 2008, p.1807)

centerpoint. It is however a lot more complicated when the object found during the location phase is a street, for the following reasons:

1. A street may consist of multiple unrelated segments.
2. The shape of a street can make it difficult to define a "centerpoint".

Streets consisting of multiple unrelated segments

This problem is due to the inherent weaknesses of the OSM data model described in section 4.1.2. If two segments of a street are not logically joined via a relation object (they hardly ever are) the only relation between those segments is their name tag (storing the street name) and their distance. In order to retrieve the corresponding segments belonging to a given street segment, the following approach has thus been implemented:

1. Form the initial street geometry of just the single segment.
2. Search for additional segments sharing the same street name within a buffer zone of 500m around the street's geometry.
3. If additional segments are found, add them to the street geometry and repeat step 2.

This approach turned out to be very effective in most cases yet two limitations have to be considered:

- If the segments are further away than 500m, they are not recognized as belonging together

```
SELECT '1' as example,municipal,strname,geocode_quality
FROM   freetext_search('Hellbrunnerstraße Salzburg')
UNION
SELECT '2' as example,municipal,strname,geocode_quality
FROM   freetext_search('Hellbrunerstraße Sallzburg');

 example | municipal |      strname      | geocode_quality
---------+-----------+-------------------+-----------------
 1       | Salzburg  | Hellbrunnerstraße |            1.00
 2       | Salzburg  | Hellbrunnerstraße |            0.61
```

Listing 5.4: Usage of geocode_quality parameter in geocode resultset

- If two streets sharing the same name but belonging to different cities are within distance of 500m, they are mistakenly considered as beloning together

Calculating the centerpoint for non-trivial street geometries

If a street consists of a single segment defined by a simple linestring, the centerpoint may be determined by simply calculating the middle of the line, yet if the street consists of multiple segments forming a more complex shape, there is no simple definition of a *center* for this street. The following process has thus been developed as a general rule to retrieve the centerpoint of a street:

1. Calculate the centroid of the street geometry.
2. Calculate the point of the street geometry closest to the centroid and define this as the centerpoint.

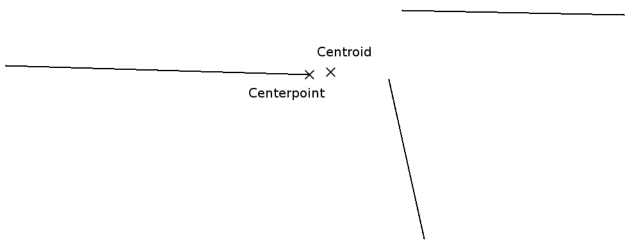
Figure 5.2 on the next page demonstrates this approach on the example of two geometries stored in the reference datatset, each representing a street located in Paderborn, Germany[86].

The process of finding all segments of a street and the calculation of the street's centerpoint has been implemented within a separate PL/pgSQL function called `get_whole_street()`. The code for this function is shown in listing B.6 on page 99.

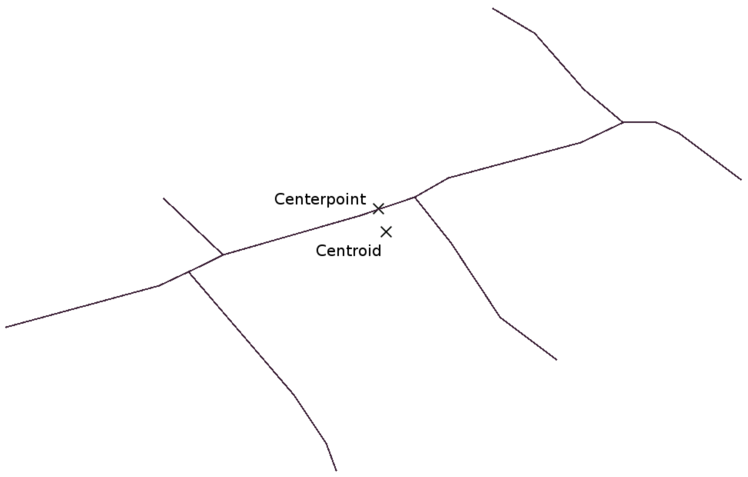
5.3.3 Structured Search

The `structured_search()` function has been implemented similar to the free-text based function with the difference that no segmentation of the search text is needed, because the search terms are given as separate parameters to the function. Listing 5.5 on the next page shows an example call of this function with two search parameters defining the *street* and *municipal* to be searched for.

[86]the first example shows the street *Lichtenauer Weg* and the second one shows the street *Lippeaue*



(a) unconnected segments



(b) complex geometry

Figure 5.2: Retrieving the centerpoint for non-trivial street geometries
Source: own assembly

```
SELECT countrycode,municipal,strname,point
FROM struct_search(NULL,NULL,NULL,'Salzburg',NULL,'
   Hellbrunnerstraße',NULL,NULL);

------------------------------------------------------
countrycode | at
municipal   | Salzburg
strname     | Hellbrunnerstraße
point       | POINT(13.0586889344801 47.7665960489067)
```

Listing 5.5: Example usage of struct_search function

5.3.4 Reverse Geocoding

The function `reverse_geocode()` is given a lat/lon coordinate pair as parameter and returns a corresponding address as a geocode resultset object. Its implementation is much less complex than the two use cases shown before and has been implemented according to three basic steps:

1. Search for the closest house number entry in the reference dataset using a search radius of 20m from the given point.
2. If no match found in step 1 search for the closest street segment using a maximum search radius of 1km.
3. If no match found in step 2 search for the closest city using a maximum search radius of 10km.

Additionally, in this use case the *geocoding_quality* parameter of the geocode resultset is used to inform the user about the actual distance from the point which was searched for and the object returned in the result. Listing B.5 on page 96 shows the complete function as currently used in production. The search radius parameters for house numbers, streets and cities are just given initial values and may be changed later if more appropriate ones can be determined or even dynamically calculated based on spatial analysis of the study area.

5.4 Locating House Numbers

The task of locating house numbers has been implemented using a separate PL/pgSQL function `get_housenumber_of_street()`. It is given two parameters: the id of a street segment and a house number to be searched for. The result is the location of the house number as a point geometry and an information about how this house number has been determined. The function is used by `freetext_search()` and `struct_search()` to determine house number positions during the location phase of the geocoding process.

5.4.1 Exact Match

If the house number which was searched for can be found in the `house numbers` table, its point geometry is retrieved and returned by the function. Listing 5.6

shows an example call of the function returning an exact match for the street "Augartenstraße" in Karlsruhe, Germany (segment nr 1816327 is a part of the street) whereas listing 5.7 shows its usage in combination with the `freetext_search()` function.

```
SELECT AsText(geom) AS point,housenr
FROM get_housenumber_of_street(1816327,46);

            point            | housenr
-----------------------------+---------
 POINT(8.4080637 48.9990288) | 46
```

Listing 5.6: Example usage of function get_housenumber_of_street()

```
SELECT municipal,strname,housenr,point
FROM freetext_search('Augartenstraße 46 Karlsruhe');

---------------------------------------
municipal | Karlsruhe
strname   | Augartenstraße
housenr   | 46
point     | POINT(8.4080637 48.9990288)
```

Listing 5.7: Retrieving exact house number locations with freetext_search()

5.4.2 Interpolation

As defined in section 4.4.3.3, a house number should be interpolated if it lies numerically between two known house numbers of a given street segment. The following algorithm has been implemented to fulfill this requirement:

1. Determine all known house number locations of the street.
2. If searched house number is odd, remove all even numbers and vice versa.
3. Determine closest higher and lower numbers to searched house number.
4. Determine points on street closest to higher and lower number.
5. Interpolate between these two points along the street segment to determine the searched house number location.

6. Return the resulting position of the interpolation and the two house numbers between which interpolation took place.

Figure 5.3 visualizes this process giving the example of the street “Augartenstraße” located in Karlsruhe and listing 5.8 shows an example usage of the function `freetext_search()` returning an interpolated house number location.

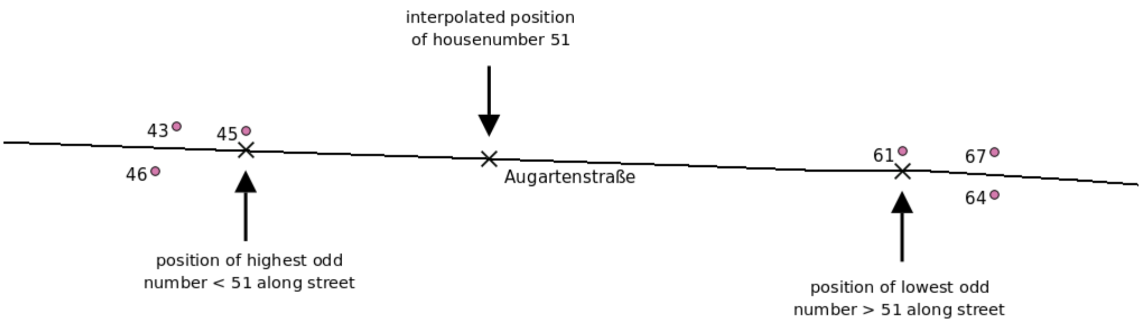


Figure 5.3: Example of house number interpolation between two known house numbers
Source: own assembly

```
SELECT municipal,strname,housenr,point
FROM freetext_search('Augartenstraße 50 Karlsruhe');

-----------------------------------------------------
municipal  | Karlsruhe
strname    | Augartenstraße
housenr    | 46-64
point      | POINT(8.40867975628639 48.9990866918913)
```

Listing 5.8: Retrieving interpolated house number locations using the function freetext_search()

5.4.3 Probability Based Approaches

The hypotheses to determine house numbers by probability based approaches as developed in the analysis phase (see section 4.4.4) have been implemented in function `get_housenumber_of_street()` as well. When no exact match for a house number can be found and interpolation is impossible too, the probability based approaches are used if applicable.

5.4.3.1 Guessing Directions

The first task in guessing the house number location is to determine the direction of the street. This has been implemented using the three different approaches developed in the analysis phase (see section 4.4.4).

Dead-end streets

The following algorithm has been implemented in order to detect dead-end streets:

1. Select number of streets touching the *starting point* of the linestring representing the street.
2. Select number of streets touching the *end point* of the linestring representing the street.
3. If the result of the first step is greater than 0 whereas the result of the second is 0 or vice versa, consider street as being a dead-end street.
4. If we found a potential dead-end street and the *starting point* of its linestring touches no other street *reverse* current direction of street.

Streets proceeding away from city center

In order to detect streets proceeding away from the city center, the following algorithm has been developed and implemented:

1. Locate centerpoint of nearest city.
2. Construct a triangle defined by the corner points A, B and C where…

A	is the end point of the street closer to the city center.

B	is the other end point of the street.
C	is the city center.

3. Calculate the lengths of the sides a, b and c.
4. Calculate the degree of arc β using $\beta = arccos(\frac{a^2+c^2-b^2}{2ac})$

The degree of arc β can now be used as a measure of the relative direction to the city center. Figure 5.4 visualizes this approach giving the example of two streets located in Paderborn, Germany.

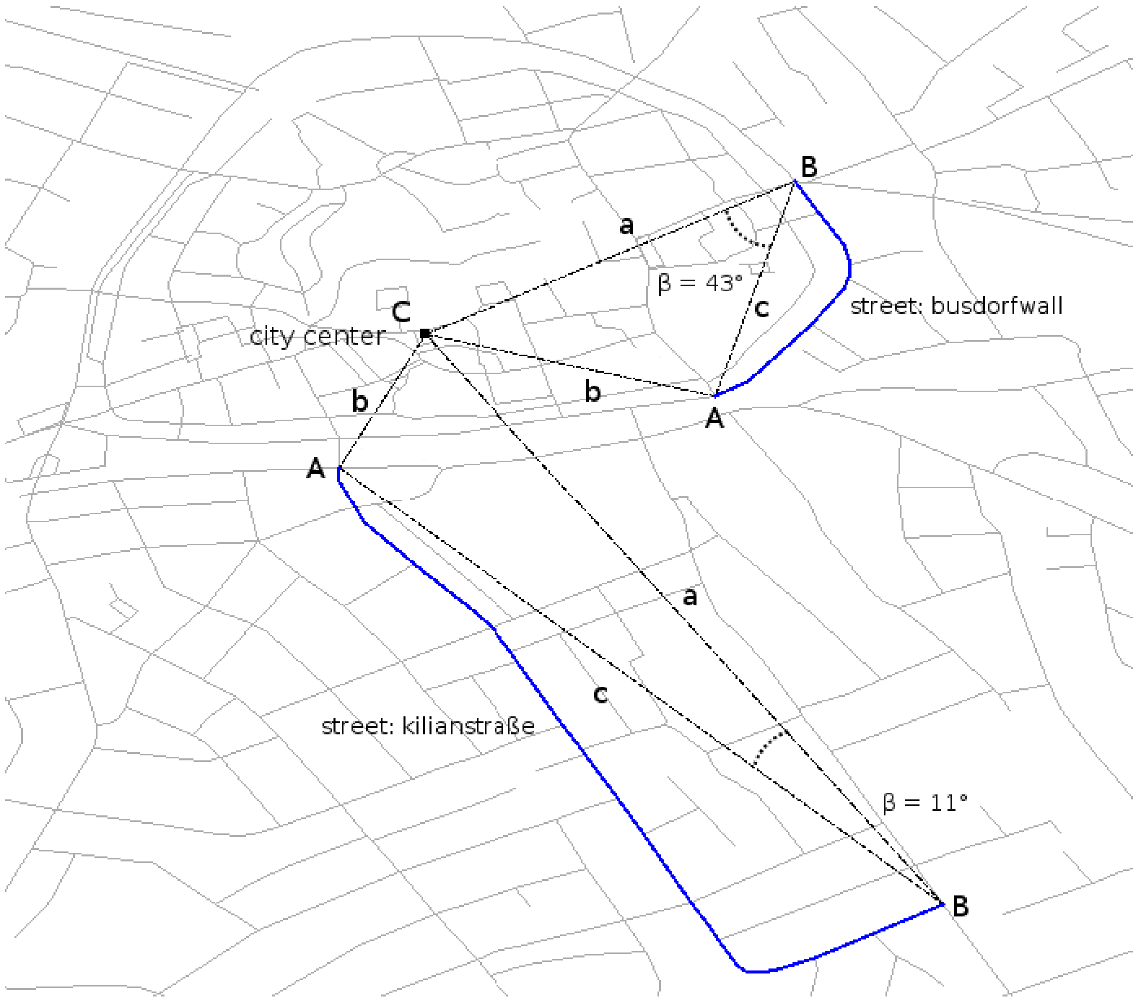


Figure 5.4: Determining street direction related to city center
Source: own assembly

The actual threshold value for β, up to which a street should be considered proceeding radial to the city center, is not easy to determine. Different values have been tested for their suitability and a value of 45 degrees produced good results in most cases (as will be shown in the evaluation phase later). Additionally, the algorithm has been limited to streets no further than 3km away from the city center.

Streets of different ranks

Within the OSM data model, the rank of a street is defined by using the `highway`-tag. In order to compare these ranks they are first transformed to numerical values according to table 5.1[87].

highway tag value	integer value for comparison
motorway	70
trunk	60
primary	50
secondary	40
tertiary	30
residential	20
living_street	10

Table 5.1: Transforming street ranks to integer values for comparison

The following algorithm has then been implemented in order to detect streets, which are connected to *exactly one* other street on boths ends, but where the *ranks* of the connected streets differ:

1. Select number of streets touching the *starting point* of the linestring representing the street.
2. Select number of streets touching the *end point* of the linestring representing the street.
3. If both ends are connected to exactly one street retrieve the numerical value of the ranks of theses streets.
4. If ranks differ order the street so that it starts at the point connected to the street with a higher rank.

[87] ordering taken from http://wiki.openstreetmap.org/wiki/Map_Features#Highway (last accessed: April 14th, 2009)

5.4.3.2 Guessing Distances

If the presumable direction of a street can be estimated by one of the three approaches described before, the position of the searched house number may be interpolated along the street. As shown in section 4.4.4.1, the approximate position p of house number x along a given street can be calculated as…

$$p = o + (x + x \pmod 2 - 2) * d$$

…with d being the average distance between house numbers and o being the position of the first house along the street.

Average distance between house numbers

The average distance between house numbers in the survey area has been calculated using data from the official surveying office, containing the positions of more than 2,000,000 buildings of which a sample set of 25,670 buildings has been used for the calculation. The average distance between two house numbers has been calculated as 6.65 meters[88].

Additionally, the following hypotheses have been tested[89]:

1. The average distance between house numbers correlates with the street's distance to the city center ("buildings in the city center are closer than in suburbian areas").
2. The average distance between house numbers correlates with the length of the street ("the longer the street, the bigger the gaps between single buildings").
3. The average distance between house numbers correlates with the number of sorrounding streets (bigger lots in less urban areas).
4. The average distance of houses along a street depends on the sorrounding landuse (houses in residential areas are closer than in industrial areas).

The first three hypotheses have statistically been tested by calculating the *Pearson product-moment correlation coefficient* for each of the relations using a set of 25,670

[88] It is important to keep in mind that a *sequential alternating* addressing scheme (see section 4.4.2) is assumed for the survey area, i.e. the average distance between two *buildings* is actually twice the average distance between house numbers.

[89] as developed in section 4.4.4.2

house number pairs[90]. The correlation coefficient for the first hypothesis was calculated as 0.07 whereas it was 0.18 for the second hypothesis.

In order to test the third hypothesis, the number of surrounding streets within a distance of 100m has been calculated for each street and this value has been tested for correlations to the average distance between house numbers. The correlation coefficient was calculated as 0.04 hence this hypothesis has been discarded.

The fourth hypothesis has been tested by grouping house number distances by their corresponding landuse areas *residential*, *commercial* and *industrial*, which are (sporadically) available in OSM as polygon data. Table 5.2 shows the results of this test, indeed revealing significant differences in house number distances for industrial and commercial areas. Nevertheless, due to the very low number of available polygons referencing those areas (hence the low sample size) it has been decided not to include this parameter in the approximation.

landuse	sample size	average distance between house numbers	
all	25670	6.65m	
commercial	132	9.13m	
industrial	222	12.48m	
residential	10853	6.55m	

Table 5.2: Relation between landuse and house number distance

It was instead decided to further investigate the second hypothesis (showing the highest correlation coefficient) by visually analyzing the correlation. Figure 5.5 on the following page shows a chart displaying the average distance of house numbers for ranges of street lengths in steps of 100m, which supports the assumption that the distance of house numbers positively correlates with the length of the street the house numbers belong to. Table 5.3 on the next page shows the actual values used in this chart and the number of measures for each group.

Considering this result, it was decided to determine the average distance dynamically. With l being the length of the street and d being the average distance between house numbers, the following algorithm has been implemented:

- If l is less than 100m, d is set to 4.1
- If l is greater than 400m, d is set to 7.9

[90]"house number pair" meaning two known sequential house numbers (e.g. 2/4 or 13/15, etc.) along a given street

- If l is between 100 and 400m, d is calculated using the linear interpolation $d = 4.1 + (7.9 - 4.1) * \frac{l-100}{400-100}$

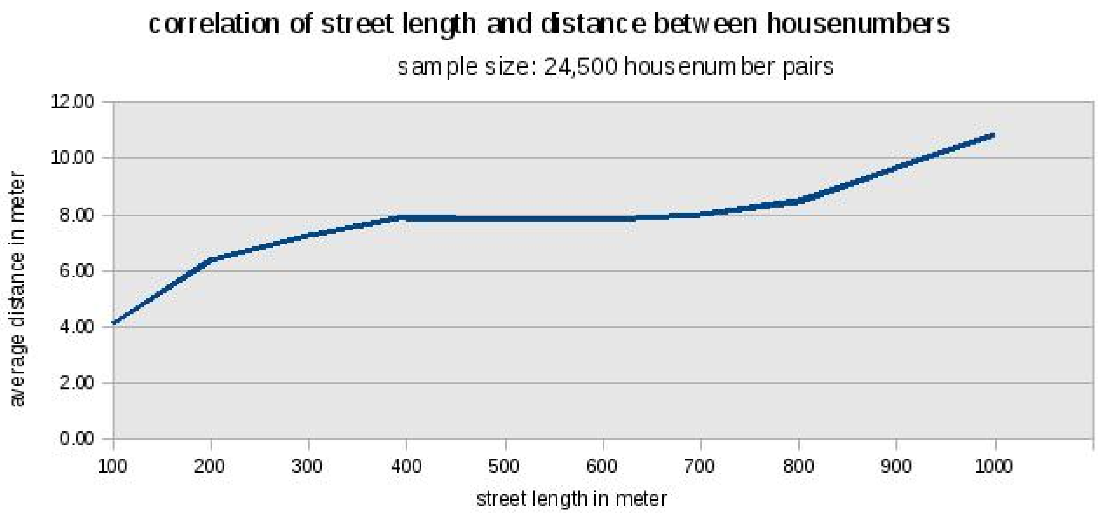


Figure 5.5: Correlation between street length and house number distance
Source: own assembly

street length (maximum)	measures	average house number distance (meters)
100	5516	4.12
200	7870	6.40
300	4586	7.26
400	2478	7.92
500	1518	7.81
600	959	7.83
700	669	7.99
800	437	8.47
900	313	9.69
1000	239	10.87

Table 5.3: Correlation between street length and house number distance

Position of the first house number on the street

To determine the average offset of the first house along a street to the street's starting point, the following algorithm has been implemented:

1. Select all known locations referencing a house number of 1 or 2.
2. Determine the distance to the starting point of their associated streets.
3. Calculate the average offset for the location of a house number to the street's starting point.

The average offset has been calculated as 50.31 meters using a sample set of 3545 house number locations. In order to improve the approximation, a hypothesis has been constructed, assuming that the offset may correlate with the length of the street. A calculation of the correlation coefficient between the street's length and the offset of the first house number along the street has revealed a value of 0.65, indicating a considerably high correlation, hence supporting the hypothesis. Further investigation by visual analysis confirmed this result as shown in figure 5.6.

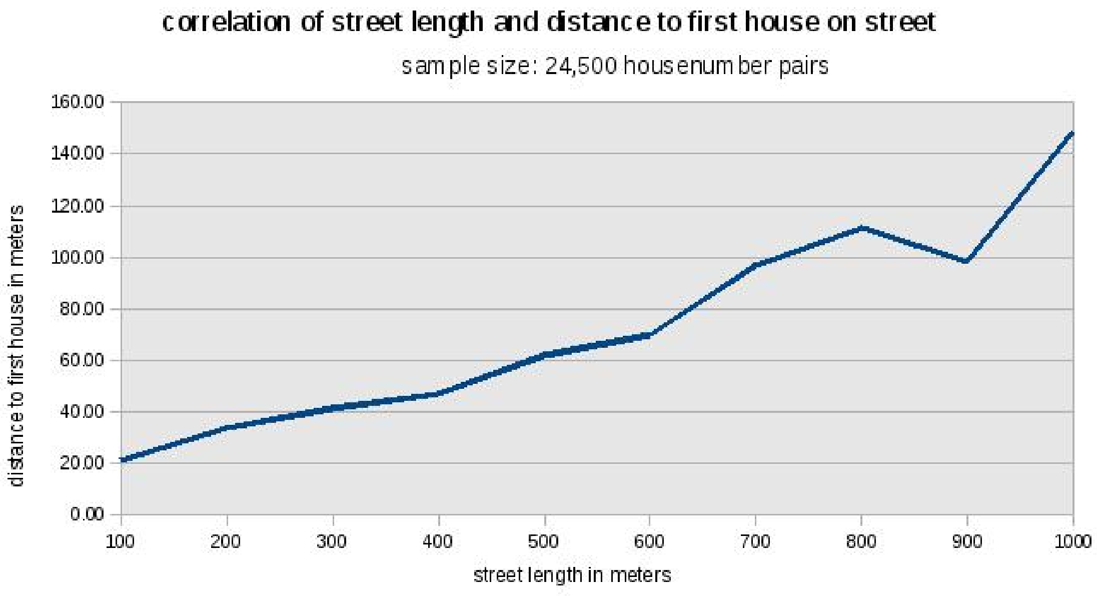


Figure 5.6: Correlation between street length and corner lot offset
Source: own assembly

It has thus been decided to dynamically determine the offset using the following algorithm, with *l* being the length of the street and *o* being the assumed offset for the first house number along the street:

- If l is less than 100m, o is set to 20m.
- If l is greater than 800m, o is set to 100m.
- If l is between 100 and 800m, o is calculated using the linear interpolation: $o = 20 + (100 - 20) * \frac{l-100}{800-100}$

6 Evaluation

In this chapter the success of the geocoder implementation will be measured in terms of its geocoding qualities. The quality of a geocoder is generally determined by examing two measurements:

Match Rate:	The *match rate* indicates the percentage of geocoding requests returning a match (Cayo and Talbot, 2003, p.2).
Positional Accuracy:	The *positional accuracy* of the geocoder indicates the deviation from the point, as returned by the geoder, to the *real* position of the address, which has been geocoded.

Both measurements will now be evaluated in order to determine the overall quality and usability of the geocoder and the positional accuracy will as well be compared to the commercial geocoding service offered by Google$^{\text{TM}}$.

6.1 Match Rate

The *Match Rate* is an indicator for both the quality and completeness of the reference dataset and for the implementation of the geocoding process. In order to return a correct match for a geocode request, the place which has been searched for has to be...

- present in the reference dataset
- found during the matching phase of the geocoding process

6.1.1 Importance of the match rate

The importance of the match rate depends on the intended use of the geocoder. When used in routing applications e.g., it basically determines if the start or end point of a routing request can be found and thus determines if the whole request can be processed or not. Hence for the user of the application it simply determines if the service "works" or "does not work". In this case the match rate directly influences the usability and acceptance of the service.

When the geocoder is used instead as a preprocessing step to spatially analyze address datasets, the match rate can be of critical importance. Ratcliffe (2001) examined the influence of the geocoding match rate in the context of crime analysis using monte carlo simulations. He found the minimum acceptable hit rate of a geocoder, which still allows to reliably detect spatial patterns in the distribution of crime incidents, to be 85%.

6.1.2 Examining the match rate

In order to measure the match rate of the geocoding implementation presented in this thesis, a sample set of geocoding requests has been sent to the geocoder for three different precision levels: municipal, street and house number. The geocoding results have then been declared as a match, if a response of the same precision level could be returned.

Two different types of sample requests have been used. The first sample set was randomly generated from official surveying data of the federal state of *Northrhine-Westfalia, Germany*, whereas a second sample set was randomly extracted from the protocols of the real geocoding application in production, i.e. it has been tested by replaying geocoding requests which had already been sent to the service.

Municipal Level

In this test only municipal level requests have been sent to the geocoder. Table 6.1 shows the results of the test.

Test Data	No. of Requests	Matches	Match Rate in %
Original Queries	100	85	85.00
Random Sample	334	289	86.53

Table 6.1: Match rate of municipal level requests

Street Level

In the next step, street level requests have been sent to the geocoder, again using both randomly generated addresses and replaying real requests extracted from the protocols. Table 6.2 shows the results of this test.

Test Data	Requests	Matches	Match Rate in %
Original Queries	100	72	72.00
Random Sample	1000	598	59.80

Table 6.2: Match rate of street level requests

House Number Level

The last test was performed using geocode requests including house numbers. The results have been separated into house number locations actually found in the reference dataset (exact match), house number locations interpolated between two known house numbers (interpolated match) and house number locations determined using probability based approaches (probability match). Table 6.3 shows the results of this test. The reason for the significantly higher match rate of original queries may be due to users, who actually mapped a specific house number, and later tried to locate "their" house number by sending a corresponding request to the geocoder.

Test Data	Requests	Matches	Match Rate in %
Original Queries	100	5 (exact match)	5.00
		16 (probability match)	16.00
		0 (interpolated match)	0.00
Random Sample	1000	7 (exact match)	0.70
		142 (probability match)	14.20
		1 (interpolated match)	0.10

Table 6.3: Match rate of house number level requests

6.2 Positional Accuracy of House Number Requests

The match rate of a geocoder merely determines whether a match for a geocode request was found or not, yet it says nothing about the positional accuracy of the geocode result (Grubesic and Murray, 2004). In order to examine the accuracy of the geocoder, a sample set of requests has again been sent to the geocoder and the point locations returned for the requests have been compared to the real locations of the addresses, ascertained by the official surveying agency for the study area, the federal state *North-Rhine Westfalia, Germany*.

The tests have been divided into two categories. The first category determines the accuracy of the geocoding results when existing house number data was available in OSM whereas the second determines the accuracy of the geocoding results when probability based approaches had been used to approximate house number locations.

6.2.1 Overall Average Accuracy

As a first measure, the overall average positional error of the geocoder has been measured and determined to be 174.73m as shown in table 6.4. Nevertheless, this value may be biased by extreme positional errors induced from a small set of geocoding requests. With respect to this, the percentiles of requests which fall within a maximum error range have been determined as well, e.g. showing that 50% of all geocode results include a positional error of less than 91,33m.

Number of sample requests	159,498
Average positional error	174.73m
Standard deviation of positional error	286.12m
Percentiles of requests within a maximum error range	
Percentile	Maximum Positional Error
95%	596.59m
90%	369.12m
80%	220,43m
70%	156,43m
60%	117,58m
50%	91,33m

Table 6.4: Overall average of geocoding accuracy

6.2.2 Accuracy Using Existing Data

In the next step, the accuracy of the geocoder has been tested for house number level requests, for which actual house number data was available in the OSM data. This could either be exact house number matches or interpolated locations using interpolation lines or data received by interpolating along a street between two known house number locations. The measured positional error when using house number data has further been compared to the positional error which *would* have been measured when *deliberately omitting* the house number data available and just returning the centerpoint of the street instead. This has been done in order to measure the improvement of positional accuracy when house number data is available.

For exact house number matches (i.e. a node or building referencing the exact house number was found), the positional error was measured to be 13.19m. Comparing this to the positional error for the same sample requests if no house number data was available shows an impressive improvement of 168.84m for the accuracy of the geocoding as shown in table 6.5.

Sample size	2894
Average positional error	13.19m
Average positional error omitting house number data	182.03m
Improvement of accuracy	168.84m

Table 6.5: Positional accuracy for exact house number matches

The same test has been carried out for house number locations determined using OSM interpolation lines and plain interpolation along a street segment between two known house numbers. Tables 6.6 and 6.7 on the next page show the results of the tests, indicating that both methods are well suitable to significantly improve the geocoding result compared to simply returning the centerpoint of the corresponding street.

Sample size	363
Average positional error	36.81m
Average positional error omitting house number data	225.76m
Improvement of accuracy	188.95m

Table 6.6: Positional accuracy using interpolation between two known house numbers

6.2.3 Accuracy using Probability Based Approaches

The probability based approaches to house number locating as developed in section 4.4.4 and implemented in section 5.4.3 have been evaluated considering their applicability to improve the geocoding results. The main question to be answered has been whether theses approaches are suitable to generate *more accurate* results than simply returning the centerpoint of a street. The results shown in table 6.8 lead to the conclusion that two of the approaches are indeed suitable to significantly improve geocoding accuracy whereas the "streets of different rank" approach has failed to generate an improvement.

Considering these results, the most efficient approach seems to be the one determining the relation of the street to the city center in order to guess the direction of the street. This hypothesis was however based on the assumption that the city of interest *has* a centerpoint from which most of the streets may proceed in a radial direction. It may thus be assumed that this approach is more efficient in cities,

Sample size	1434
Average positional error	40.73m
Average positional error omitting house number data	154.34
Improvement of accuracy	113.62m

Table 6.7: Positional accuracy using interpolation lines

Dead-end streets	
Sample size	23656
Average positional error using probability approach	77.38m
Average positional error returning street centerpoint	106.90m
Improvement of accuracy	29.53m
Streets proceeding away from city center	
Sample size	10423
Average positional error using probability approach	130.34m
Average positional error returning street centerpoint	184.78m
Improvement of accuracy	54.43m
Streets of different ranks	
Sample size	17242
Average positional error using probability approach	109.06m
Average positional error returning street centerpoint	108.62m
Improvement of accuracy	-0.43m

Table 6.8: Positional accuracy of probability based approaches

which actually *have* a distinct center, than in cities, where a distinct center is hard or even impossible to determine.

In order to investigate this assumption, the effectiveness of the approach has been tested for different types of cities within the study area and ranked by the percentage of improvement achieved. The results shown in table 6.9 indeed reveal significant differences between cities, ranging from an improvement of 52% to a deterioration of 57%.

City	Sample Size	Positional error city-center	Positional error street centerpoint	Improvement in percent
Münster	1490	90.21m	186.73m	51,69
Paderborn	814	77.37m	157.85m	50,99
Siegen	1240	115.72m	213.66m	45,84
Essen	1224	113.27m	160.12m	29,26
Düsseldorf	1524	115.67m	154.44m	25,10
Köln	534	111.12m	121.10m	8,24
Gelsenkirchen	1026	210.09m	133.77m	-57,06

Table 6.9: Effectiveness of street relation to city center approach for different cities

Comparing the city generating the best result (Münster) with the one generating the worst (Gelsenkirchen), it becomes clear that the layout of the city is of critical importance to the success of the approach. Looking at figure A.5 on page 90, showing the detection of streets radial to the city center of Münster, it seems perfectly reasonable that the approach generated good results, because the city layout comprises a distinct city center with streets proceeding in radial directions. When comparing this with figure A.6 on page 90, showing the same approach for the city of Gelsenkirchen, it becomes very clear why it generated *no* good results in this case instead. The layout of the city of Gelsenkirchen comprises no obvious city center which is why the city centerpoint in OpenStreetMap was supposedly set to a location which was probably seen as defining the "centroid" of the city's multiple parts.

Nevertheless, when using the city centerpoint coordinates as returned by the *Google Maps API* (Google Maps, 2009) instead, the results are entirely different. Table 6.10 on the next page shows that when using these coordinates as the city centerpoint, the deterioration of 57% could be turned into an improvement of almost 13%. Figure A.7 on page 91 visualizes the effect of the two different city centerpoints on the results of this approach.

Sample Size	Positional error city-center approach	Positional error street centerpoint	Improvement in percent
751	145.27m	166.25m	12.62

Table 6.10: Effectiveness of street relation to city center approach for Gelsenkirchen when using Google Maps city centerpoint

The conclusion to be drawn from these findings is that the *street relation to city center* approach may render significantly improved accuracy on the condition that a suitable centerpoint for the city in question can be determined. Figure 6.1 on the next page visualizes the overall improvement of geocoding accuracy by comparing the positional errors generated by this approach with the positional errors when returning the street centerpoint instead.

Summarizing these results it could be shown that for a sample size of 154,807 house number locations, for which no data was provided by OpenStreetMap (neither exact nor interpolated locations), 23,656 locations could be approximated using the "dead-end street" approach and 10,423 positions could be approximated using the "streets proceeding away from city center" approach. From this it follows that for about 22% of the requests probability based approaches where applicable.

6.2.4 Comparison with Google Maps

In order to judge the accuracy values measured before, it has been decided to compare the results to a common commercial geocoding application. The *Google Maps API* (Google Maps, 2009) by Google™ offers a simple HTTP interface allowing to send and retrieve geocode requests using specially formed URLs. Listing 6.1 shows a sample URL requesting a geocode response for the string "Hellbrunnerstraße 34 Salzburg" in XML format and listing B.8 on page 103 shows the corresponding result, including the lat/lon coordinate pair.

```
http://maps.google.com/maps/geo?q=Hellbrunnerstraße+34+Salzburg&
    output=xml&key=abcdefg
```

Listing 6.1: Sample geocode request to Google Maps API

For all of the different methods to locate house number positions (exact match, interpolation and probability based) a sample set of addresses has been geocoded using both the geocoder implemented in this thesis and the geocoder offered by Google Maps. The results have then again been compared to the real positions of the house numbers and corresponding positional errors have been determined. Table 6.11 shows the results of these comparisons.

Location Method	Sample Size	Mean error of ORS gecoder	Mean error of Google geocoder	Difference Google - ORS
Exact Match	2652	13.01m	40.87m	27.86
Plain interpolation	293	37.46m	37.63m	0.18
Interpolation lines	1018	30.99m	26.44m	-4.54
Dead-end streets	17275	88.86m	40.04m	-48.82
Street relation to city center	7013	135.20m	34.24m	-100.96

Table 6.11: Geocoding accuracy comparison of ORS to Google Maps for different house number locating approaches

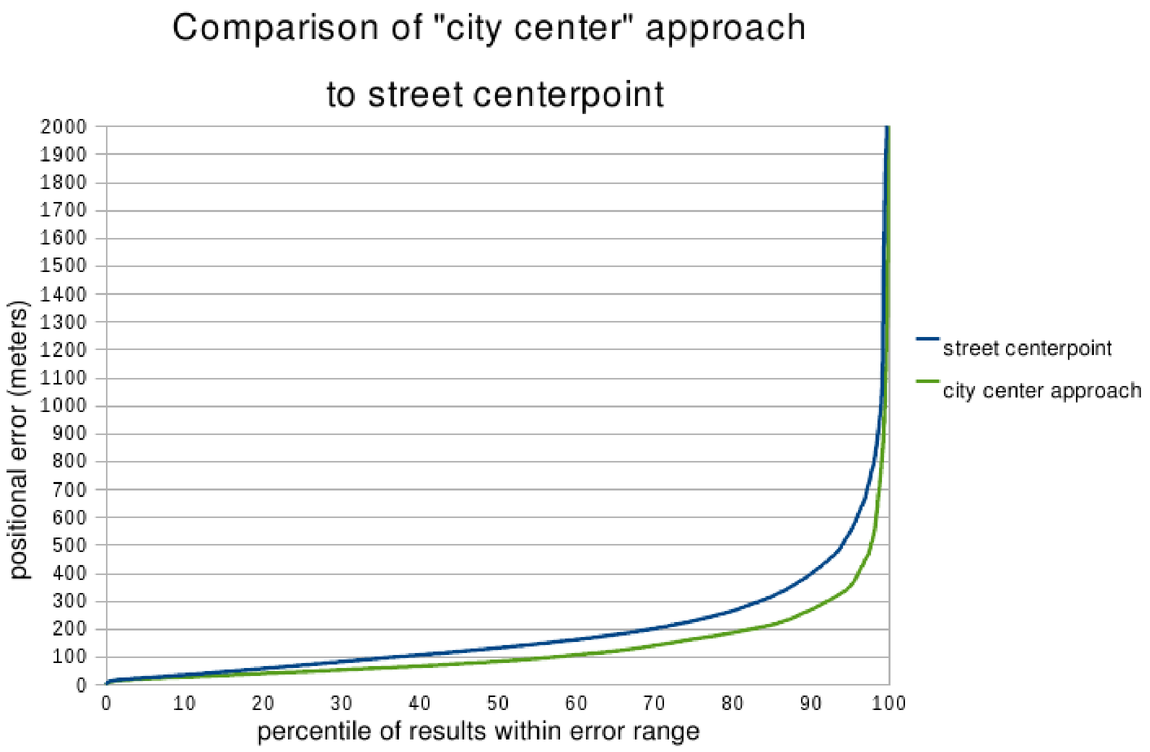


Figure 6.1: Effectiveness of street relation to city center approach
Source: own assembly

Based on these figures the following conclusions can be drawn:

- When exact house numbers locations are availabe in OSM data, the mean positional error of the ORS geocoder is significantly lower than the one offered by Google.
- When interpolation is possible using interpolation lines or by interpolation along street segments between two known house number locations, the mean positional error is not significantly different.
- Probability based approximations of house number locations implemented in the ORS geocoder show a significantly higher mean positional error than Google's geocoder.

Figure 6.2 visualizes the differences in geocoding accuracy between ORS and GoogleMaps when house number data is available in OpenStreetMap.

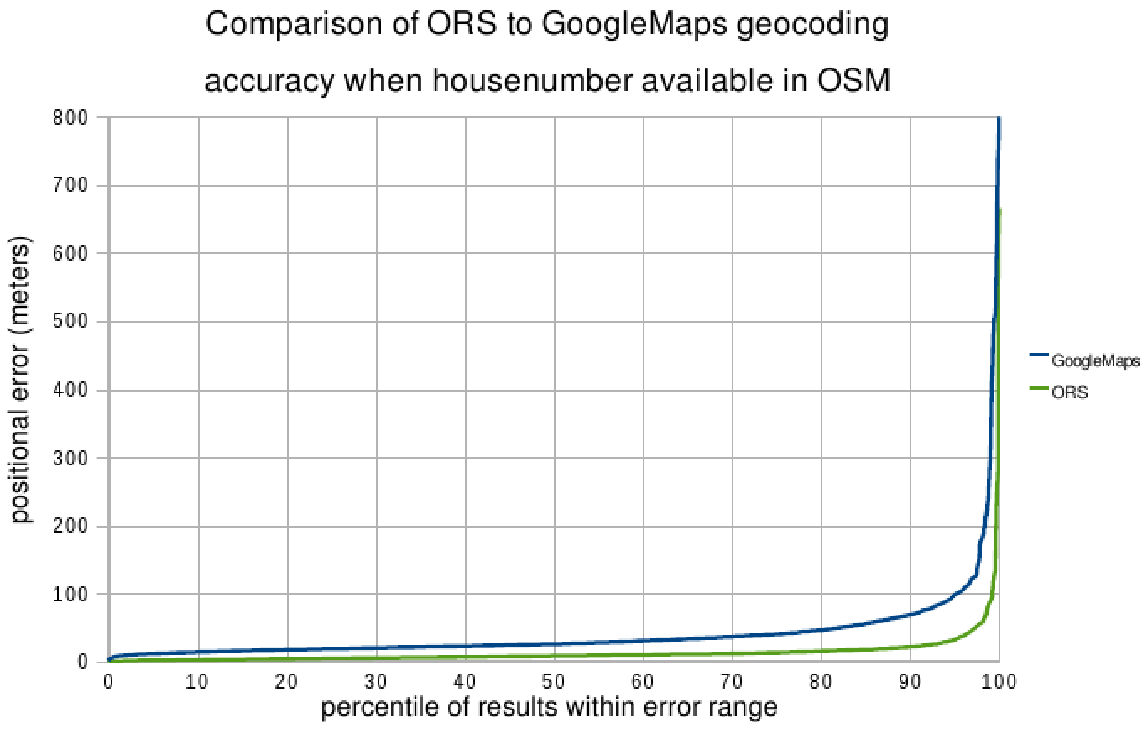


Figure 6.2: Positional errors in OpenRouteService and GoogleMaps geocoding when house number data available in OSM
Source: own assembly

7 Summary

7.1 Conclusions

The key findings of this work and the conclusions to be drawn from them will be presented according to the questions this thesis was supposed to answer[91]:

Is it possible to build a working geocoding service based on the volunteered spatial data provided by the OpenStreetMap project?

The work has shown that it is indeed possible to build a geocoding service based on OpenStreetMap data, although the inherent weaknesses of its data model require substantial preprocessing and transformational steps as well as concessions in terms of relational integrity.

Is it possible to effectively compensate incomplete spatial data (particularly house number locations) using probability based approaches?

It has been demonstrated that it is indeed possible to effectively approximate house number locations using probability based approaches based on hypotheses according to official house numbering guidelines. For the sample set of cities within the study area, an improvement of up to 52% in terms of positional accuracy could be reached. It was nevertheless as well found that the effectiveness of these approaches, although showing a significant overall average improvement, depends heavily on the suitability of the study area. It was further found that these improvements are still not sufficient to generate accuracy levels comparable to cases where actual house number data was available.

[91] as stated in section 1.5

Which match rate and positional accuracy can be achieved by a geocoder based on volunteered spatial data and how do these figures compare to commercial geocoding services?

Match Rate
When considering a match rate of 85% to be the minimum acceptable rate necessary to reliably detect spatial patterns in address datasets (as proposed by Ratcliffe (2001)) it has to be concluded that the achieved match rate on street and house number level[92] is not yet sufficient for spatial analysis purposes. It may however be considered a sufficient initial basis for the OpenRouteService online routing application.

Positional Accuracy
The overall positional accuracy for house number level geocoding requests was measured as being 174,73m, with 95% of the results located within a maximum distance of 597m from their true positions. These figures have to be considered not suitable for fine-scale spatial analyses of address datasets. Zandbergen (2007*a*) e.g. showed that even a medium error of 41 meters with a 90th percentile of just 100 meters can significantly bias analysis results on the example analysis of traffic-related air pollution on school children (using a sample of 104,865 addresses).

The average positional accuracy achieved when interpolation was possible (ranging from 37m to 41m) is nevertheless close to the medium error of 41m measured by Zandbergen (2007*a*) for 104,865 sample addresses located in Orange County, Florida and geocoded using official street centerline and parcel data of the *Property Appraisers Office of Orange County*.

When exact house number data is available in OSM, the measured medium positional error of merely 13m, can be considered to be an extraordinary accuracy. Literature research revealed no case study presenting a geocoding service providing accuracy figures even close (e.g. compare Cayo and Talbot (2003); Dearwent et al. (2001); Mazumdar et al. (2008); Goldberg et al. (2008); Grubesic and Murray (2004); Ratcliffe (2001)).

[92]the match rate for street level requests ranged from 60% to 70% and for house number requests it was merely up to 5% for perfect matches and up to 16% for probability based matches

Google Maps Comparison

The comparison with *Google Maps* showed that when distinct house number data is available in OpenStreetMap, the average positional accuracy of the geocoder implemenmented is significantly better than the accuracy provided by Google (41m vs. 13m). In those cases where house number *interpolation* was possible (either because of interpolation lines or by plain interpolation between two known house numbers) the differences in accuracy are negligible.

Yet for the case when *no house number data* is available, the average positional accuracy proved significantly worse than the one provided by Google Maps. Although the average positional error for these cases may be reduced when probability based house number locating approaches are applicable, the results are still not equivalent to the accuracy provided by Google.

7.2 Outlook

The geocoding service implemented within the scope of this thesis is currently used in production for the geocoding component of OpenRouteService [93] but has already successfully been integrated into other research projects, e.g. *OSM-3D Germany*[94]. The feedback received so far has been very encouraging and further research will presumably be conducted focusing the following aspects:

- Adaptations for different study areas.
- Optimization of probability based approaches by constructing and testing further hypotheses.
- Adaptations of house number locating approaches for different house numbering schemes (opposed to the *alternating sequential* schema used within the study area of this thesis).

The recent development of the OpenStreetMap project is very promising. Within a mere four months during the implementational phase of this thesis, the amount of house number locations for the area of Germany, stored in the OpenStreetMap database, has almost doubled from approximately 172,000 house numbers at the

[93]available at: http://www.openrouteservice.org
[94]available at: http://www.osm-3d.org/

end of December 2008 to more than 330,000 house numbers at the end of April 2009.

Within the same time, the number of requests directed to the OpenRouteService geocoder has steadily increased as well. Figure 7.1 shows the progress of the number of requests per day from December 2008 to April 2009. These figures may indicate that the work presented in this thesis has positively influenced the usability and thus the acceptance of the service. This view is as well supported by the fact that the percentage of requests directed to the geocoder, which included house numbers, has significantly increased since the first prototype of the house number-level geocoder had been published (December 2008) as visualized in figure 7.2.

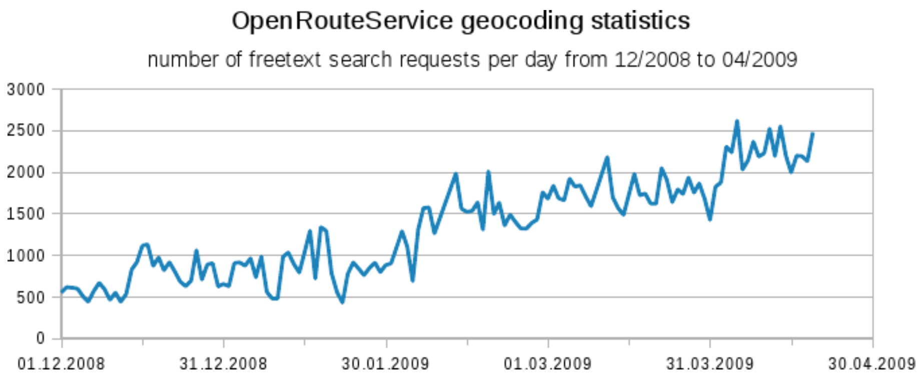


Figure 7.1: Requests sent to the ORS geocoder from 12/2008 to 04/2009
Source: own assembly based on ORS geocoder access protocols

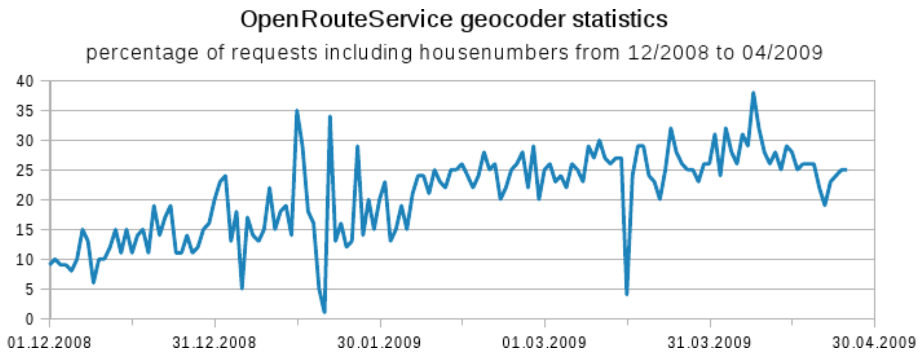


Figure 7.2: Requests including house numbers sent to the ORS geocoder from 12/2008 to 04/2009
Source: own assembly based on ORS geocoder access protocols

A Figures

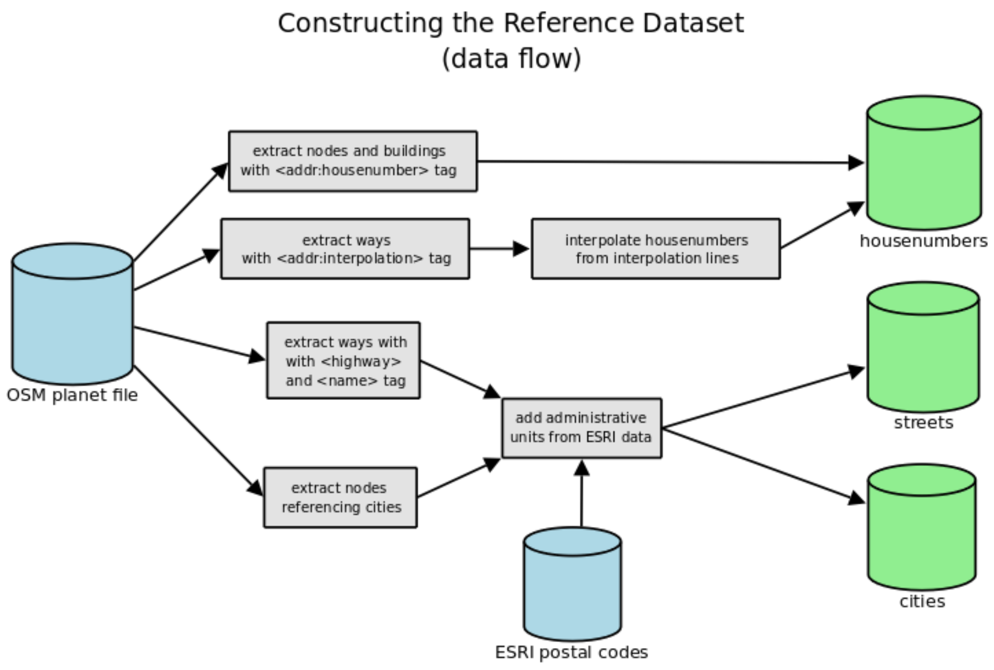
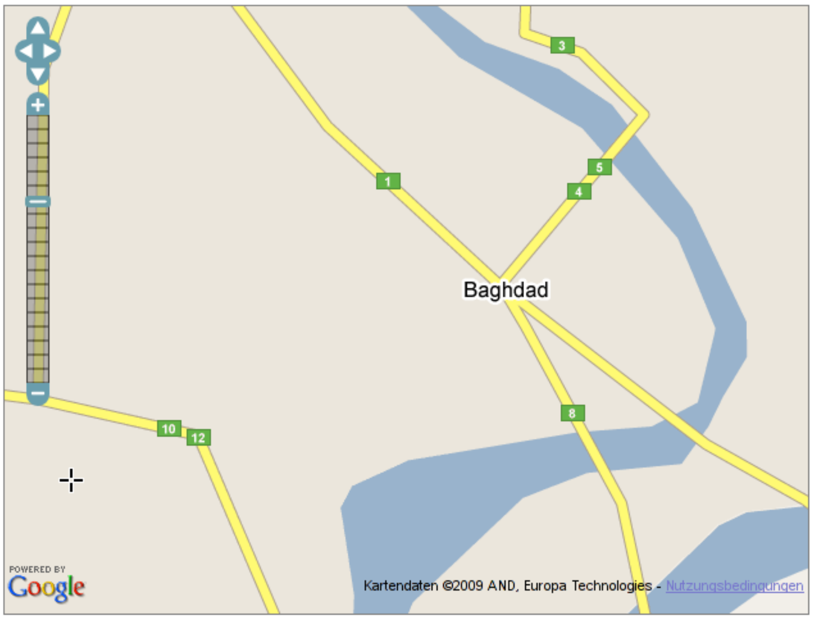
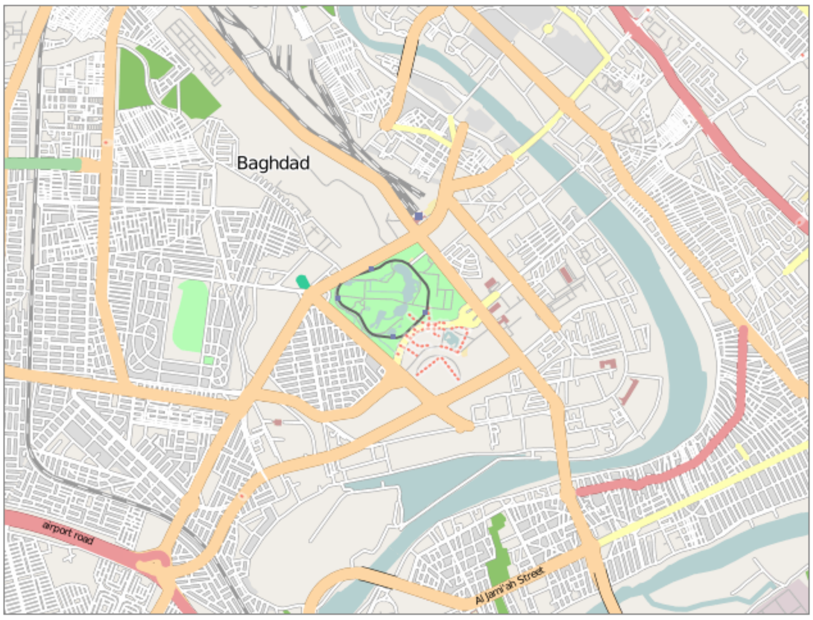

Constructing the Reference Dataset
(data flow)
extract nodes and buildings with <addr:housenumber> tag
extract ways with <addr:interpolation> tag
interpolate housenumbers from interpolation lines
housenumbers
OSM planet file
extract ways with with <highway> and <name> tag
add administrative units from ESRI data
streets
extract nodes referencing cities
cities
ESRI postal codes

Figure A.1: Implementation of the data integration process
Source: own assembly


Baghdad
1
3
4
5
8
10
12
POWERED BY Google
Kartendaten ©2009 AND, Europa Technologies - Nutzungsbedingungen

(a) Google Maps



(b) OpenStreetMap

Figure A.2: City center of Baghdad as shown by Google Maps and OpenStreetMap
Source: screenshots from http://maps.google.com/ and http://www.openstreetmap.org/ taken March 28th, 2009

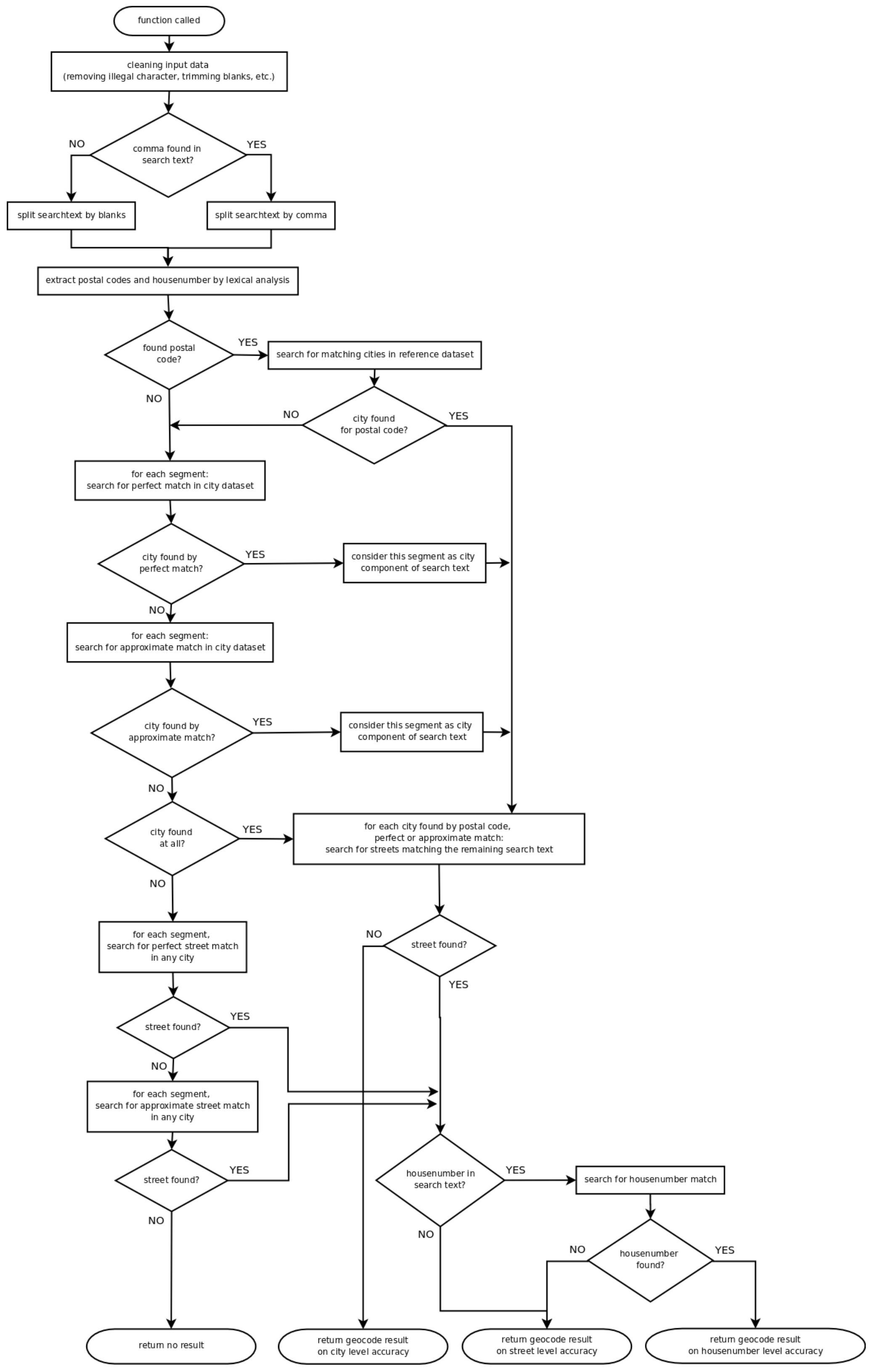

function called
cleaning input data (removing illegal character, trimming blanks, etc.)
comma found in search text?
NO
YES
split searchtext by blanks
split searchtext by comma
extract postal codes and housenumber by lexical analysis
found postal code?
YES
search for matching cities in reference dataset
NO
city found for postal code?
NO
YES
for each segment: search for perfect match in city dataset
city found by perfect match?
YES
consider this segment as city component of search text
NO
for each segment: search for approximate match in city dataset
city found by approximate match?
YES
consider this segment as city component of search text
NO
city found at all?
YES
for each city found by postal code, perfect or approximate match: search for streets matching the remaining search text
NO
for each segment, search for perfect street match in any city
street found?
NO
YES
street found?
YES
NO
for each segment, search for approximate street match in any city
street found?
YES
NO
housenumber in search text?
YES
search for housenumber match
NO
housenumber found?
NO
YES
return no result
return geocode result on city level accuracy
return geocode result on street level accuracy
return geocode result on housenumber level accuracy

Figure A.3: Flow chart for freetext_search function
Source: own assembly

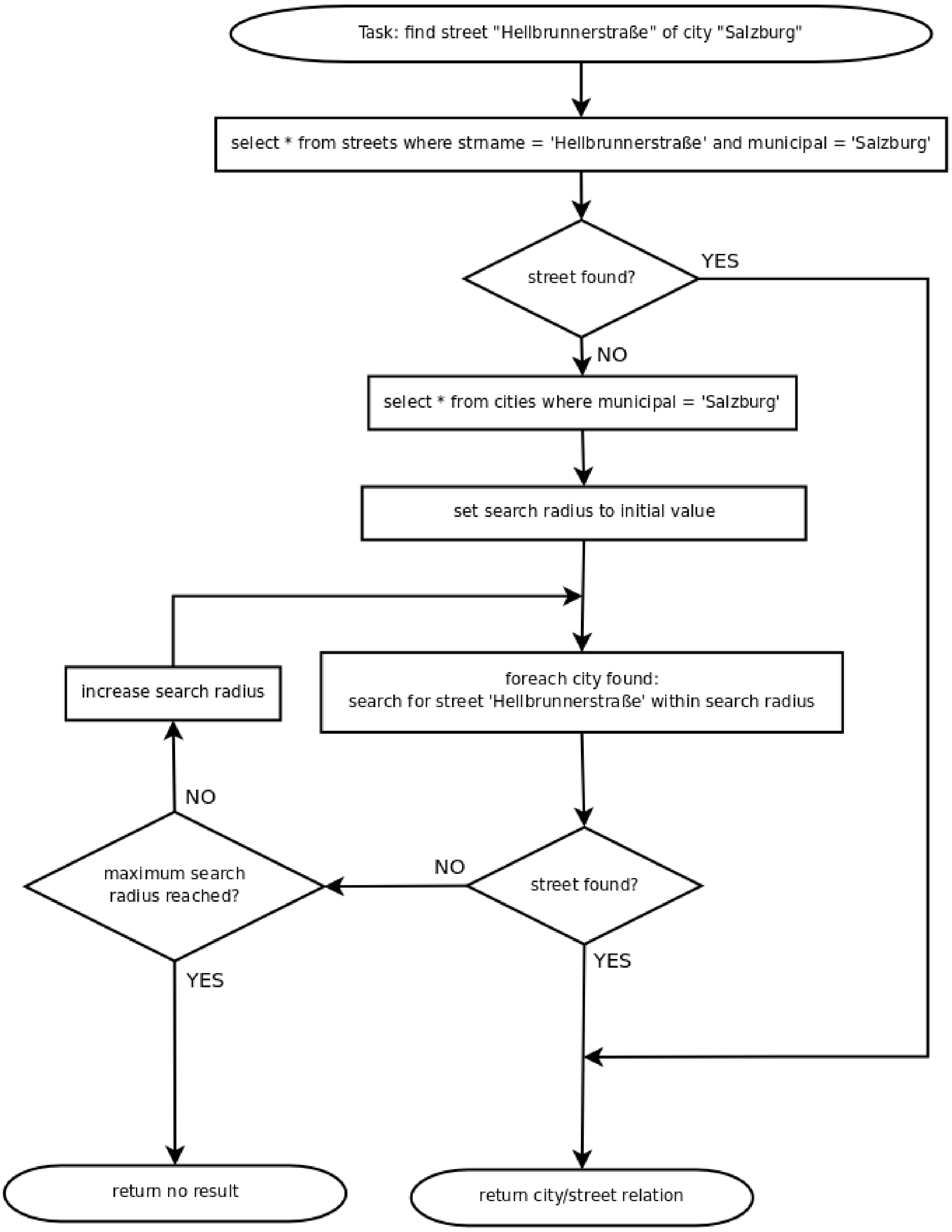

Task: find street "Hellbrunnerstraße" of city "Salzburg"
select * from streets where strname = 'Hellbrunnerstraße' and municipal = 'Salzburg'
street found?
YES
NO
select * from cities where municipal = 'Salzburg'
set search radius to initial value
increase search radius
foreach city found:
search for street 'Hellbrunnerstraße' within search radius
NO
maximum search
radius reached?
NO
street found?
YES
YES
return no result
return city/street relation

Figure A.4: Determining street/city relations (flow chart)
Source: own assembly

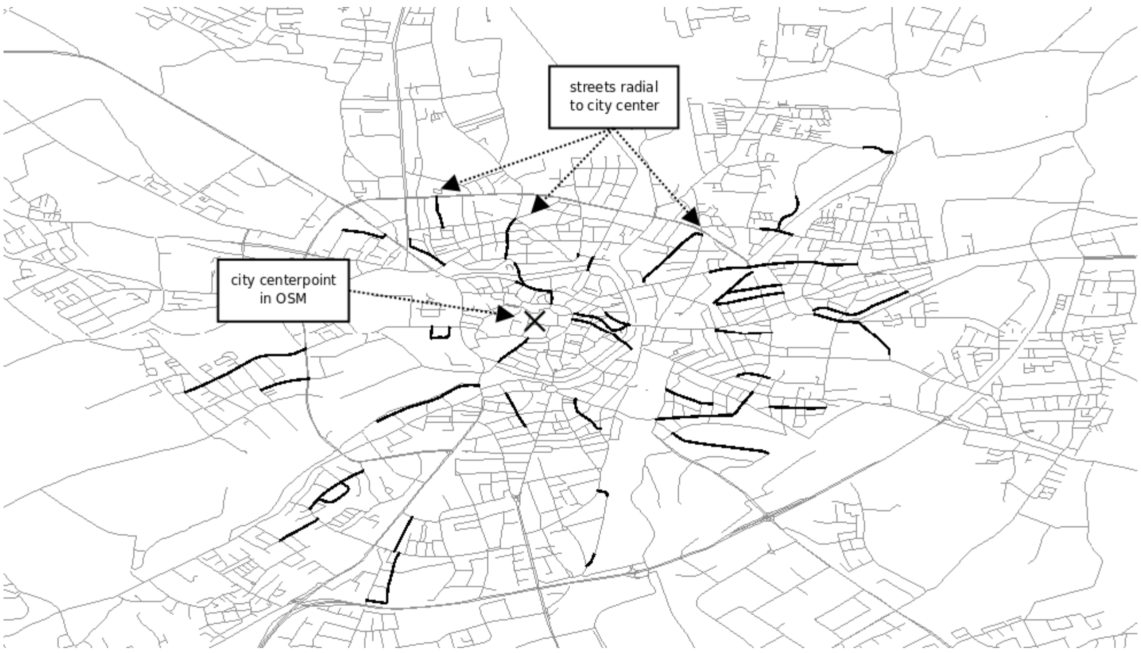


Figure A.5: Determining streets radial to the city center of Münster
Source: own assembly

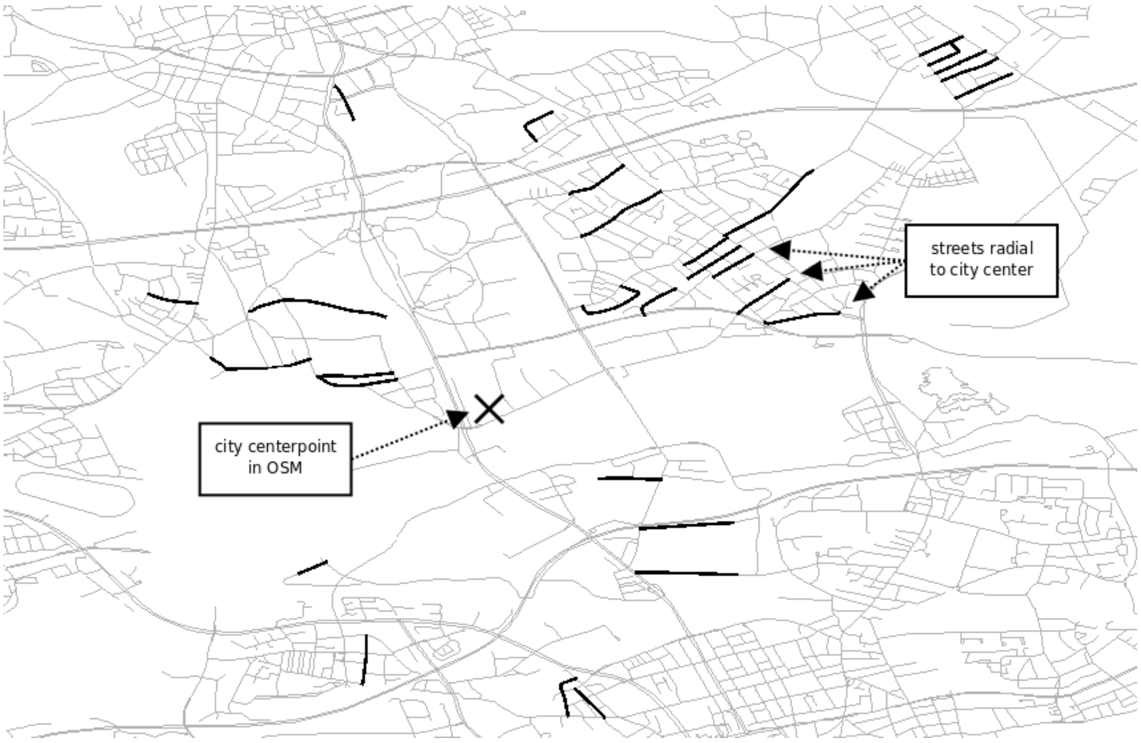


Figure A.6: Determining streets radial to the city center of Gelsenkirchen
Source: own assembly

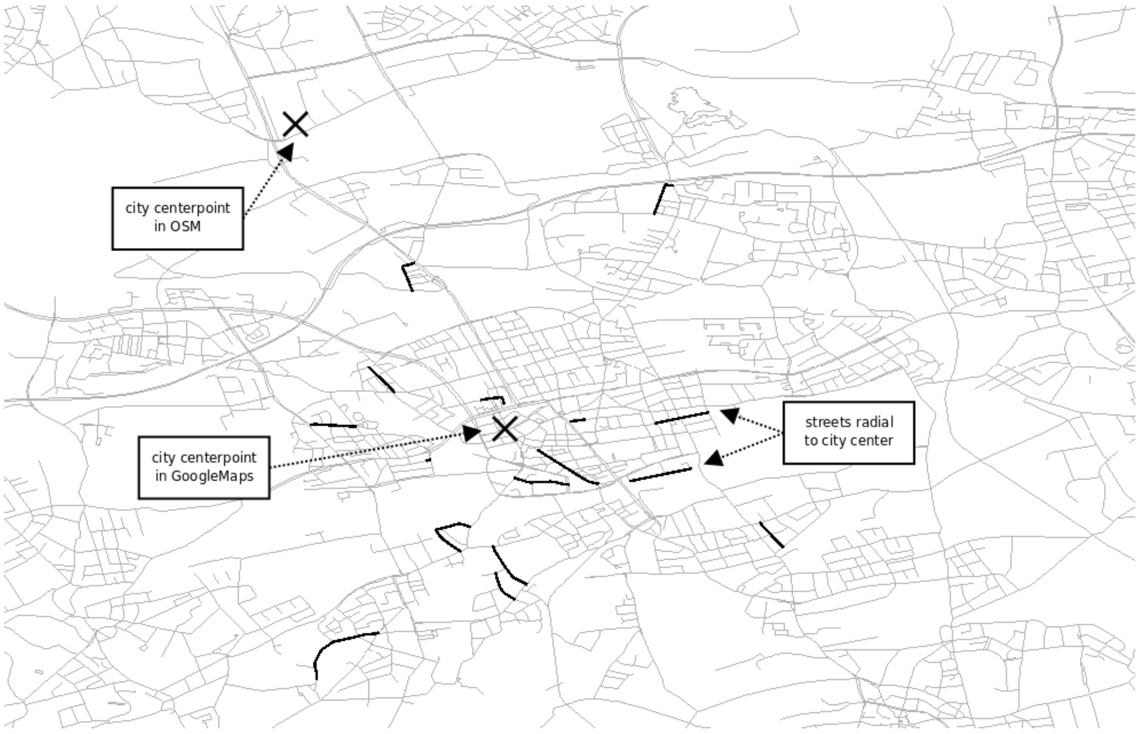


Figure A.7: Determining streets radial to the city center of Gelsenkirchen using Google Maps centerpoint
Source: own assembly

B Listings

```
<?xml version="1.0" encoding="UTF-8"?>
<xls:XLS xmlns:xls="http://www.opengis.net/xls"
         xmlns:sch="http://www.ascc.net/xml/schematron"
         xmlns:gml="http://www.opengis.net/gml"
         xmlns:xlink="http://www.w3.org/1999/xlink"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.opengis.net/xls
         http://schemas.opengis.net/ols/1.1.0/
            LocationUtilityService.xsd"
version="1.1">
  <xls:RequestHeader/>
    <xls:Request methodName="GeocodeRequest" requestID="123"
       version="1.1">
      <xls:GeocodeRequest>
        <xls:Address countryCode="at">
          <xls:freeFormAddress>Hellbrunnerstraße Salzburg</
             xls:freeFormAddress>
        </xls:Address>
   </xls:GeocodeRequest>
  </xls:Request>
</xls:XLS>
```

Listing B.1: Example geocode request using a freeform address

```
<?xml version="1.0" encoding="UTF-8"?>
<xls:XLS xmlns:xls="http://www.opengis.net/xls"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:gml="http://www.opengis.net/gml" version="1.1"
   xsi:schemaLocation="http://www.opengis.net/xls
   http://schemas.opengis.net/ols/1.1.0/LocationUtilityService.
      xsd">
<xls:ResponseHeader xsi:type="xls:ResponseHeaderType"/>
 <xls:Response xsi:type="xls:ResponseType" requestID="123"
       version="1.1" numberOfResponses="1">
  <xls:GeocodeResponse xsi:type="xls:GeocodeResponseType">
    <xls:GeocodeResponseList numberOfGeocodedAddresses="1">
      <xls:GeocodedAddress>
        <gml:Point>
         <gml:pos srsName="EPSG:4326">
              13.0580134145655 47.7659168508311
         </gml:pos>
        </gml:Point>
        <xls:Address countryCode="at">
          <xls:StreetAddress>
           <xls:Street officialName="Hellbrunnerstraße"/>
          </xls:StreetAddress>
          <xls:Place type="Municipality">Salzburg</xls:Place>
        </xls:Address>
        <xls:GeocodeMatchCode accuracy="1.0"/>
      </xls:GeocodedAddress>
     </xls:GeocodeResponseList>
    </xls:GeocodeResponse>
  </xls:Response>
</xls:XLS>
```

Listing B.2: Example geocode response

```
<?xml version="1.0" encoding="UTF-8"?>
<xls:XLS
   xmlns:xls="http://www.opengis.net/xls"
   xmlns:sch="http://www.ascc.net/xml/schematron"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/xls
   http://schemas.opengis.net/ols/1.1.0/LocationUtilityService.
      xsd"
   version="1.1">

 <xls:RequestHeader/>
 <xls:Request methodName="ReverseGeocodeRequest" requestID="123"
     version="1.1">
   <xls:ReverseGeocodeRequest>
     <xls:Position>
       <gml:Point srsName="4326">
        <gml:pos>13.0580134145655 47.7659168508311</gml:pos>
       </gml:Point>
     </xls:Position>
   <xls:ReverseGeocodePreference>StreetAddress</
      xls:ReverseGeocodePreference>
  </xls:ReverseGeocodeRequest>
 </xls:Request>
</xls:XLS>
```

Listing B.3: Example reverse geocode request

```
<?xml version="1.0" encoding="UTF-8"?>
<xls:XLS
   xmlns:xls="http://www.opengis.net/xls"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:gml="http://www.opengis.net/gml" version="1.1"
   xsi:schemaLocation="http://www.opengis.net/xls
   http://schemas.opengis.net/ols/1.1.0/LocationUtilityService.
      xsd">
 <xls:ResponseHeader xsi:type="xls:ResponseHeaderType"/>
 <xls:Response xsi:type="xls:ResponseType" requestID="123"
    version="1.1" numberOfResponses="1">
  <xls:ReverseGeocodeResponse xsi:type="
     xls:ReverseGeocodeResponseType">
   <xls:ReverseGeocodedLocation>
    <gml:Point>
      <gml:pos srsName="EPSG:4326">
        13.058013414565503 47.7659168508311
      </gml:pos>
    </gml:Point>
    <xls:Address countryCode="at">
      <xls:StreetAddress>
        <xls:Street officialName="Hellbrunnerstraße"/>
      </xls:StreetAddress>
      <xls:Place type="Municipality">Salzburg</xls:Place>
    </xls:Address>
    <xls:SearchCentreDistance uom="M" value="0.0"/>
   </xls:ReverseGeocodedLocation>
  </xls:ReverseGeocodeResponse>
 </xls:Response>
</xls:XLS>
```

Listing B.4: Example reverse geocode response

Listing B.5: Reverse geocoding function written in PL/pgSQL

```
CREATE OR REPLACE FUNCTION reverse_geocode (varchar)
RETURNS setof geocode_result AS $$

DECLARE
    v_points text;
    v_point geometry;
    v_resultset geocode_result;
    v_minimum_distance real;
    v_temp_real real;
    v_housenumber varchar;
    v_maximum_distance_to_housenumber_match real;
    v_housenumber_row housenumbers%ROWTYPE;
    v_distance real;
BEGIN

-- first parameter is lat/lon coordinate pair
v_points := $1;

-- maximum distance to consider housenumber as perfect match
v_maximum_distance_to_housenumber_match := 0.0002;

-- construct point geometry from input parameter
v_point := GeomFromText('POINT(' || v_points || ')',4326);

-- check for perfect matching housenumber first
SELECT housenr INTO v_housenumber
FROM (  SELECT housenr,ST_Distance(the_geom,v_point) AS dist
    FROM housenumbers
        WHERE ST_DWithin(the_geom,
             v_point,
             v_maximum_distance_to_housenumber_match
            )
        ORDER BY dist LIMIT 1 ) AS x;

IF v_housenumber IS NOT NULL
THEN
    v_resultset.housenr = v_housenumber;
    v_resultset.geocode_quality := 1 - ST_Distance(v_point,
       v_resultset.the_geom) * 10;
    RETURN NEXT v_resultset;
    RETURN;
END IF;

-- search for street within 1km
-- looping over increasing search radius for performance
FOR v_temp_real IN 0..3
LOOP
    v_minimum_distance := 1 / POWER((5-v_temp_real)*10,2);
    FOR v_resultset IN SELECT id, countrycode,
                  countrys, postalcode,
                  municipal, municipals,
                  strname,'','',
                  ''||v_points AS point,
                  type, the_geom, 1 FROM (
                   SELECT *, asText(the_geom),
                      ST_Distance(the_geom,v_point) AS dist
                   FROM addressbook
                   WHERE ST_DWithin(the_geom,v_point,
                      v_minimum_distance )
                   AND strname IS NOT NULL ORDER BY dist LIMIT 1
                  ) AS x
    LOOP
        v_resultset.geocode_quality := 1 - ST_Distance(v_point,
           v_resultset.the_geom) * 10;
        RETURN NEXT v_resultset;
        RETURN;
    END LOOP;
END LOOP;

-- search for city within distance of 10km if no street found
-- looping over increasing search radius for performance
FOR v_temp_real IN 1..5
LOOP
    v_minimum_distance := v_temp_real * 0.02;
    FOR v_resultset IN SELECT id,countrycode,
                  countrys,postalcode,
                  municipal,municipals,
                  '','','',
                  ''||v_points AS point,
                  type, the_geom, 1 FROM (
                    SELECT *, asText(the_geom),
                       ST_Distance(the_geom,v_point) AS dist
                    FROM addressbook
                    WHERE type < 8
                    AND ST_DWithin(the_geom,v_point,
                       v_minimum_distance )
                    ORDER BY dist LIMIT 1
                  ) AS x
    LOOP
        v_resultset.geocode_quality := 1 - ST_Distance(v_point,
           v_resultset.the_geom) * 10;
        RETURN NEXT v_resultset;
        RETURN;
    END LOOP;
END LOOP;
RETURN;
END;

$$ LANGUAGE plpgsql;
```

Listing B.6: Retrieving all segments of a street and returning the centerpoint

```
DROP TYPE geocode_get_street_result CASCADE;
CREATE TYPE geocode_get_street_result AS
(
  geom geometry,
  centerpoint geometry
);

-- collect all segments of a street identified by a single
   segment of addresbook
CREATE OR REPLACE FUNCTION get_whole_street (v_addressbook_id
   int)
RETURNS geocode_get_street_result AS $$

DECLARE
    v_addressbook_entry addressbook%ROWTYPE;
    v_addressbook_result addressbook%ROWTYPE;
    v_result geocode_get_street_result;
    v_geom geometry;
    v_counter int;
    v_samefields_list text[];
    v_found_street boolean;
    v_used_ids int[];
    v_segments geometry[];
    v_nr_segments int;
    v_centroid geometry;
    v_middle_of_street geometry;
    v_center_segment geometry;
    v_center_segment_linestring geometry;
    v_dist_centroid_segment float;
BEGIN

v_counter = 1;
v_found_street = FALSE;

SELECT * FROM addressbook INTO v_addressbook_entry WHERE id =
   v_addressbook_id;
```

```
v_used_ids := array_append(v_used_ids,v_addressbook_entry.id);
IF v_addressbook_entry IS NULL
THEN
    RAISE EXCEPTION 'ERROR: no addressbook entry for id % found!
       ',v_addressbook_id;
    RETURN NULL;
END IF;

v_geom := v_addressbook_entry.the_geom;
v_segments := array_append(v_segments,v_geom);
LOOP
    FOR v_addressbook_result IN SELECT * FROM addressbook WHERE
    strname = v_addressbook_entry.strname
    AND id <> ALL(v_used_ids)
    AND   ST_DWithin(the_geom,v_geom,0.005)
    LOOP
    v_segments := array_append(v_segments,v_addressbook_result.
       the_geom);
    v_geom = ST_Union(v_geom,v_addressbook_result.the_geom);
    v_used_ids := array_append(v_used_ids,v_addressbook_result.
       id);
    END LOOP;
    IF v_addressbook_result IS NULL THEN EXIT; END IF;
END LOOP;

v_nr_segments := array_upper(v_segments,1);
v_centroid := ST_Centroid(v_geom);
v_dist_centroid_segment := ST_Distance(v_centroid,v_segments[1])
   ;
v_center_segment := v_segments[1];

IF v_nr_segments > 1 THEN
    v_centroid := ST_Centroid(v_geom);
    RAISE NOTICE 'searching for closest segment to centroid...';
    FOR i in 2..array_upper(v_segments,1)
    LOOP
        IF ST_Distance(v_centroid,v_segments[i]) <
           v_dist_centroid_segment
        THEN
            RAISE NOTICE 'found segment closer to centroid: %',i
               ;
            v_dist_centroid_segment := ST_Distance(v_centroid,
               v_segments[2]);
            v_center_segment := v_segments[i];
        END IF;
    END LOOP;
    v_center_segment_linestring := ST_LineMerge(v_center_segment
       );
    v_middle_of_street := ST_line_interpolate_point(
       v_center_segment_linestring,ST_line_locate_point(
       v_center_segment_linestring,v_centroid));
ELSE
    v_middle_of_street := ST_Centroid(v_geom);
END IF;

v_result.geom := v_geom;
v_result.centerpoint := v_middle_of_street;
RETURN v_result;

END;
$$ LANGUAGE plpgsql;
```

```
# create osm database user
createuser -s osm

# create osm database
createdb --owner=osm --encoding=utf8 osm

# add pl/pgsql language
createlang plpgsql osm

# add trgm module
psql -f pg_trgm.sql osm

# add postgis extension
psql -f lwpostgis.sql osm

# add spatial reference systems
psql -f spatial_ref_sys.sql osm
```

Listing B.7: UNIX commands to create the database

```
<?xml version="1.0" encoding="UTF-8" ?>
<kml xmlns="http://earth.google.com/kml/2.0"><Response>
  <name>Hellbrunnerstraße 34 Salzburg</name>
  <Status>
    <code>200</code>
    <request>geocode</request>
  </Status>
  <Placemark id="p1">
    <address>Hellbrunner Straße 34, 5020 Salzburg, Österreich</
       address>
    <AddressDetails Accuracy="8" xmlns="
       urn:oasis:names:tc:ciq:xsdschema:xAL:2.0"><Country><
       CountryNameCode>AT</CountryNameCode><CountryName>
       Österreich</CountryName><AdministrativeArea><
       AdministrativeAreaName>Salzburg</AdministrativeAreaName><
       SubAdministrativeArea><SubAdministrativeAreaName>Salzburg
        (Stadt)</SubAdministrativeAreaName><Locality><
       LocalityName>Salzburg</LocalityName><Thoroughfare><
       ThoroughfareName>Hellbrunner Straße 34</ThoroughfareName>
       </Thoroughfare><PostalCode><PostalCodeNumber>5020</
       PostalCodeNumber></PostalCode></Locality></
       SubAdministrativeArea></AdministrativeArea></Country></
       AddressDetails>

    <ExtendedData>
      <LatLonBox north="47.7917662" south="47.7854710" east="
         13.0634045" west="13.0571093" />
    </ExtendedData>
    <Point><coordinates>13.0602569,47.7886186,0</coordinates></
       Point>
  </Placemark>
</Response></kml>
```

Listing B.8: Google Maps geocode result in XML format

C Bibliography

Bakshi, R., Knoblock, C. A. and Thakkar, S. (2004), Exploiting online sources to accurately geocode addresses., *in* 'Proceedings of the 12th annual ACM international workshop on Geographic information systems', ACM, Washington DC, USA, pp. 194–203.

Behr, F.-J., Rimayanti, A. and Li, H. (2008), Opengeocoding.org - a free, participatory, community oriented geocoding service, Technical report, Department of Geomatics, Computer Science and Mathematics, University of Applied Sciences Stuttgart, Stuttgart, Germany.

Borkar, V. R., Deshmukh, K. and Sarawagi, S. (2001), Automatic segmentation of text into structured records, *in* 'Proceedings of the SIGMOD Conference', Santa Barbara, California.

Brownstein, J. S., Cassa, C., Kohane, I. S. and Mandl, K. D. (2005), Reverse geocoding: Concerns about patient confidentiality in the display of geospatial health data, *in* 'Proceedings of the Annual Symposium', Amercian Medical Informatics Association, Washington, DC, USA.

Cayo, M. R. and Talbot, T. O. (2003), 'Positional error in automated geocoding of residential addresses', *International Journal of Health Geographics* **2:10**.

Christen, P. (2006), A comparison of personal name matching: Techniques and practical issues, *in* 'Proceedings of the Sixth IEEE International Conference on Data Mining', Hong Kong, pp. 290–294.

Christen, P. and Churches, T. (2005), A probabilistic deduplication, record linkage and geocoding system, *in* 'Proceedings of the ARC Health Data Mining workshop', Adelaide, Australia.

Christen, P. and Zhu, J. X. (2002), Probabilistic name and address cleaning and standardisation, *in* 'Proceedings of The Australasian Data Mining Workshop', Canberra, Australia.

Creative Commons (2009), 'Attribution-share alike license 2.0'. Last accessed: March 27th, 2009.
URL: *http://creativecommons.org/licenses/by-sa/2.0/legalcode*

Davis, C., Fonseca, F. and Borges, K. A. V. (2003), A flexible addressing system for approximate geocoding, *in* 'Brazilian Symposium on GeoInformatics'.

Dearwent, S., Jacobs, R. and Halbert, J. (2001), 'Locational uncertainty in georeferencing public health datasets', *Journal of Exposure Analysis and Environmental Epidemiology* **11**, 329–334.

ESRI (2005), 'An overview of esri data and maps, release 9.1'. Last accessed: April 24th, 2009.
URL: *http://webhelp.esri.com/arcgisdesktop/9.1/index.cfm?ID=2038&TopicName=An%20overview%20of%20ESRI%20Data%20and%20Maps*

ESRI (2008), 'Introducing arcsde implementation for postgresql', Transcript of the Educational Services Podcast series. Last accessed: April 15th, 2009.
URL: *http://www.esri.com/news/podcasts/transcripts/introducingarcsdeimplementationforpostgresql.pdf*

Farnsworth, G. (1970), *The DIME geocoding system*, Census use study, U.S. Bureau of the Census, Washington, DC, USA.

Farvacque-Vitkovic, C., Godin, L., Leroux, H., Verdet, F. and Chavez, R. (2005), *Street Addressing and the Management of Cities*, The International Bank for Reconstruction and Development / The World Bank, Washington, DC, USA.

Fonda-Bonardi, P. (1994), House numbering systems in los angeles, *in* 'GIS/LIS '94 Proceedings', Los Angeles County Urban Research Section, pp. 322–331.

Free Software Foundation (1991), 'Gnu general public license, version 2'. Last accessed: April 15th, 2009.
URL: *http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt*

Goldberg, D. W. (2008), *A Geocoding Best Practices Guide*, University of Southern California, GIS Research Laboratory.

Goldberg, D., Wilson, J., Knoblock, C., Ritz, B. and Cockburn, M. (2008), 'An effective and efficient approach for manually improving geocoded data', *International Journal of Health Geographics* **7**(1), 60.

Goodchild, M. F. (2007*a*), 'Citizens as sensors: the world of volunteered geography', *GeoJournal* **69**(4), 211–221.

Goodchild, M. F. (2007*b*), 'Citizens as sensors: Web 2.0 and the volunteering of geographic information', *Geofocus* **7**, 8–10.

Google Maps (2009), 'Google maps terms and conditions'. Last accessed: March 22nd, 2009.
URL: *http://www.google.com/intl/en_ALL/help/terms_local.html*

Grubesic, T. H. and Murray, A. T. (2004), Assessing positional uncertainty in geocoded data, *in* 'Proceedings of the 24th Urban Data Management Symposium', Chioggia, Italy.

Haklay, M. (2008), How good is openstreetmap information? a comparative study of openstreetmap and ordnance survey datasets for london and the rest of england, Technical report, Department of Civil, Environmental and Geomatic Engineering, UCL.

Han, J. and Kamber, M. (2006), *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management, 2nd edn, Diane Cerra, San Francisco, USA.

Harris, K. (1999), *Mapping Crime: Principle And Practice*, Diane Pub Co.

IETF: The Internet Engineering Task Force (1999), 'Rfc2616: Hypertext transfer protocol – http/1.1'. Last accessed: March 27th, 2009.
URL: *http://www.ietf.org/rfc/rfc2616.txt*

Kim, U. (2001), A historical study on the parcel number and numbering system in korea, *in* 'Proceedings of the Technical Conference during the FIG Working Week, 8-10 May', Seoul, Korea.

Krieger, N., Chen, J. T., Waterman, P. D., Rehkopf, D. H. and Subramanian, S. (2005), 'Painting a truer picture of us socioeconomic and racial/ethnic health inequalities: The public health disparities geocoding project', *American Journal of Public Health* **95**(2), 312–323.

Königlich Hannoversches Ministerium des Inneren (1857), 'Bekanntmachung des königlichen ministeriums des innern, die allgemeine einführung fester hausnummern betreffend'.

Lait, A. and Randell, B. (1996), An assessment of name matching algorithms, Technical report, Departement of Computing Science, University of Newcastle upon Tyle, Newcastle, UK.

Mazumdar, S., Rushton, G., Smith, B., Zimmerman, D. and Donham, K. (2008), 'Geocoding accuracy and the recovery of relationships between environmental exposures and health', *International Journal of Health Geographics* **7**(1), 13.

MLP (2001), 'Mobile location protocol specification'.

Navarro, G. (2001), 'A guided tour to approximate string matching', *ACM Computing Surveys* **33**(1), 31–88.

Neis, P. (2006), Routenplaner für einen emergency route service auf basis der openls spezifikation, Diplomarbeit, University of Applied Sciences Mainz.

Neis, P. (2008), Location based services mit openstreetmap daten, Master's thesis, Fachhochschule Mainz Fachbereich I.

Neis, P. and Zipf, A. (2008), Openrouteservice.org is three times open: Combining opensource, openls and openstreetmaps, *in* 'Proceedings of the GISRUK 2008 conference', UNIGIS UK, Manchester.

OGC (2008), 'Opengis location service (openls) implementation specification: Core services'.

OpenStreetMap (2009), 'The free wiki world map'. Last accessed: March 25th, 2009.
URL: *http://www.openstreetmap.org/*

OpenStreetMapStats (2009), 'Openstreetmap stats report'. Last accessed: March 22nd, 2009.
URL: *http://www.openstreetmap.org/stats/data_stats.html*

OSMProtocol (2009), 'Osm protocol version 0.5'. Last accessed: March 26th.
URL: *http://wiki.openstreetmap.org/wiki/OSM_Protocol_Version_0.5*

Perkins, C. and Dodge, M. (2008), 'The potential of user-generated cartography: a case study of the openstreetmap project and mapchester mapping party', *North West Geography* **8**.

PostgreSQL (2008), *PostgreSQL 8.3.4 Documentation*, The PostgreSQL Global Development Group.

Pouliquen, B., Steinberger, R., Ignat, C. and De Groeve, T. (2004), Geographical information recognition and visualization in texts written in various languages, *in* 'SAC '04: Proceedings of the 2004 ACM symposium on Applied computing', ACM, New York, USA, pp. 1051–1058.

Rahm, E. and Do, H. H. (2000), 'Data cleaning: Problems and current approaches', *IEEE Data Engineering Bulletin* **23**, 2000.

Ratcliffe, J. H. (2001), 'On the accuracy of tiger-type geocoded address data in relation to cadastral and census areal units', *Geographical Information Science* **15**, 473–485.

Ratcliffe, J. H. (2004), 'Geocoding crime and a first estimate of a minimum acceptable hit rate', *International Journal of Geographical Information Science* **18**, 61–72.

Rushton, G., Armstrong, M. P., Gittler, J., Greene, B. R., Pavlik, C. E., West, M. M. and Zimmerman, D. L. (2006), 'Geocoding in cancer research: A review', *American Journal of Preventive Medicine* **30**, 16–24.

Snae, C. (2007), A comparison and analysis of name matching algorithms, *in* 'Proceedings of World Academy of Science, Engineering and Technology', Vol. 21, World Academy of Science, Prague, Czech Republic.

Städtetag NRW (1979), 'Richtlinien für die nummerierung von gebäuden oder bebauten grundstücken'.

TagWatch (2009), 'Openstreetmap tagwatch europe'. Last accessed: April 15th, 2009.
URL: *http://tagwatch.stoecker.eu/Europe/De/index.html*

Tantner, A. (2006), 'Wer ist die nummer 1?', *Jungle World* **23**, 28–31.

Waldner, U., Machguth, H., Simonet, S. and Axhausen, K. W. (2005), Geokodierung von adressen: Methodik und erfolgsquoten bei unterschiedlichsten datenquellen, Technical report, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland.

Wiezoreck, J., Guo, Q. and Hijmans, R. T. (2004), 'The point-radius method for georeferencing locality descriptions and calculating associated uncertainty', *International Journal of Geographical Information Science* **18**(8), 745–767.

Zandbergen, P. (2007*a*), 'Influence of geocoding quality on environmental exposure assessment of children living near high traffic roads', *BMC Public Health* **7**(1), 37.

Zandbergen, P. A. (2007*b*), 'A comparison of address point, parcel and street geocoding techniques', *Computers, Environment and Urban Systems* **32**(3), 214–232.