Institut für Informatik

Bachelorarbeit

zur Erlangung des Grades "Bachelor of Science (B.Sc.)"

Offline Geokodierung für mobile Geräte mit Daten der Openstreetmap

von
Sebastian Kürten
sebastian.kuerten@fu-berlin.de

Gutachter:
Prof. Dr. Agnès Voisard
Prof. Dr. Heinz F. Schweppe

Betreuer:
Dipl. Inf. Jürgen Broß

Berlin, den 6. April 2011

Zusammenfassung

In dieser Arbeit wird erfolgreich ein hochwertiger Referenzdatensatz für die *Geokodierung von Adressen* aus der Datenbank des *Openstreetmap*-Projekts erzeugt. Als Testregion wurde sich auf die Bundesrepublik Deutschland beschränkt. Durch diese Beschränkung war es möglich, die ermittelten Daten zum Teil manuell zu überprüfen, sodass eine bestimmte Gewissheit über die Qualität hergestellt werden konnte.

Die thematischen Entitäten in der Openstreetmap wurden untersucht und durch geeignete Transformationen und Integration in ein Datenmodell überführt, dass sich für die Verwendung in einer georeferenzierten Adressdatenbank, und damit als Grundlage für die Geokodierung, eignet. Dieser Prozess wurde in einem Vorverarbeitungsvorgang implementiert, sodass regelmäßig und automatisiert die Extraktion eines Datensatzes vorgenommen werden kann.

Hierzu wurden viele technische und konzeptionelle Probleme in Umgang und Weiterverarbeitung der Rohdaten der Openstreetmap gelöst.

Der erhaltene Datensatz lässt sich in der vorliegenden Form dazu verwenden, um strukturierte Suchanfragen nach gängigen Suchmustern beantworten zu können. Dies zeigt sich in der Implementierung einer einfachen Suchkomponente für die Zielplattform, dem Mobilgerätesystem *Android*. Dort wurde eine Suchkomponente in Form einer grafischen Schnittstelle in die Anwendung *AdvancedMapViewer* des *mapsforge*-Projekts integriert.

Durch die Verwendung einer lokalen Datenbank kann bei der Ausführung auf dem Gerät auf das Bestehen einer Internetverbindung verzichtet werden. Dadurch integriert sich die Suchkomponente in das *mapsforge*-Projekt, welches auf Grundlage der Openstreetmap *offline* nutzbare Funktionalitäten für mobile Plattformen bereitstellt.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Berlin, den 6. April 2011

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Das Projekt *mapsforge* entwickelt eine Kartenapplikation zur Verwendung auf der Mobilgeräteplattform *Android*. Diese Applikation hebt sich insofern von anderen Kartenapplikationen ab, als dass sie im Betrieb nicht auf eine Internetverbindung angewiesen ist, um das Bildmaterial in Form von Rastergrafiken zu erhalten. Stattdessen wird über eine lokale Datenbank auf Vektordaten zugegriffen und diese werden ad-hoc in eine visuelle Darstellung für die Applikation umgesetzt. Vorzüge dieser Vorgehensweise finden sich beispielsweise im Bereich der Kostenersparnis oder in der Wahrung von Privatsphäre.

Eine Suchfunktion ist ein naheliegender Bestandteil einer solchen Applikation. Während der Benutzer bisher lediglich rein räumlich in der Applikation navigieren kann, ist es wünschenswert, Orte aufgrund ihrer sprachlichen Beschreibung ausfindig zu machen. Die Notwendigkeit einer solchen Komponente wird schon an einem einfachen, alltäglichen Beispiel klar: Angenommen der Benutzer kennt in einer fremden Stadt den Namen einer Straße und möchte die Lage dieser ausfindig machen. Ohne eine Suchkomponente bleibt ihm momentan lediglich die Option, die Karte der Stadt gewissermaßen sequentiell zu durchsuchen, bis er den Namen der Straße auf dem Bild entdeckt.

Solche Suchfunktionen sind im Allgemeinen nichts Neues, sondern heutzutage eher selbstverständlicher Bestandteil von kartenbasierten Applikationen. Diese basieren jedoch in aller Regel auf *proprietären* Daten, wodurch Einschränkungen in deren Nutzung bestehen. Das Openstreetmap-Projekt stellt Geodaten unter einer *freien* Lizenz zur Verfügung. Daher ist eine Verwendung der Daten der Openstreetmap erstrebenswert.

1.2 Ziel der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung eines Geokodierers auf Basis der Daten des *Openstreetmap*-Projekts. Dieser soll *offline* auf der Zielplattform, dem Mobilgerätebetriebssystem *Android*, verwendet werden können.

Dazu soll eine geeignete Komponente für die Softwarebibliothek des *mapsforge*-Projekts, sowie eine Benutzerschnittstelle für die Applikation *AdvancedMapViewer* geschaffen werden.

1.3 Problemstellung

Die Openstreetmap liefert ihre Daten in einer einfachen und flexiblen Repräsentation. In dieser Form eignen sie sich noch nicht, um damit eine Adresssuche, oder allgemein eine Geokodierung, zu implementieren. Vielmehr müssen die benötigten Daten erst aus dem Datenbestand extrahiert und für ihre Verwendung in einer Georeferenzierungskomponente angepasst werden.

Ein- und Ausgabe des Geokodierungsvorgangs sollen auf dem mobilen Gerät stattfinden, auf dem durch Hard- und Software nur begrenzte Ressourcen zur Verfügung stehen. Trotzdem sollte die entstehende Software leistungsfähig genug sein, um heutigen Ansprüchen an die Benutzerfreundlichkeit entsprechen zu können. Diese Anforderungen müssen bei Planung und Implementierung berücksichtigt werden.

1.4 Einordnung und Relevanz

Die Arbeit ist thematisch im Forschungsgebiet der *Geographischen Informationssysteme (GIS)* angesiedelt. Die Grundlagen für die meisten Teilprobleme dieser Arbeit finden sich in entsprechender Literatur:

Rigaux, Scholl und Voisard behandeln in [RSV02] die Grundlagen räumlicher Datenbanksysteme im Hinblick auf geographische Informationssysteme. Goldberg, Wilson und Knoblock erörtern in [GWK07] Grundlagen der Geokodierung und Davis, Fonseca, und Borges beschäftigen sich in [DFB03] grundlegend mit der Georeferenzierung von Adressen für die Geokodierung.

Auch zum Thema Openstreetmap ist mit [RT08] Grundlagenliteratur zu finden. Es gibt sogar mehrere Master- bzw. Diplomarbeiten, die sich mit den Themen Geokodierung auf Basis von Openstreetmap-Daten im Allgemeinen [Ame09] oder im Zusammenhang mit deren Anwendung auf mobilen Geräten [Nei08, Vet10] beschäftigen.

Die vorliegende Arbeit unterscheidet sich von diesen Arbeiten im Fokus auf die Bereitstellung eines qualitativ hochwertigen Referenzdatensatzes für die Geokodierung:

- Daten stammen grundsätzlich nur aus der Openstreetmap.
- Wo immer möglich werden hochaufgelöste Grenzregionen zur Adressbildung verwendet.
- Die Bildung des Referenzdatensatzes wird nachvollziehbar dargestellt und erläutert.

Im Gegensatz dazu greifen andere Arbeiten entweder auf proprietäres Material zurück oder verwenden grundsätzlich wesentlich niedriger aufgelöste Daten zur Adressbildung (Punkte anstelle von Polygonen als Repräsentation von Städten).

1.5 Methoden

Bei der Arbeit mit der Openstreetmap ist das *Wiki*[1] des Projekts eine unverzichtbare Informationsquelle. Da sich bei den Nutzern der Openstreetmap jedoch oft unterschiedliche Standards in der Modellierung geographischer Realität etablieren, können die Informationen des Wikis nur zur Orientierung genutzt werden. Um Aussagen über die Verwendbarkeit einer Modellierungsmethode machen zu können, ist immer ein Blick in die tatsächlich vorliegenden Daten und deren statistische Auswertung notwendig. Alle Angaben in dieser Arbeit beziehen sich auf ein Datenbankabbild vom 9. März 2011.

Ein wichtiges Mittel hierbei ist die graphische Darstellung der geographischen Objekte, sodass Eindrücke über die Relevanz einer Klasse von Objekten schnell gewonnen werden können. In der Regel kann die Relevanz jedoch auch mit Hilfe von Verhältnissen ausgedrückt werden, die sich beispielsweise über eine flächenmäßige Abdeckung des betrachteten Gebiets bilden lassen.

Zur Bewertung, insbesondere der Vollständigkeit von Daten, müssen dann externe Quellen wie beispielsweise Informationen des *statistischen Bundesamts* hinzugezogen werden. Die Analyse und Auswertung der Daten beschränkt sich in den meisten Teilen auf das Staatsgebiet der Bundesrepublik Deutschland.

1.6 Zentrale Beiträge

Für das Testgebiet Deutschland konnte eine georeferenzierte Adressdatenbank mit sehr zufriedenstellender Qualität aus den ursprünglichen Daten der Openstreetmap extrahiert werden. Diese Datenbank bildet Adressen durch Straßen, Gemeinden, Stadtteile und Postleitzahlen ab. Die kleinste adressierbare Einheit ist damit die Straße, weil Hausnummern derzeit nicht mit aufgenommen werden. Dennoch können damit viele typische Anwendungsfälle im Bereich der mobilen Geräte gut bedient werden.

Die Qualität des Referenzdatensatzes misst sich dabei vor allem am Abdeckungsgrad der gefundenen Gemeindegrenzen und Postleitzahlen, die verwendet werden, um in Kombination mit Straßen die Adressen

[1] http://wiki.openstreetmap.org/wiki/Main_Page

herzuleiten. Der Abdeckungsbereich beider Komponenten liegt dabei flächenmäßig inzwischen bei über 90%. Der verbleibende Bereich wurde bezüglich der Gemeinden mit einem weiteren Datenbestand (den Ortsmittelpunktsknoten) interpoliert, sodass eine theoretische Abdeckung von 100% erreicht wurde. Zwar können so auch fehlerhafte oder inkonsistente Daten aufgenommen werden, aber insgesamt wird die Nutzbarkeit durch diese Vollständigkeit nochmal deutlich verbessert.

Der Vorverarbeitungsprozess wurde in *Java* unter Verwendung geeigneter Softwarebibliotheken, insbesondere *JTS, JSI, Osmosis* und *Hibernate*, umgesetzt. Dieser produziert zuverlässig den Referenzdatensatz in Form einer *SQLite*-Datenbank aus einem Datenbankabbild der Openstreetmap im XML-Format. Für Deutschland läuft dieser Prozess in knapp 4 Stunden ab und produziert in dieser Zeit bereits eine Reihe von Statistiken zur Auswertung und Grafiken zur Veranschaulichung der Ergebnisse. Dadurch bleibt die Qualität der Ergebnisse leicht überprüfbar.

Außerdem wurde beispielhaft eine Benutzerschnittstelle für den *AdvancedMapViewer* erstellt. Tests mit dieser Komponente zeigen, dass die Performance mit der dateibasierten relationalen Datenbank so gut ist, dass selbst eine komfortable automatische Vervollständigung der Eingabe vorgenommen werden kann.

1.7 Struktur der Arbeit

In Abschnitt 2 werden zunächst einige Grundlagen und Begriffe zusammengestellt. Darunter fallen insbesondere geometrische und geographische Grundbegriffe, aber auch die grundlegenden Konzepte der *Geokodierung*, sowie kurze Vorstellungen der Projekte *Openstreetmap* und *mapsforge*.

Abschnitt 3 stellt das Datenmodell der Openstreetmap dar. Die Entitäten dieses Modells sind von Terminologie und Abstraktionsgrad her in einschlägiger Literatur zum Thema *geographischer Informationssysteme* eher nicht gebräuchlich. Stattdessen werden dort Begriffe wie *geographische Objekte* und *Themes* verwendet. In Abschnitt 3.2 wird daher gezeigt, wie sich die Komponenten im Datenmodell der Openstreetmap zu solchen Objekten abstrahieren lassen.

Abschnitt 4 wirft einen Blick auf eine Auswahl relevanter *thematischer Entitäten* aus der Openstreetmap und stellt deren Eigenschaften vor. In der Regel wird dazu auf den Grad der Abdeckung des Testgebiets Deutschland eingegangen, sowie auf die Relevanz einzelner Attribute.

Die rudimentäre Umsetzung einer kompletten Geokodierungskomponente wird in Abschnitt 5 konzeptionell entwickelt. Basierend auf Informationen [DFB03] über weltweite Gemeinsamkeiten in den Komponenten von Adressen und den tatsächlich vorhandenen, verwertbaren Daten in der Testregion wird hier ein einfaches Modell zur Repräsentation von Adressen entwickelt. Dieses stellt zwar nicht den Anspruch tatsächlich weltweit einsetzbar zu sein, sondern es sollte sich vielmehr zumindest auf einige andere Staaten übertragen lassen und nicht ausschließlich in Deutschland anwendbar sein. Ebenfalls in diesem Abschnitt wird die Analyse der Anforderungen an die Schnittstelle der Geokodierungskomponente auf dem mobilen Gerät durchgeführt.

Das Modell aus Abschnitt 5 bildet Adressen zusammen mit einer Georeferenz ab und stellt als Referenzdatenmodell die Grundlage für den gesamten Geokodierungsprozess dar. Abschnitt 6 beschäftigt sich dann damit, die Entitäten dieses Modells und insbesondere deren gegenseitige Beziehungen aus den Entitäten der Openstreetmap herzuleiten. Dazu wird hier ein Extraktions- und Integrationsprozess vorgestellt, der einerseits die gesuchten Objekte herleitet, zum Teil aber auch mehrere thematische Entitäten integriert.

Die Implementierung dieses Prozesses wird dann in Abschnitt 7 präsentiert. Auch die Implementierung einer einfachen Komponente zur Geokodierung in Form einer einfachen API für die Softwarebibliothek von *mapsforge*, sowie in Form einer Benutzerschnittstelle für den *AdvancedMapViewer* wird hier vorgestellt. Außerdem wird die Kompatibilität zur Geokodierungsschnittstelle von *Android* angesprochen.

Abschließend werden in Abschnitt 8 die Ergebnisse zusammengefasst und ein Ausblick gegeben.

2 Grundlagen

2.1 Geometrische Begriffe

Die folgenden geometrischen Begriffe werden in dieser Arbeit verwendet und sollen daher kurz erläutert werden. Deren Definitionen orientieren sich an den Ausführungen aus [RSV02]. Es wird zwischen *Punkten, linienartigen* und *polygonalen* Objekten unterschieden, bei denen es sich um in die Ebene eingebettete geometrische Objekte handelt.

Punkte

Punkte sind nulldimensionale Objekte, d.h. sie haben keinerlei räumliche Ausprägung. Sie stellen eine Position auf der Ebene dar und werden nicht weiter unterschieden.

Linienartige Objekte / Linien

Linienartige Objekte sind Objekte mit eindimensionaler Ausprägung. Auch wenn sie zusammenfassend oft kurz als *Linien* bezeichnet werden, lassen sich folgende Begriffe unterscheiden:

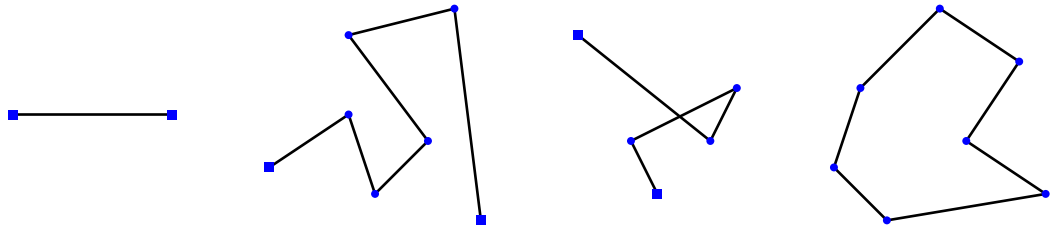
- *Geradenstück.* Ein Geradenstück ist ein durch zwei Punkte begrenzter Teil einer Geraden auf der Ebene (Abbildung 1(a)). Die beiden Punkte, die ein Geradenstück definieren, werden als Endpunkte bezeichnet.
- *Streckenzug.* Ein Streckenzug ist als endliche Menge von Geradenstücken definiert. Die Endpunkte dieser Geradenstücke heißen dann Punkte des Streckenzugs. Dabei gilt für jeden dieser Punkte, dass er Endpunkt von genau zwei der Geradenstücke ist. Je zwei Geradenstücke, die einen gemeinsamen Endpunkt haben, werden als *aufeinanderfolgende* Geradenstücke bezeichnet. Einzige Ausnahme bilden die beiden Endpunkte des Streckenzugs, auch Extrempunkte genannt, welche nur von jeweils einem Geradenstück als Endpunkt verwendet werden. Abbildung 1 zeigt solche Streckenzüge. Dabei sind gewöhnliche Punkte des Streckenzugs als Kreise und die Extrempunkte als Quadrate dargestellt.

  Streckenzüge können folgendermaßen weiter unterschieden werden:

  - Ein Streckenzug heißt *einfach*, wenn für alle nicht aufeinander folgenden Geradenstücke gilt, dass sie sich in keinem Punkt schneiden. Aufeinanderfolgende Geradenstücke schneiden sich nur in ihrem gemeinsamen Endpunkt. Gibt es weitere Schnittpunkte, wird der Streckenzug als *nicht einfach* bezeichnet. Im folgenden sind, wenn nicht genauer bezeichnet, immer *einfache* Streckenzüge gemeint. Abbildung 1(b) zeigt ein Beispiel für einen einfachen Streckenzug und Abbildung 1(c) ein Beispiel für einen nicht einfachen Streckenzug.
  - Wenn die beiden Extrempunkte identisch sind, spricht man von einem *geschlossenen* Streckenzug. Möchte man betonen, dass ein Streckenzug nicht geschlossen ist, kann er als *offen* bezeichnet werden. Wird dieses Kriterium nicht explizit erwähnt, sind offene Streckenzüge gemeint. Abbildung 1(d) zeigt einen geschlossenen Streckenzug.

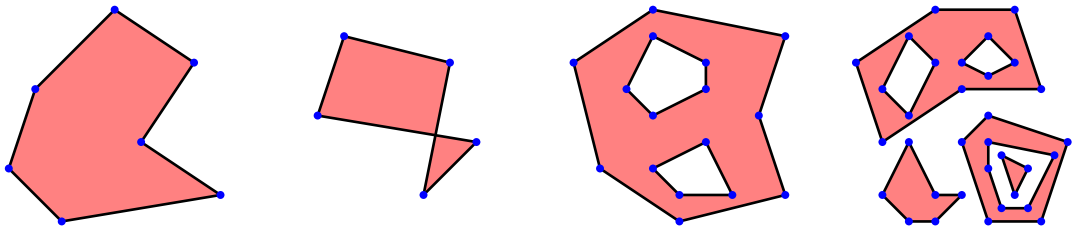
Polygonale Objekte / Flächen

- *Polygon.* Ein geschlossener Streckenzug umschließt eine Fläche, die als Polygon bezeichnet wird. Die Eigenschaft der *Einfachheit* von Streckenzügen überträgt sich auf Polygone. Ein Polygon heißt einfach, genau dann wenn der das Polygon begrenzende Streckenzug einfach ist. Dementsprechend ist das Polygon in Abbildung 2(a) einfach und das Polygon in Abbildung 2(b) nicht einfach.
- *Verallgemeinertes Polygon.* Schneidet man aus einem einfachen Polygon ein zweites, enthaltenes Polygon aus, entsteht im ersten Polygon ein Loch. Ein allgemeines Polygon kann endlich viele



(a) (b) (c) (d)

Abbildung 1: Eindimensionale Objekte: Geradenstück (a), einfacher Streckenzug (b), nicht-einfacher Streckenzug (c) und geschlossener Streckenzug (d)



(a) (b) (c) (d)

Abbildung 2: Zweidimensionale Objekte: einfaches Polygon (a), nicht-einfaches Polygon (b), Polygon mit Löchern (c) und Region (d)

solcher Löcher haben. Abbildung 2(c) zeigt ein Polygon mit zwei Löchern. Im Weiteren sind immer allgemeine Polygone gemeint, wenn von Polygonen gesprochen wird.

- *Region.* Eine Menge von allgemeinen Polygonen wird als *Region* bezeichnet. In der Regel wird davon ausgegangen, dass sich die einzelnen Polygone nicht überschneiden. Sehr wohl können sich Polygone aber innerhalb der Löcher anderer Polygone befinden. Ein Beispiel für eine solche Region ist Abbildung 2(d).

2.2 Geometrische Operationen und Prädikate

Für geometrische Objekte lassen sich geometrische Operationen und Prädikate definieren. Im folgenden wird eine relevante Auswahl dieser dargestellt. Die Signaturen der Operationen wurden dabei zur Übersicht vereinfacht und spiegeln nicht mathematische Realität wider. Ein Beispiel hierfür ist der Schnitt zweier Regionen: Ergebnis dieser Operation könnte genaugenommen einerseits eine Linie sein, andererseits aber auch eine Menge von Objekten unterschiedlichen Typs. Die Operationen werden mit den Signaturen vorgestellt, mit denen sie in dieser Arbeit relevant sind.

Mengentheoretische Operationen

Viele mengentheoretische Operationen lassen sich auf geometrische Objekte übertragen:

- Vereinigung: Region × Region → Region
- Schnitt: Region × Region → Region
- Differenz: Region × Region → Region

- Vereinigung: {Linie} × {Linie} → {Linie}
- Schnitt: Linie × Linie → {Punkt}

Prädikate

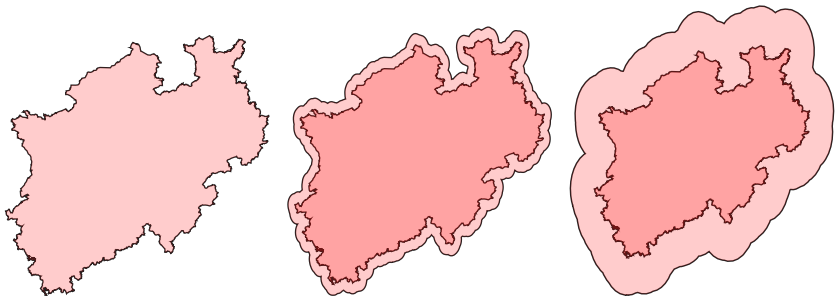
Aus den mengentheoretischen Operationen lassen sich auch boolesche Prädikate ableiten, wie Schnitt und Überdeckung:

- Schnitt: (geometrisches Objekt) × (geometrisches Objekt) → {true, false}
- Überdeckung: Region × (geometrisches Objekt) → {true, false}

Weitere Operationen

Daneben sind die folgenden Operationen von Bedeutung:

- Flächeninhalt: Region → $\mathbb{R}$. Berechnet den Flächeninhalt einer Region.
- Distanz: (geometrisches Objekt) × (geometrisches Objekt) → $\mathbb{R}$. Berechnet den minimalen Abstand zwischen zwei Objekten beliebigen Typs.
- Puffer: Region × $\mathbb{R}$ → Region. Bildet aus einer Region eine neue Region, dessen äußere Begrenzung um eine bestimmten Distanz erweitert wurde. Abbildung 3 zeigt einige Beispiele für einen solchen Puffer anhand der Grenze Nordrhein-Westfalens.
- Vereinfachung: Region × $\mathbb{R}$ → Region. Bildet aus einer Region eine neue, zu einem bestimmten Grad vereinfachte, Region. Abbildung 4 zeigt Beispiele der sogenannten Douglas-Peucker-Vereinfachung, bei der alle Punkte der resultierenden Region in einer maximalen Distanz zur gegebenen Region liegen [Peu76].
- Voronoi: {Punkt} → {Region}. Bestimmt für eine Menge von Punkten eine Zerlegung der Ebene in Polygone [Aur91]. Abbildung 5 zeigt ein Voronoi-Diagramm für 20 zufällig gewählte Punkte.

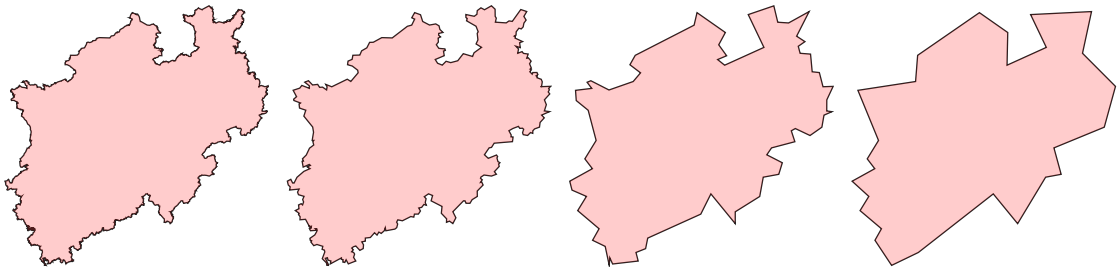


(a) (b) (c)

Abbildung 3: Regionspuffer: Original (a), Original und 0,1 Grad Puffer (b), Original und 0,4 Grad Puffer (c)

2.3 Geographische Begriffe und Operationen

Folgende Begriffe sind im Zusammenhang mit *geographischen Informationssystemen* gebräuchlich [RSV02]:



(a) (b) (c) (d)

Abbildung 4: Douglas Peucker Vereinfachung: Original (a), maximale Distanz: 0,01 Grad (b), 0,05 Grad (c) und 0,1 Grad (d)

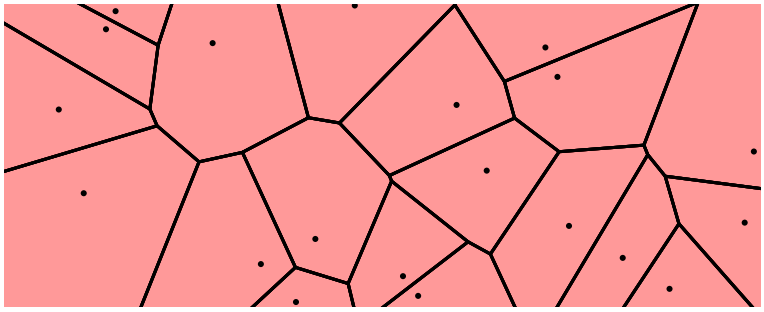


Abbildung 5: Ein Voronoi Diagramm für 20 Punkte

Geographisches Objekt

Geographische Objekte sind Objekte, die eine geographische Gegebenheit modellieren. Sie bestehen grundsätzlich aus zwei Arten von Komponenten:

- *Deskriptive Attribute* oder *Sachattribute*. Diese Attribute beschreiben alle nicht-räumlichen Eigenschaften eines Objekts.
- *Geometrische Komponente*. Hiermit wird die räumliche Komponente eines Objekts abgebildet.

Theme

Ein *Theme* ist eine Sammlung von geographischen Objekten, die thematisch zusammengehören. Sie definieren Klassen von Objekten, welche im Weiteren als thematische Entitätsklassen bezeichnet werden. Ein Beispiel für ein Theme ist die Menge der Städte in Deutschland.

Geographische Operationen

- Thematische Selektion (Theme Selection) Die thematische Selektion ist die Auswahl geographischer Objekte aufgrund ihrer Sachattribute. Ein Beispiel hierfür wäre die Auswahl aller Städte deren Name mit dem Buchstaben 'B' beginnt.
- Räumliche Selektion (Spatial Selection) Die räumliche Selektion ist eine Auswahl geographischer Objekte aufgrund ihrer räumlichen Komponente gemäß eines geometrische Prädikats, das mit einem anderen räumlichen Objekt evaluiert wird. Ein Beispiel hierfür ist die Auswahl aller Bundesländer in Deutschland aufgrund des geometrischen Prädikats der Überlagerung.

2.4 Openstreetmap

Das *Openstreetmap*-Projekt sammelt seit der Gründung im Jahr 2004 geographische Daten und stellt diese der Öffentlichkeit zur Verfügung. Die Vorgehensweise lässt sich mit der des Wikipedia-Projekts vergleichen [RT08]: Die Daten des Projekts werden von Individuen gesammelt und zentral zusammengeführt. Dabei kann, das technische Know-How vorausgesetzt, jeder teilnehmen, wobei der Einzelne an wenige inhaltliche Vorgaben gebunden ist.

Daraus ergibt sich auch eines der größten Probleme im Umgang mit den gesammelten Daten in Bezug auf diese Arbeit: Es gibt in der Regel verschiedenste Möglichkeiten, geographische Fakten abzubilden und es kann nicht davon ausgegangen werden, dass weltweit die selben Standards Anwendung finden.

2.5 Mapsforge / AdvancedMapViewer

Das *mapsforge*-Projekt[2] ist ein im Jahr 2008 an der Freien Universität entstandenes Projekt, das sich mit der Bereitstellung einer Softwarebibliothek zur Verwendung der Daten der Openstreetmap auf mobilen Geräten befasst. Kern des Projekts ist eine Bibliothek für das Mobilgerätebetriebssystem *Android*. Diese Bibliothek kann zur Darstellung einer Karte in Applikationen verwendet werden. Ein Fokus des Projekts liegt darauf, alle Funktionen offline, also unabhängig von einer bestehenden Internetverbindung, zur Verfügung zu stellen. Dabei ist eine zentrale Komponente der Renderer, der auf Basis der Roh-Vektordaten das anzuzeigende Bild live erzeugt. Darüber hinaus gibt es weitere Funktionalitäten wie beispielsweise eine API zur Darstellung von Zusatzinformationen (Overlays) und eine Routingkomponente.

Der *AdvancedMapViewer* ist eine im Rahmen des Projekts mitentwickelte Applikation für Android, welche die Bibliothek in geeigneter Weise für den Endanwender nutzbar macht. Kernfunktionalität dieser Applikation ist momentan die Darstellung einer interaktiven Karte, auf der der Nutzer navigieren kann.

2.6 Geokodierung

Geokodierung (Geocoding) beschreibt ganz allgemein den Prozess, einer beliebigen ortsbezogenen Beschreibung ein geographisches Objekt zuzuordnen [GWK07]: Im einfachsten Fall handelt es sich bei einem solchen geographischen Objekt um einen Punkt auf der Erde, aber im Grunde kann es sich dabei um beliebige geographische Objekte handeln. Ein typischer Fall ist beispielsweise, einer Adresse in Textform ihre Position zuzuordnen.

Der Prozess der Geokodierung lässt sich in die folgenden fundamentalen Bestandteile zerlegen, die von [GWK07] so beschrieben werden:

- Eingabe: ist eine ortsbezogene Information, die der Nutzer in eine geographische Information umwandeln möchte.
- Ausgabe: ist die geographische Information, die vom Verarbeitungsalgorithmus für die Eingabe bestimmt wird.
- Verarbeitungsalgorithmus: Dieser bestimmt eine adäquate Ausgabe für die Eingabe. Typischerweise kommen hier Methoden wie Standardisierung und Normalisierung der Eingabedaten zur Herstellung der Kompatibilität mit dem Referenzdatensatz und Matching zur Auswahl der Georeferenz zum Einsatz.
- Referenzdatensatz: ist die Komponente, aus der vom Verarbeitungsalgorithmus Ausgaben für Eingaben abgeleitet werden.

[2] http://mapsforge.com

3 Geographische Daten in der Openstreetmap

Geographischen Daten sind in vielerlei Hinsicht sehr komplex, weil die Realität, die sie modellieren komplex ist. Die *Openstreetmap* nutzt zur Ablage ihrer geographischen Daten jedoch ein sehr simples Datenmodell. Dieses Modell, welches im Weiteren als *strukturelles Modell* bezeichnet wird, stelle ich in Abschnitt 3.1 vor. Da es sich beim strukturellen Modell um ein sehr generisches Datenmodell handelt, gehe ich dann in Abschnitt 3.2 darauf ein, wie verschiedene Komponenten *geographischer Objekte* in diesem Modell systematisch abgelegt werden. Dies führt zu einem abstrakteren Datenmodell, welches im Folgenden als *geographisches Modell* bezeichnet wird.

3.1 Strukturelles Modell

Das strukturelle Modell (Abbildung 6) definiert in erster Linie die drei Entitäten *Knoten, Weg* und *Verbund*. Die Entität *Objekt* ist abstrakt und wurde lediglich zur Modellierung gemeinsamer Sacheigenschaften dieser drei Entitäten ins Modell mit aufgenommen.

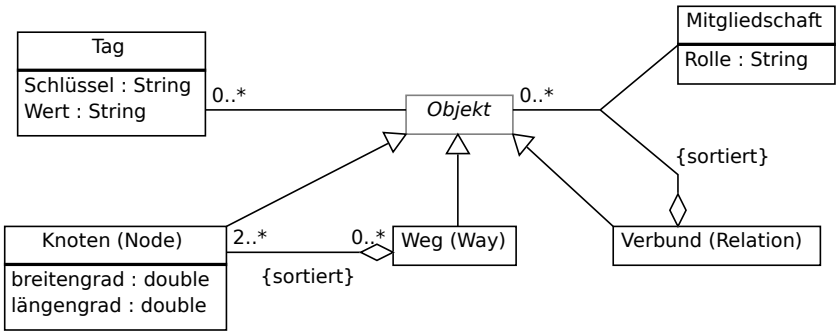


Abbildung 6: Strukturelles Schema der Openstreetmap

Die drei Basistypen lassen sich so zusammenfassen:

- Ein **Knoten** beschreibt einen Punkt auf der Erdoberfläche. Er enthält als Attribute eine Positionsangabe gegeben durch Längen- und Breitengrad.
- Ein **Weg** ist eine Liste von Knoten mit Reihenfolge. Diese Knoten definieren mit ihren Koordinaten einen Streckenzug. Damit dies möglich ist, muss ein Weg aus mindestens zwei Knoten bestehen.
- Ein **Verbund** ist eine Liste von Objekten mit Reihenfolge. Die Bestandteile eines Verbunds werden *Mitglieder* genannt. Mitglieder einer Relation können Objekte jeden Typs sein, also Knoten, Wege und auch Verbünde selbst. Für jedes Mitglied kann der Typ der Mitgliedschaft über eine Zeichenkette, die sogenannte *Rolle*, näher klassifiziert werden.

Bei Knoten und Wegen handelt es sich um sehr einfache Objekte mit einer intuitiv verständlichen Bedeutung der Attribute bzw. Bestandteile: Ein Knoten modelliert einen Punkt und ein Weg einen Streckenzug auf der Erdoberfläche. Der Verbund hingegen fasst eine Menge von Objekten zu einer logischen Einheit zusammen und ist damit ein vergleichsweise komplexes Objekt, das sich zur Darstellung verschiedener Zusammenhänge eignet:

- Gruppierung mehrerer geographischer Objekte zu einer logischen Einheit: Im Grunde können so beliebige semantische Beziehungen zwischen Objekten hergestellt werden. Ein Beispiel hierfür findet sich in Abschnitt 3.2.4.

- Ein wichtiger Spezialfall einer solchen Gruppierung ist die Gruppierung einer Menge von Wegen, sodass diese eine Region definieren. Ein solcher Verbund eignet sich dann zur Darstellung beliebiger Flächen. Näheres hierzu findet sich in Abschnitt 3.2.2.

Jedes Objekt kann über **Tags** beliebige textuelle Attribute haben. Ein Tag ist ein Schlüssel mit einem zugehörigen Wert. Die Menge der Tags eines Objekts wird wörterbuchartig interpretiert, d.h. jeder Schlüssel ist pro Exemplar eindeutig. Die durch Tags definierten Attribute werden vor allem dazu verwendet, die Eigenschaften geographischer Objekte zu beschreiben.

Ein Beispiel im XML-Format

Eine weit verbreitete Repräsentation von Daten in diesem Schema ist die in Form eines XML-Dokuments wie dem folgenden. Es gibt ein Wurzelobjekt des Typs *osm*. Alle Exemplare der strukturellen Entitäten sind im Dokumentenbaum Kindknoten dieses Objekts. Tags der Objekte und Knotenverweise der Wege sind Kindknoten des jeweiligen Basisobjekts.

Das folgende Beispiel zeigt, wie eine einfache Straße in diesem Modell dargestellt wird:

```
1  <?xml version='1.0' encoding='UTF-8'?>
   <osm version="0.6">
3    <node id="26554142" lat="52.4789866" lon="13.3624623" />
     <node id="26554134" lat="52.4791453" lon="13.3624614">
5      <tag k="highway" v="traffic_signals"/>
     </node>
7    <node id="26554136" lat="52.4809377" lon="13.3624918"/>
     <node id="26554137" lat="52.4813744" lon="13.362502"/>
9    <node id="26554138" lat="52.483262" lon="13.3625222"/>
     <way id="4710940">
11     <nd ref="26554142"/>
       <nd ref="26554134"/>
13     <nd ref="26554136"/>
       <nd ref="26554137"/>
15     <nd ref="26554138"/>
       <tag k="created_by" v="JOSM"/>
17     <tag k="highway" v="residential"/>
       <tag k="maxspeed" v="30"/>
19     <tag k="name" v="Gustav-Müller-Straße"/>
       <tag k="postal_code" v="10829"/>
21   </way>
   </osm>
```

Die Datei definiert zunächst fünf Knoten. Jeder dieser Knoten hat als Attribute eine *id* als Identifier, sowie seinen Längen- und Breitengrad. Einer dieser Knoten ist mit dem Tag *highway=traffic_signals* ausgestattet, was bedeutet, dass hier eine Ampel steht. Ab Zeile 10 wird ein Weg definiert. Durch den Tag mit dem Schlüssel *highway* ist der Weg als Straße gekennzeichnet. Ein Tag mit dem Schlüssel *name* benennt sie als Gustav-Müller-Straße und weitere Tags definieren die Straße als im Wohngebiet gelegen, mit einem Tempolimit von 30 Stundenkilometern. Außerdem ist ein Verweis auf die Postleitzahl *10892* gegeben. Die Straße referenziert die weiter oben definierten fünf Knoten, wodurch der Verlauf der Straße auf der Erdoberfläche gegeben ist.

3.2 Geographisches Modell

Mit Hilfe des strukturellen Modells werden alle Facetten *geographischer Objekte* abgebildet. Darunter fallen *gewöhnliche Sachattribute*, die mit primitiven Datentypen beschrieben werden können, *räumliche Komponenten*, die intrinsischer Bestandteil geographischer Objekte sind, die *Taxonomie*, welche die unterschiedlichsten Klassen von Objekten definiert, sowie die *Beziehungen*, die zwischen den Objekten bestehen. Im Folgenden gehe ich auf jeden dieser Aspekte ein und erläutere, wie sie im strukturellen Modell abgebildet werden.

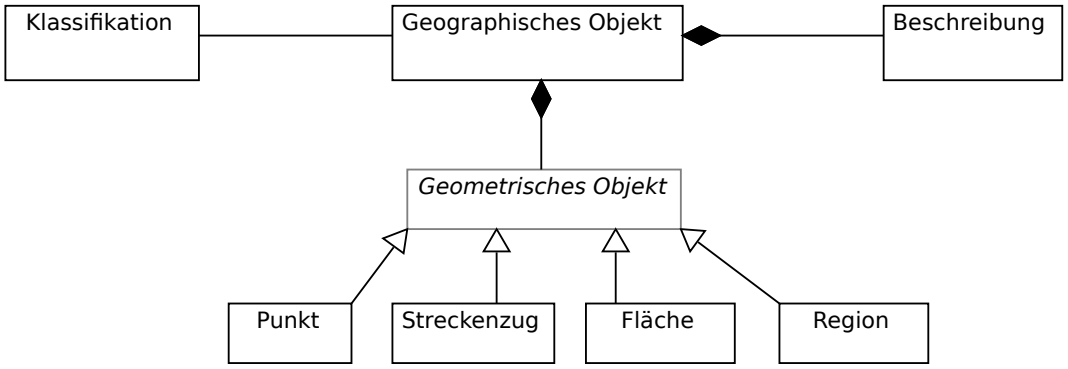


Abbildung 7: Geographisches Schema der Openstreetmap

3.2.1 Einfache Sachattribute

Gewöhnliche Attribute, die mit primitiven Datentypen, wie etwa Ganzzahlen, Gleitkommazahlen oder Zeichenketten ausgedrückt werden können, werden mit Tags dargestellt. Allerdings existiert der Einfachheit halber lediglich ein einziger Typ als Wertemenge dieser Attribute: die Zeichenkette. Andere primitive Datentypen müssen daher in Form einer Zeichenkette abgelegt werden.

Beispielsweise werden einer Stadt in der Regel eine ganze Reihe solcher Attribute zugeordnet. Zum Beispiel hat die Stadt Berlin ein Attribut *name*, welches dem Objekt den Namen "Berlin" zuschreibt. Darüber hinaus gibt es 157 Attribute mit Schlüsseln der Form *name:sprachkürzel*, welche den Namen der Stadt in der jeweiligen Sprache angeben. Andere Attribute beschreiben Eigenschaften wie die Anzahl der Einwohner (*population*), die URL der offiziellen Repräsentanz im WWW (*website*) oder einen Verweis auf Wikipedia-Artikel in verschiedenen Sprachen (*wikipedia:sprachkürzel*).

Welche Attribute an einem Objekt vorhanden sind, ist insbesondere davon abhängig, wie ein Objekt taxonomisch einzuordnen ist (Siehe hierzu Abschnitt 3.2.3). Zum Beispiel kann erwartet werden, dass das Objekt, das eine Stadt modelliert zumindest einen Namen hat.

3.2.2 Räumliche Komponenten

Geographische Objekte bestehen insbesondere auch aus einer räumlichen Komponente. Vereinfacht man die Erde durch geeignete Projektion auf eine zweidimensionale Ebene, so kann man sich in der Darstellung der räumlichen Komponente gut auf zweidimensionale Objekte beschränken. Um Objekte auf der Ebene zu beschreiben, werden in der Regel eine Auswahl von geometrischen Basistypen verwendet. Im Wesentlichen benötigt man jeweils einen Typen um 0-, 1- und 2-dimensionale Objekte darstellen zu können. Mithilfe dieser Basistypen werden dann auch komplexere geometrische Gebilde dargestellt. In der Openstreetmap finden sich folgende Basistypen:

räumliche Komponente	Dimension	strukturelles Objekt	Merkmale
Punkt	0	Knoten	keine
Streckenzug	1	Weg	keine
einfache Fläche	2	Weg	geschlossen; area=yes, o.ä.
Region	2	Verbund	type=multipolygon

Abbildung 8: Abbildung von räumlichen Objekten im strukturellen Modell

- *Punkte*: werden direkt über den strukturellen Typ *Knoten* abgebildet.

- *Streckenzüge*: werden durch den Basistyp *Weg* repräsentiert. Die Liste von Knoten in einem *Weg* des strukturellen Modells definiert die Punkte des Streckenzugs. Das *n*-te Geradenstück eines Streckenzugs ist gegeben durch den *n*-ten und den *n+1*-ten Knoten des Wegs. Dadurch ist per Definition gegeben, dass jeder Punkt (außer den Endpunkten) von mindestens zwei Geradenstücken verwendet wird.
- *Einfache Flächen*: Sind der letzte und erste Knoten eines Wegs identisch, so handelt es sich um einen geschlossenen Weg. Dementsprechend repräsentiert ein solcher Weg einen geschlossenen Streckenzug, der als Polygon interpretiert werden kann. Das Attribut *area=yes* zeichnet einen solchen geschlossenen Weg als Polygon aus. Darüber hinaus gibt es noch Attribute, welche die Interpretation als Fläche implizieren. Ein Beispiel hierfür ist der Tag *landuse=forrest*, welcher nur für Flächen Sinn macht.
- *Regionen*: werden mittels der Basisentität *Verbund* modelliert. Ein solcher Verbund ist in der Regel mit dem Tag *type=multipolygon* markiert. Der Verbund hat als Mitglieder eine Menge von Wegen, welche innere und äußere Begrenzungslinien der Polygone der Region bilden. Die Bildung einer Region aus den Mitgliedern des Verbunds geschieht durch Zusammensetzen dieser Wege zu geschlossenen Streckenzügen. Im Abschnitt 7.1.4 wird auf diesen Vorgang genauer eingegangen und ein konkreter Algorithmus vorgestellt.

3.2.3 Klassifikation / Taxonomie thematischer Entitäten

In der Openstreetmap sind viele Arten von geographischen Objekten und somit auch verschiedene thematische Entitäten vorhanden. Diese spiegeln sich jedoch nicht direkt in Form von Entitäten im strukturellen Modell wider, sondern grundsätzlich sind alle Objekte entweder vom Typ Knoten, Weg oder Verbund. Der strukturelle Typ in Kombination mit den Tags eines Objekts bestimmt, um welche Art von Objekt es sich thematisch handelt.

Eine thematische Entität kann also als Unterklasse einer strukturellen Entität verstanden werden, die sich aus seiner Oberklasse durch eine bestimmte Attributkonfiguration der Exemplare ergibt. Die hierfür zum Einsatz kommenden Tags werden hier als *klassifizierende* Tags bezeichnet.

Ein Beispiel für eine solche thematische Entität sind Restaurants. Diese werden in der Regel als Punkt dargestellt, die als klassifizierendes Attribut den Tag *amenity=restaurant* haben. Ein Objekt dieser Klasse hat dann in der Regel weitere Attribute, die das Restaurant näher beschreiben. Üblich ist beispielsweise, die Verwendung des Tags *cuisine* um zu beschreiben welche Art von Speisen im Restaurant serviert werden. Ebenfalls ist in der Regel zu erwarten, dass einem Restaurant ein Name zugewiesen wird.

In Abschnitt 4 werden die thematischen Entitäten vorgestellt, die im Weiteren für diese Arbeit relevant sind.

3.2.4 Beziehungen zu anderen Objekten

Zwischen den Exemplaren thematischer Entitäten können in der Regel Beziehungen bestehen. Das Datenmodell der Openstreetmap bietet drei Möglichkeiten, solche Beziehungen zu modellieren. Erstens können Sachattribute Referenzen auf andere Objekte enthalten. Zweitens können implizite, topologische Beziehungen bestehen, die durch die geometrischen Komponenten der Objekte bedingt sind. Drittens kann die strukturelle Komponente *Verbund* benutzt werden. Im Folgenden werden diese drei Arten genauer erläutert. Als Beispiel dient jeweils die Lagebeziehung, die zwischen Bundesländern und Städten ausgemacht werden kann:

- Sachattribute haben Werte primitiven Typs und werden eigentlich verwendet, um die Objekte an sich zu beschreiben. Sie können aber darüberhinaus auch benutzt werden, um so etwas wie Fremdschlüssel zu anderen Objekten zu kodieren. Im Beispiel könnte im thematischen Objekt Stadt

ein Attribut *Bundesland* vorgesehen sein, welches als Wert den Namen des die Stadt beinhaltenden Bundeslands enthalten soll.

- Zwischen geometrischen Objekten können diverse topologische Beziehungen bestehen. Diese lassen sich mit Hilfe geometrischer Prädikate formulieren (Siehe Abschnitt 2.2). Die Semantik dieser Beziehung ist dann vom Typ der Objekte und des Prädikats abhängig.

  In unserem Beispiel könnte also die Lagebeziehung zwischen den Objekten Stadt und Bundesland auch über ihre geometrische Komponente entschieden werden. Voraussetzung dafür ist jedoch, dass:

  – die geometrischen Bestandteile der Objekte akkurat vorliegen (in diesem Fall die Grenzen von Stadt und Bundesland)
  – die Beziehung über ein solches Prädikat herstellbar ist (in diesem Fall das Prädikat der Überdeckung).

  Dann lässt sich definieren, dass eine Stadt in dem Bundesland gelegen ist, dessen Grenzregion die Region der Stadt überdeckt.

- Das strukturelle Element *Verbund* kann auch zur Herstellung von Beziehungen dienen. Im Grunde können über die Verbünde beliebige *m:n*-Beziehungen modelliert werden. Dazu kann beispielsweise ein eigener Verbundstyp in der Taxonomie definiert werden, bei dem die eine Seite der Beziehung über Attribute und die andere Seite über die Mitglieder des Verbunds modelliert wird. Das strukturelle Modell bietet hier viele Möglichkeiten.

  Für das Beispiel könnte ausgenutzt werden, dass das Bundesland ohnehin als Verbund abgelegt ist, da es sich um eine Region handelt. Dann könnten alle im Bundesland gelegenen Städte als Mitglied in den Verbund aufgenommen werden und die Mitgliedschaft mit einer geeigneten Rolle versehen werden, wie der Rolle *Enthaltene-Stadt*.

4 Relevante thematische Entitäten in der Openstreetmap

Im geographischen Schema der Openstreetmap sind sehr viele thematische Entitäten vorhanden.[3] Die folgenden Entitäten werden hier detailliert dargestellt:

- *Straßensegmente*: hiermit wird das Straßennetzwerk modelliert.
- *Postleitzahlregionen*: definieren Regionen, in denen Postleitzahlen gültig sind.
- *Administrative Einheiten*: modellieren politische und verwaltungsmäßige Grenzen wie zum Beispiel Grenzen von Bundesländern und Städten.
- *Ortsmittelpunkte*: modellieren Gemeinden und Ortschaften.

Insbesondere wird auch darauf eingegangen, welche Relevanz sie für die Testregion Deutschland haben. Abbildung 9 fasst die Eigenschaften dieser Entitäten zusammen.

Entität	geometrische Komponente	klassifizierende Attribute	weitere Attribute
Straßensegment	Streckenzug oder einfache Fläche	highway=*Straßentyp*	name=*Name*
Postleitzahlregion	Region	postal_code=* oder postcode=*	
Administrative Einheit	Region	boundary=administrative	name=*Name* admin_level=*[1..11]*
Ortsmittelpunkt	Punkt	place={city | town | village}	name=*Name*
Adresse	Punkt oder Streckenzug	addr:housenumber=* addr:interpolation=*	addr:*=*

Abbildung 9: Thematische Entitäten mit ihren Eigenschaften

Diese Entitäten beeinflussen die Konzeption der Geokodierungskomponente in Abschnitt 5 und bilden die Grundlage für den Extraktionsprozess in Abschnitt 6.

4.1 Straßensegmente

Abbildung 10: Straßensegmente in Berlin-Mitte

[3] http://wiki.openstreetmap.org/wiki/Map_Features

Straßensegmente werden zu Modellierung des Straßennetzwerks eingesetzt. Geometrisch handelt es sich dabei um Streckenzüge, die den Straßenverlauf modellieren. Dabei wird die Straße als eindimensionales Objekt modelliert, wobei der Streckenzug die Mitte der Straße angibt. Optional kann die Breite der Straße über das Attribut *width* angegeben werden.

Für Fußgängerzonen und Plätze kann anstelle des Streckenzugs auch eine einfache Fläche als geometrische Komponente verwendet werden. Dadurch sollen diese mit ihrer oftmals eigentümlichen Form besser abgebildet werden können. In Deutschland finden sich gut 16000 solcher Segmente, was 0,3% gemessen an der Gesamtzahl der Straßensegmente entspricht.

Auftreten		Schlüssel
relativ	absolut	
100,0%	1 828 386	name
100,0%	1 828 386	highway
22,6%	413 281	maxspeed
13,1%	239 900	created_by
16,6%	303 783	surface
10,0%	182 721	ref
8,6%	**167 013**	**postal_code**
7,4%	136 153	oneway
4,8%	88 345	source
4,2%	76 022	tracktype
4,1%	75 715	layer
3,8%	68 913	width
3,4%	62 413	bicycle
3,0%	54 250	foot
2,5%	44 922	lit
2,4%	43 285	bridge
2,2%	40 178	access
2,1%	38 409	cycleway
1,7%	31 463	lanes
1,2%	22 503	motorcar
1,1%	19 727	zone:traffic
1,0%	18 091	service
0,9%	15 793	note
0,8%	**15 793**	**is_in**
0,4%	**7 226**	**addr:postcode**
...	...	...
0,0%	**138**	**is_in:municipality**
0,0%	**124**	**is_in:town**
0,0%	**16**	**is_in:city**
0,0%	**16**	**is_in:village**

Abbildung 11: Tags der Straßensegmente in Deutschland nach Häufigkeit der Verwendung

Klassifikation

Straßensegmente werden über ein Attribut mit dem Namen *highway* als solche klassifiziert. Dieses Attribut benennt über seinen Wert den Typ der Straße. Typische Werte sind *motorway*, *primary*, *secondary*, *tertiary*, *residential*, *unclassified* und *track*. Diese Werte ordnen die Segmente in eine Hierarchie bezüglich ihrer Bedeutung für den Straßenverkehr ein. In Deutschland wird beispielsweise *motorway* verwendet um Autobahnen zu markieren, *primary* für Bundesstraßen und *residential* oder *unclassified* für Wohngebietsstraßen.

Attribute

Straßensegmente haben als Attribut oft einen Namen. In Deutschland haben insgesamt 43% der Straßensegmente einen Namen. Ob ein Name vorhanden ist, ist stark vom Typ der Straße abhängig. So haben zum Beispiel 89% der Segmente von Wohngebietsstraßen einen Namen, 57% der Autobahnen aber nur 8% der Feld- und Waldwege. Die benannten Straßensegmente bilden eine Untermenge der Straßensegmente. Da ein Name als Bezeichner für die Straßensegmente für die weitere Verarbeitung für die Adressdatenbank notwendig sein wird, beziehen sich die weiteren Ausführungen auf eben diese Untermenge.

Darüber hinaus haben Straßensegmente in der Regel weitere Attribute. Abbildung 11 listet für benannte Straßensegmente auf, welche Attribute in welcher Häufigkeit verwendet werden. Dargestellt werden alle Schlüssel, die in mindestens 1% der Straßensegmente Verwendung finden, sowie einige Schlüssel mit geringerer Verwendung, die jedoch von der Semantik her interessant sind, da sie Beziehungen modellieren.

Die meisten dieser Tags beschreiben jedoch einfache Sachattribute der Straße: Zum Beispiel wird *maxspeed* verwendet, um die zulässige Höchstgeschwindigkeit, *surface* um die Art oder Qualität des Straßenbelags und *oneway* um eine Einschränkung auf Benutzung in eine Richtung zu modellieren.

Beziehungen

Die in der Tabelle fett markierten Schlüssel modellieren Beziehungen zu anderen Entitäten. *Postal_code* und *addr:postcode* können verwendet werden um Beziehungen zu Postleitzahlen herzustellen und alle Schlüssel mit dem Präfix *is_in* stellen Beziehungen Verwaltungseinheiten dar. Eine genauere Untersuchung dieser Beziehungen finden sich in den Abschnitten 6.2 und 6.3

4.2 Postleitzahlregionen

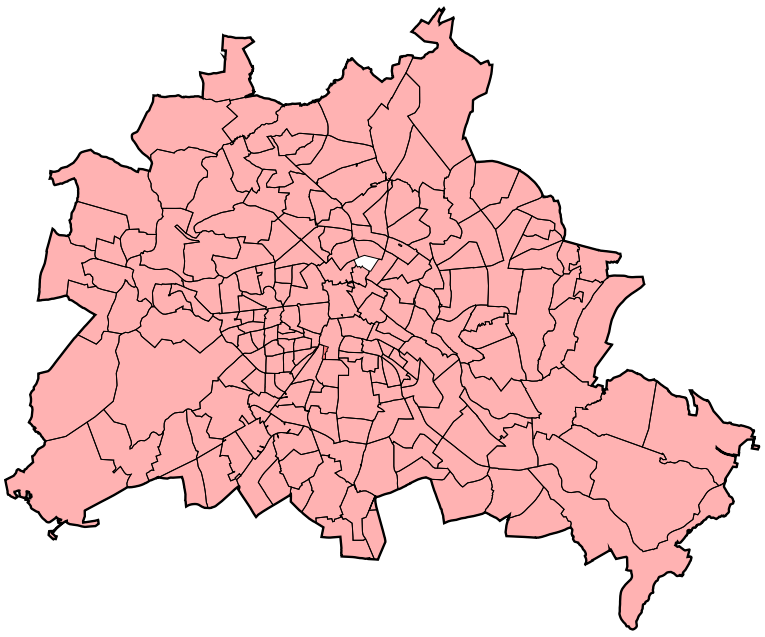


Abbildung 12: erfasste Postleitzahlregionen in Berlin

Eine Postleitzahlregion ist eine Postleitzahl mit einem zugehörigen Gültigkeitsbereich. Geometrisch wird dieser Bereich als *Region* bereitgestellt. Abbildung 12 zeigt beispielhaft die in Berlin erfassten Regionen.

Klassifikation

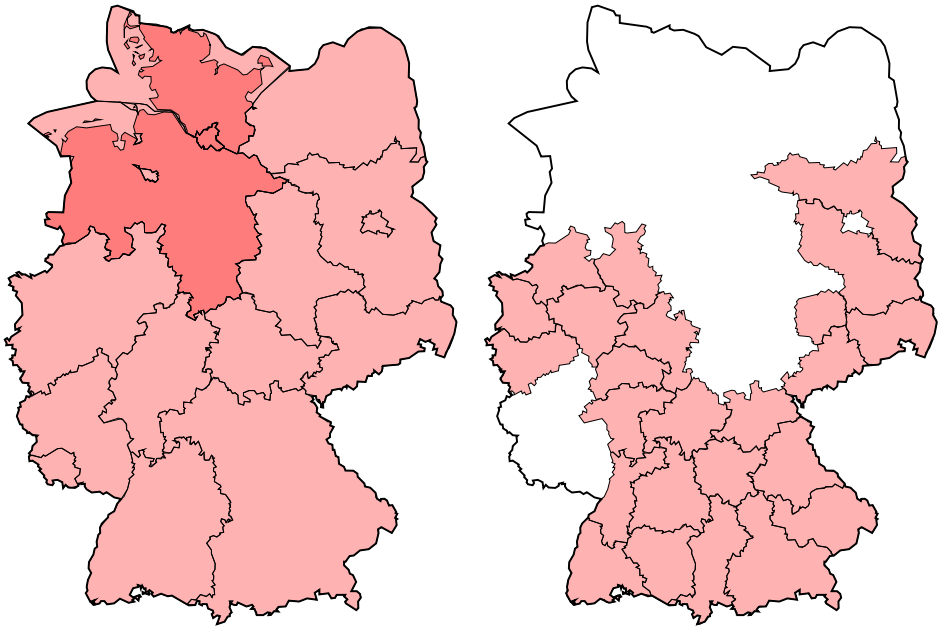
Regionen, die mit einem der Attribute *postal_code* oder *postcode* ausgezeichnet wurden, sind Postleitzahlregionen. Der Wert des Attributs gibt jeweils die Postleitzahl an.

Postleitzahlregionen werden zwar auch oft mit dem Attribut *boundary=postal_code* ausgezeichnet, aber zur eindeutigen Identifizierung reicht dieses Attribut nicht aus. Das liegt daran, dass ein Objekt nicht zwei Attribute mit dem selben Schlüssel haben kann. Der selbe Schlüssel wird aber verwendet, um Verwaltungseinheiten zu kennzeichnen und oftmals sind Postleitzahlregionen deckungsgleich mit diesen. In solchen Fällen wird in der Datenbank in der Regel nur ein Objekt angelegt (zur Vermeidung von Redundanz). Diesem Objekt können dann aber nicht die beiden Attribute *boundary=administrative* und *boundary=postal_code* gleichzeitig gegeben werden. Ansonsten haben die Postleitzahlregionen keine bemerkenswerten Attribute.

Situation in Deutschland

In Deutschland sind derzeit insgesamt 7 498 solcher Regionen erfasst. Nach Angaben der *deutschen Post*[4] gibt es in der BRD 8 259 Postleitzahlen, die Gebiete bezeichnen. Dementsprechend sind etwa 91% der Postleitzahlen erfasst. Gemessen an der Fläche decken die erfassten Postleitzahlen sogar knapp 93% von Deutschland ab. Abbildung 32 stellt die vorhandenen Regionen grafisch dar.

4.3 Verwaltungseinheiten



(a) Ebene 4 (b) Ebene 5

Abbildung 13: Verwaltungseinheiten in Deutschland

Bei Verwaltungseinheiten handelt es sich um Gebiete, die Zuständigkeiten bezüglich der öffentlichen Verwaltung definieren. Im Sinne der Openstreetmap fallen hierunter zunächst einmal die nationalen Staatsgrenzen. Die administrative Aufteilung gemäß der Organisation des jeweiligen Staats ist ebenfalls in Form von Verwaltungseinheiten erfasst. Beispiele aus Deutschland hierfür sind Bundesländer, Regierungsbezirke, Landkreise und Gemeinden. Diese Aufteilung variiert von Staat zu Staat und teilweise sogar innerstaatlich. Deutschland ist ein gutes Beispiel hierfür: So gibt es zwar flächendeckend Bundesländer,

[4] ttp://www.dp-dhl.com/de/presse/pressemitteilungen/2003/10_jahre_fuenfstellige_postleitzahl.html

Regierungsbezirke sind hingegen nur in Baden-Württemberg, Bayern, Hessen, Nordrhein-Westfalen und Sachsen vorhanden.

In vielen Staaten ist diese Aufteilung hierarchisch aufgebaut. Dieser Sachverhalt wird in der Openstreetmap durch die Einteilung in sogenannte *verwaltungsmäßige Ebenen* wiedergegeben. Um von den vielen unterschiedlichen Verwaltungssystemen dieser Welt zu abstrahieren, werden diese Ebenen nicht benannt, sondern lediglich durchnummeriert. Zur Übersetzung dieser Nummern in das jeweilig lokale Verwaltungsschema wird eine detaillierte Liste im Wiki geführt[5].

Klassifikation

Verwaltungseinheiten sind Regionen, die mit dem Attribut *boundary=administrative* markiert werden. Zusätzlich haben administrative Einheiten ein Attribut *admin_level*, welches den Wertebereich der natürlichen Zahlen im Intervall [1,11] hat. Dieses Attribut ordnet eine Verwaltungseinheit in die verwaltungmäßige Hierarchie ein.

Bemerkenswert ist, dass sich in der Openstreetmap eine Region immer genau auf einer administrativen Ebene befindet. Die Zuordnung ist über das Tag *admin_level=n* realisiert. Durch die Einschränkung, dass jeder Schlüssel pro Entität nur einmal verwendet werden kann, ist es nicht möglich, mehrere Werte für diesen Schlüssel zu vergeben. Von der Möglichkeit, im Wert des Tags mehrere Zahlen abzulegen (z.B. getrennt durch Semikolons) wird kein Gebrauch gemacht. Beispielsweise können deshalb Einheiten nicht gleichzeitig als kreisfreie Stadt und als Gemeinde verzeichnet werden (siehe Abschnitt 6.3.3).

Situation in Deutschland

Die Abbildungen 13, sowie 35 bis 40 zeigen die jeweilige administrative Ebene in Deutschland. Für jede vorhandene Region auf der jeweiligen Ebene wurde diese mit 30%-iger Deckkraft eingezeichnet, sodass Mehrfachabdeckungen sichtbar werden.

- Ebene 4 entspricht den Bundesländern. Auf Abbildung 13(a) ist einerseits zu erkennen, dass alle Bundesländer in der Datenbank vorhanden sind, zum anderen aber auch, dass Hamburg, Niedersachsen und Schleswig-Holstein doppelt vorhanden sind. Grund hierfür ist, dass man diese Bundesländer einmal inklusive und einmal exklusive des Hoheitsgewässers und der Anschlusszone eingepflegt hat. Obwohl Mecklenburg-Vorpommern ebenfalls am Meer liegt, wurde nur eine Region gefunden, weil die zugehörige zweite Region bewusst nicht mit dem Tag *boundary=administrative* versehen ist.
- Ebene 5 entspricht in Baden-Württemberg, Bayern, Hessen, Nordrhein-Westfalen und Sachsen den Regierungsbezirken (Abbildung 13(b)). Obwohl es in Brandenburg keine Regierungsbezirke gibt, existieren dennoch zwei Einheiten: diese sind Regionen auf Ebene 2 des sogenannten NUTS-Systems[6], einem europäischen Klassifikationsschema für administrative Regionen. Solche Einheiten existieren eigentlich auch in Niedersachsen und Rheinland-Pfalz, sind jedoch bisher nicht in der Openstreetmap enthalten.
- Ebene 6 umfasst Landkreise und kreisfreie Städte. Wie auf Abbildung 35 zu erkennen, ist Deutschland auf dieser Ebene – abgesehen von wenigen Ausnahmen – flächendeckend erfasst. Die folgenden Kreise oder kreisfreien Städte konnten entweder nicht gefunden werden oder die Regionen waren beschädigt und konnten daher nicht verwertet werden: Ostalbkreis, Göppingen (beides Baden-Württemberg), München, Nürnberger Land (beides Bayern), Münster (Nordrhein-Westfalen) und Hamburg.
- Die Bedeutung der Regionen auf Ebene 7 (Abbildung 36) ist nicht klar definiert. Es scheint sich jedoch um solche Verwaltungsgebiete zu handeln, die mehrere Gemeinden, meist des selben Kreises, in irgendeiner Form administrativ gruppieren. Verbünde von Gemeinden dieser Art werden je nach Bundesland unterschiedlich als *Samt- oder Verbandsgemeinden* oder als *Ämter* bezeichnet. Exem-

[5] http://wiki.openstreetmap.org/wiki/Key:admin_level
[6] http://de.wikipedia.org/wiki/NUTS

plare dieses Typs kommen aktuell in manchen Bundesländern gar nicht, in anderen jedoch gehäuft vor. Eine Abschätzung über die Vollständigkeit ist aufgrund der vagen Definition schwierig.

- Ebene 8 sind Gemeinden. Abbildung 37 macht deutlich, dass zwar weite Teile erfasst sind, aber andere Teile auch mangelhaft sind. So scheinen beispielsweise Bayern und Baden-Württemberg vollständig erfasst während in Niedersachsen noch weite Teile leer sind. Flächenmäßig sind insgesamt 87,7% abgedeckt. Insbesondere fehlen hier aber die größten Städte, weil diese schon als Stadtkreise oder Bundesländer erfasst sind.
- Die Ebenen 9-11 sind Stadtteile. Diese finden sich erwartungsgemäß nur in großen Städten wieder. Manche Städte haben verschiedene Ebenen von Stadtteilen, wozu dann die drei Level zur Differenzierung genutzt werden können. Offiziell unterscheiden sich die drei Level bezüglich verwaltungmäßigen Rechten. Dieser Sachverhalt wurde nicht genauer überprüft. Auch wenn viele große Städte mit Stadtteilen genauer zerlegt werden, fehlen auch einige. In Köln zum Beispiel sind keine Stadtteile erfasst, obwohl es dort welche gibt.

Die Vollständigkeit der Daten ist auf manchen Ebenen offenbar gut (4-6), auf anderen schwer zu beurteilen (7, 9-11), auf Ebene 8 jedoch teilweise mangelhaft. Eine genauere Interpretation der Objekte der Verwaltungsebenen findet im Abschnitt 6 statt.

4.4 Ortsmittelpunkte

Ortsmittelpunkte geben den Mittelpunkt eines Ortes an. Dieser Mittelpunkt ist dabei nicht notwendigerweise der geometrische Mittelpunkt eines Orts, sondern kann sich beispielsweise auf eine wichtige Stelle im Ort beziehen, wie beispielsweise das Rathaus oder ein Bahnhof. In der Praxis finden sich diese Punkte aber in der Nähe des geometrischen Mittelpunkts eines Ortes.

Klassifikation

Ortsmittelpunkte sind geometrisch Knoten und werden über ein Attribut mit dem Namen *place* ausgezeichnet, falls dieses einen der Werte *city, town, village, hamlet* oder *isolated_dwelling* hat. Damit bildet dieses Attribut auch gleichzeitig Klassen von Objekten diesen Typs. Wichtig ist hierbei, dass sich diese Zuordnung nicht auf das lokale Verständnis der Begriffe *Großstadt, Stadt* oder *Dorf*, wie die öffentliche Verwaltung sie definiert, bezieht, sondern sich allein aus der Bevölkerungszahl ergibt[7]:

Bevölkerungszahl	Kategorie	Wert des Attributs *place*
≥100 000	Großstadt	city
≥10 000	Stadt	town
≥100	Dorf	village
≥2	Weiler	hamlet
1-2	freistehende Häuser	isolated_dwelling
undefiniert	Stadt-/Ortsteil	suburb

Abbildung 14: Klassen von Ortsmittelpunkten

Das Attribut *place* kann noch andere Werte annehmen, wie beispielsweise *locality* oder *island*. Die entsprechenden Punkte, spielen hier jedoch keine Rolle und werden daher vernachlässigt. Ebenso finden im Folgenden Weiler, freistehende Häuser und die Knoten für Stadtteile keine weitere Beachtung.

Interpretation

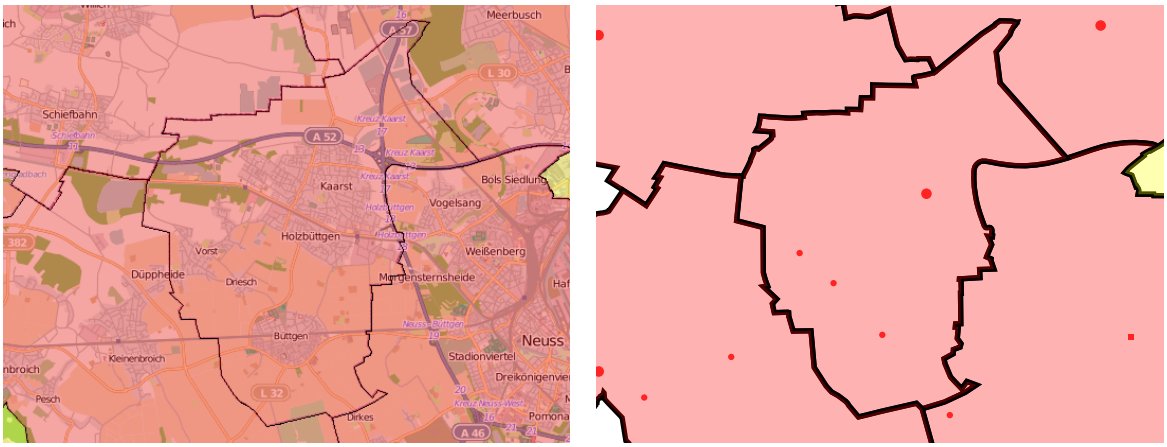
Orte im Sinne der Ortsmittelpunktsknoten und *Gemeinden* aus Sicht der öffentlichen Verwaltung sind nach unterschiedlichen Kriterien definiert. Ort in diesem Sinn ist jede menschliche Siedlung, deren Teile

[7] http://wiki.openstreetmap.org/wiki/Key:place

räumlich oder logisch zusammengehörig sind. Die weitere Kategorisierung richtet sich lediglich nach der Zahl der Einwohner eines solchen Orts. Gemeinden hingegen werden von der öffentlichen Verwaltung definiert und umfassen in der Regel mehrere Orte in diesem Sinn.

Tatsächlich fallen in Deutschland in der Regel mehrere Ortsmittelpunkte in eine Gemeinde. Dies zeigt sich schon an der Anzahl der erfassten Ortsmittelpunktsknoten. Während im Gemeindeverzeichnis des statistischen Bundesamtes 16 610 Gemeinden genannt werden[8], sind insgesamt 38 997 Ortsmittelpunkte allein des Typs Großstadt, Stadt und Dorf erfasst (Davon 90 Großstädte, 2 240 Städte und 36 667 Dörfer).

Für das Rendern von Karten stellt dies kein großes Problem dar. Möchte man die Ortsmittelpunkte jedoch zu anderen Zwecken benutzen, sollte dieser Unterschied zwischen Orten im Sinne der Ortsmittelpunktsknoten und Gemeinden klar sein. Das folgende Beispiel veranschaulicht diesen Unterschied.



(a) Mapnik mit Overlay

(b) Overlay

Abbildung 15: Kaarst im Landkreis Neuss bei Düsseldorf

Abbildung 15 zeigt die Gemeinde *Kaarst* im Landkreis Neuss in Nordrhein-Westfalen. Administrativ wird als *Gemeinde Kaarst* diejenige Region angesehen, die in der Abbildung durch eine durchgezogene, dunkle Linie dargestellt ist und als einzige Fläche in der Abbildung vollständig zu sehen ist.

In der Datenbank liegen vier Ortsmittelpunkte innerhalb Kaarsts vor. Abbildung 15(b) zeigt die vorliegenden Städte (große Kreise) und Dörfer (kleinere Kreise).

Abbildung 15(a) zeigt den selben Kartenausschnitt, wie er vom Renderer *Mapnik*[9] dargestellt wird, plus ein Overlay der Gemeindegrenzen. Die Ortsmittelpunkte werden hier einfach als Label dargestellt Innerhalb Kaarsts finden sich daher fünf Schriftzüge: *Büttgen, Driesch, Holzbüttgen, Kaarst* und *Vorst* (Holzbüttgen ist als Stadtteil vermerkt). Insgesamt präsentiert sich dem Betrachter des Kartenausschnitts hier ein sinnvolles Bild, obwohl mehr Orte verzeichnet sind, als es Gemeinen gibt.

Situation in Deutschland

Bemerkenswert ist das hohe Aufkommen dieser Ortsmittelpunktsknoten. Einen Eindruck über die Abdeckung verschafft eine Visualisierung der vorhandenen Knoten. In Abbildung 43 wurde für jede Großstadt ein roter, für jede Stadt ein blauer und für jedes Dorf ein grauer Punkt eingezeichnet.

Auftreten	Attribut
100,0%	addr:housenumber
89,2%	addr:street
77,5%	addr:city
75,1%	addr:postcode
63,5%	addr:country
10,8%	name
6,3%	amenity

(a) Gebäude (Knoten)

Auftr.	Attribut
100,0%	addr:housenumber
99,3%	building
95,2%	addr:street
69,5%	addr:city
68,2%	addr:postcode
61,8%	addr:country
30,4%	source

(b) Gebäude (Polygone)

Auftr.	Attribut
100,0%	addr:interpolation
24,2%	addr:street
16,7%	addr:postcode
16,2%	addr:city
9,8%	addr:country
5,4%	created_by
2,2%	addr:inclusion

(c) Interpolationen

Abbildung 16: Attribute der Objekte mit Adressinformation

4.5 Adressen

Adressen werden an verschiedenen Entitäten modelliert. Dazu werden den Objekten dann Attribute mit dem Präfix *addr:* gegeben. Die wichtigsten Attribute sind hierbei *addr:housenumber*, *addr:street*, *addr:city*, *addr:postcode* und *addr:country*.

Adressen finden sich hauptsächlich in den folgenden drei Formen:

1. An Knoten: der Knoten modelliert ein Gebäude, dessen Adresse über die oben genannten Attribute definiert wird.
2. An Streckenzügen:
   (a) hat ein Streckenzug das Attribut *addr:housenumber*, so handelt es sich wie bei 1. um ein Gebäude mit Adresse. Der Unterschied ist lediglich, dass die räumliche Ausprägung des Gebäudes über den geschlossenen Streckenzug modelliert ist.
   (b) hat ein Streckenzug das Attribut *addr:interpolation*, dann sind die Extrempunkte des Streckenzugs zwei Knoten, deren Hausnummer per Attribut kodiert ist. Der Verlauf des Streckenzugs gibt dann an, wie die Hausnummern zwischen diesen beiden Knoten interpoliert werden können.

Allen drei Methoden der Modellierung ist gemein, dass als elementare Einheit der Adresse das Gebäude mit seiner Hausnummer verwendet wird. Ergänzt wird die Adresse über die weiteren Attribute mit Präfix *addr:*. Abbildung 16 zeigt die jeweils häufigsten zur Anwendung kommenden Attribute mit ihrer relativen Häufigkeit.

Die Betrachtung dieser Häufigkeiten zeigt, dass die Adressattribute, die eine Hausnummer zur Adresse machen, nicht unbedingt vollständig erfasst werden. Bei Gebäuden (Knoten und Polygone) liegen die zusätzlichen Informationen aber immerhin an mehr als 60% der Objekte vollständig vor; sieht man von der Postleitzahl ab sogar an fast 70%. Etwas schlechter ist die Erfassung dieser Attribute jedoch bei den Interpolationen. Andererseits übertragen sich die Adressinformationen des Start- und Endknotens der Interpolation auf den gesamten Interpolationsstreckenzug. Unter Einbeziehung dieser Informationen sollte sich eine deutlich bessere Abdeckung der Zusatzinformationen für die Interpolationen erzielen lassen. Dies wurde aber nicht näher untersucht.

Situation in Deutschland

Es finden sich in Deutschland 720 000 Gebäudeknoten und 655 000 Gebäudepolygone mit Adressinformation. Andererseits wurden insgesamt 1,08 Millionen Straßen identifiziert. Rein rechnerisch kommt damit

[8] http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Navigation/Statistiken/Regionales/Gemeindeverzeichnis.psml

[9] der Renderer der die Karte auf http://www.openstreetmap.org erstellt

momentan nach dem Schubfachprinzip ungefähr eine Hausnummer auf jede identifizierte Straße. Geht man davon aus, dass es in der Realität in der Regel deutlich mehr als eine Hausnummern pro Straße gibt, ist das nicht sehr viel.

Zusätzlich gibt es 110 000 Interpolationsstreckenzüge. Damit könnten rein rechnerisch maximal etwa 10% der Straßen abgedeckt werden. Zu bedenken ist jedoch, dass in der Regel mindestens zwei Interpolationszüge nötig sind, um die Hausnummern entlang einer Straße zu beschreiben (z.B. für gerade und ungerade Hausnummern). Daher reduziert sich die theoretische Abdeckung auf maximal 5%.

5 Konzeption der Geokodierungskomponente

Wie in Abschnitt 2.6 beschrieben, kann die Funktionsweise eine Geokodierungskomponente in die Bestandteile *Eingabe*, *Ausgabe*, *Verarbeitungsalgorithmus* und *Referenzdatensatz* unterteilt werden. Der Fokus dieser Arbeit liegt auf der Bereitstellung eines Referenzdatensatzes auf Grundlage der Daten des Openstreetmap-Projekts. Dazu wird ein einfaches Datenmodell entwickelt, welches sich zur Aufnahme der vorhandenen Daten in strukturierter Form eignet (Abschnitte 5.1 und 5.2).

Ausgelassen wird das gesamte Themengebiet der Verarbeitungsalgorithmen, weshalb auf Themen wie Normalisierung, Standardisierung und Gewichtung der Eingabedaten, sowie das Matching mit dem Referenzdatensatz nicht näher eingegangen wird.

Dennoch soll eine rudimentäre Ein- und Ausgabe auf der Zielplattform möglich gemacht werden. Daher werden wichtige Nutzungsszenarien identifiziert, die auf Grundlage des Referenzdatensatzes implementiert werden sollen (Abschnitt 5.3).

Auf dieser Grundlage wird das Modell des Referenzdatensatzes fürs mobile Gerät in ein rein relationales Modell umgewandelt (Abschnitt 5.4).

5.1 Datenmodell des Referenzdatensatz

Gesucht ist ein Datenmodell, das die wichtigsten Komponenten von Adressen enthält und sich auf möglichst weite Teile der Welt verallgemeinern lässt. Wie in [DFB03] dargestellt wird, gibt es keinen universellen Standard zur Abbildung von Adressen. Dennoch finden sich manche Konzepte in den meisten Teilen der Welt wieder: Städte, Stadtteile, Straßen und Hausnummern. Darüber hinaus werden auch in weiten Teilen der Welt Postleitzahlen vergeben. Abbildung 17 zeigt, wie eine Adresse aus diesen Komponenten, zusammengesetzt werden kann. Anstelle der *Stadt* wird hier jedoch der Begriff der *Gemeinde* verwendet, da dieser etwas allgemeiner ist, weil er auch auf kleinere Ortschaften zutrifft.

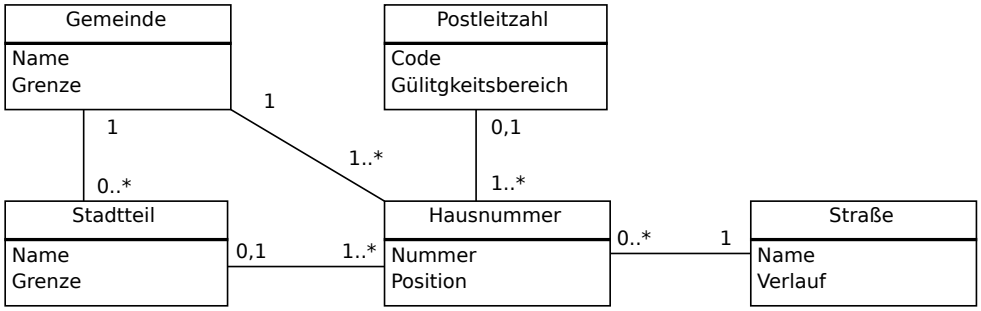


Abbildung 17: Angestrebtes Datenmodell für Adressen

Inwiefern sich dieses Modell tatsächlich auf andere Länder übertragen lässt, soll hier nicht genauer untersucht werden. Zumindest in Deutschland können mit diesen Bestandteilen aber Adressen modelliert werden. Als elementare Bestandteile werden die Gemeinde, der Stadtteil, die Postleitzahl, die Straße und die Hausnummer angesehen. Die Hausnummer ist dabei die zentrale Komponente, da sie die kleinste zu adressierende Einheit darstellt.

Dabei nehme ich folgende Beziehungen zwischen den Bestandteilen an:

- Eine Hausnummer gehört zu genau einer Straße, genau einer Gemeinde, höchstens einem Stadtteil und höchstens einer Postleitzahl.

- Eine Gemeinde muss keine Stadt- oder Ortsteile haben, aber kann aus beliebig vielen solchen bestehen. Ein Stadtteil gehört jedoch zu genau einer Gemeinde.
- Gemeinden, Stadtteile und Postleitzahlbereiche enthalten in der Regel mehrere Hausnummern.
- Zu einer Straße gehören in der Regel mehrere Hausnummern.

5.2 Vereinfachtes Modell ohne Hausnummern

Wie in Abschnitt 4.5 gesehen, ist die zu erwartende Abdeckung für Hausnummern mit etwa 5% sehr gering. Die Hausnummer wird daher aus dem Datenmodell entfernt. An die zentrale Stelle tritt stattdessen die Straße als nächstkleinere geographische Einheit, da diese nach der Hausnummer die beste räumliche Präzision bietet. Abbildung 18 zeigt das vereinfachte Modell.

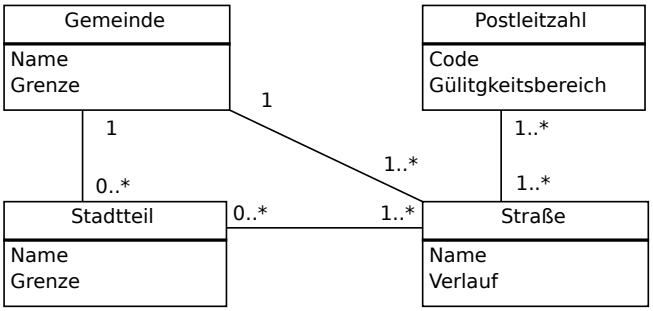


Abbildung 18: Vereinfachtes Datenmodell für Adressen

Statt der Hausnummer steht nun die Straße im Mittelpunkt des Modells. Folgende Beziehungen werden nun zwischen der Straße und den übrigen Entitäten angenommen:

- Eine Straße liegt in genau einer Gemeinde (Verbindungsstraßen werden somit als zwei unterschiedliche Straßen aufgefasst).
- Eine Straße liegt potenziell in keinem, einem oder mehreren Stadtteilen.
- Eine Straße liegt in keinem, einem oder mehreren Gültigkeitsbereichen von Postleitzahlen.
- Gemeinden, Stadtteil und Postleitzahlbereiche enthalten in der Regel mehrere Straßen.

5.3 Anforderungen zur Nutzung auf der Zielplattform

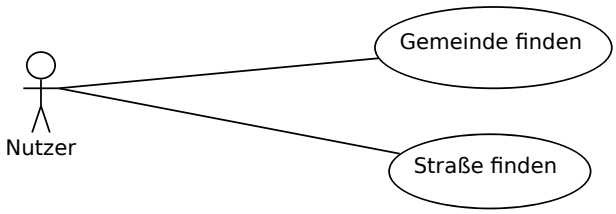


Abbildung 19: Anwendungsfälle der einfachen Geokodierungskomponente

Auf der Zielplattform soll rudimentär ein Geokodierungsprozess von Eingabe bis Ausgabe implementiert werden. Als Demonstrationskomponente der Softwarebibliothek von *mapsforge* gibt es den sogenannten *AdvancedMapViewer*, in den Basisfunktionalitäten der Geokodierungskomponente integriert werden

sollen. Im Hinblick darauf, dass diese Komponente in Zukunft mitunter die Funktion eines Navigationsgeräts einnehmen soll, bietet es sich an, dem Benutzer die folgenden Möglichkeiten zu eröffnen (siehe auch Abbildung 19):

- Auffinden einer Gemeinde: Der Benutzer sollte in der Lage sein, eine Gemeinde über ihren Namen ausfindig zu machen. Nachdem die Gemeinde textuell identifiziert wurde, sollte vom System die Position der Gemeinde bereitgestellt werden.
- Auffinden einer Straße: Der Benutzer bestimmt über geeignete textuelle Eingaben eine Adresse und die Position dieser wird abrufbar gemacht. Dies ist die wichtigste Anforderung an das System. Um aus einem potenziell enormen Datenbestand die gewünschten Ergebnisse schnell zugänglich zu machen sollte der Nutzer den Suchraum geeignet einschränken können. Des Weiteren sollten im Falle von Mehrdeutigkeiten geeignete Kriterien zur Disambiguierung bereitgestellt werden.

Nutzungsszenario

Abbildung 19 zeigt die zu beachtenden Anwendungsfälle. Diese werden in diesem Abschnitt genauer untersucht. Abbildung 20 zeigt, in welcher Weise der Benutzer ans Ziel gelangen könnte.

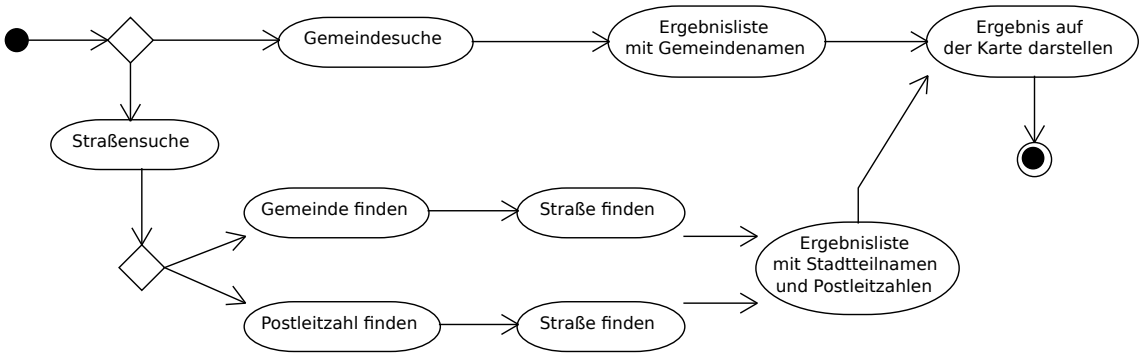


Abbildung 20: Einfache Nutzungsszenarien als Aktivitätsdiagramm

Der Benutzer entscheidet sich zunächst, wonach er suchen möchte. Er kann zwischen der Suche nach einer Gemeinde und der Suche nach einer Straße wählen.

- Suche einer Gemeinde: Der Benutzer wird um die Eingabe eines Namens gebeten. Nachdem die Suchanfrage abgesetzt wurde, wird eine Liste mit Suchergebnissen präsentiert aus der der Nutzer das gewünschte Ergebnis wählen kann.
- Suche nach einer Straße: Zunächst wird der Suchraum eingegrenzt, indem zunächst die Gemeinde oder die Postleitzahl gewählt wird. Im nächsten Schritt wird über den Namen der Straße im Bereich einer Gemeinde oder Postleitzahl gesucht.

Nicht-funktionale Anforderungen

Es ergeben sich zusätzlich eine Reihe nicht-funktionaler Anforderungen:

- Verwendbarkeit auf einem mobilen Gerät: Die Anwendung hat als Zielplattform ein mobiles Gerät. Zwar sind heutige Geräte der Zielplattform sehr leistungsstark, dennoch ergeben sich zusätzliche Anforderungen, im einzelnen sind dies:
  - Geringer Speicherverbrauch. Das Betriebssystem *Android* limitiert den Speicher jedes einzelnen Prozesses auf derzeit 24 MB. Dieser Speicher kann jedoch nicht komplett vom Geokodierer in Anspruch genommen werden, da er sich den Speicher mit den übrigen Komponenten teilt.

  - Akzeptable Laufzeiten der Anfragen. Es ist darauf zu achten, dass die Anfragen, die der Nutzer an das System stellt, keine längeren Wartezeiten produzieren.

- Offline-Nutzung. Eine Kernfunktionalität des Zielprogramms ist die Unabhängigkeit von einer Internetverbindung. Daher ist es zwingend erforderlich, dass auch die Suchfunktion rein lokal ausgeführt werden kann.
- Eingabehilfe (Autovervollständigung). Es wäre wünschenswert, dem Benutzer in geeigneter Weise eine automatische Vervollständigung seiner Eingaben zu ermöglichen, um den Komfort der Anwendung zu erhöhen.
- Erweiterbarkeit. Das System sollte in der Lage sein, an zukünftige Anforderungen anpassbar zu sein. Dazu gehört insbesondere das Erhöhen der Suchpräzision auf das Niveau von Hausnummern, sowie das Hinzufügen weiterer geokodierter Entitäten.

5.4 Relationales Modell

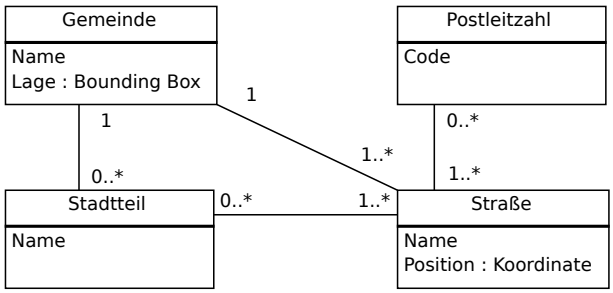


Abbildung 21: Angestrebtes relationales Modell auf der Zielplattform

Aufgrund von nicht-funktionalen Anforderungen wird das Modell aus Abschnitt 5.2 nochmals verändert. Resultat ist das einfachere relationale Modell aus Abbildung 21. Folgende Veränderungen werden vorgenommen:

- Einführung von künstlichen Primärschlüsseln für alle Entitäten. Somit können effizienteste Indexstrukturen verwendet werden.
- Relationale Modellierung der Beziehungen unter Verwendung der künstlichen Primärschlüssel. Dadurch werden Joins über räumliche Datenstrukturen vermieden, bei denen es sich um eine sehr teure Operation handelt[RSV02].
- Weglassen der geometrischen Bestandteile von Postleitzahlgebieten und Stadtteilen. Durch die relationale Modellierung der Beziehungen sind diese nicht mehr zwingend notwendig und es kann Speicherplatz gespart werden.
- Vereinfachung der Grenze von Gemeinden zu *enthaltenden Rechtecken* (Bounding Box). Da die Grenze der Gemeinde nicht mehr mit aufgenommen wird, sollte zumindest eine Bounding Box aufgenommen werden, um dem Nutzer die Region bildschirmfüllend anzeigen zu können.
- Vereinfachung des Straßenverlaufs auf einen zentralen Punkt der Straße. Dann muss der Straßenverlauf nicht abgespeichert werden und Speicherplatz wird gespart. Die Vereinfachung auf einen Punkt wird als akzeptabel angesehen, weil der tatsächliche Straßenverlauf für den Nutzer visuell erkennbar ist.

6 Extraktion, Transformation und Integration der Referenzdaten

Die Herausforderung dieser Arbeit besteht darin, einen hochwertigen Referenzdatensatz für die Geokodierung von Adressen bereitzustellen. Dazu soll das in Abschnitt 5.4 entwickelte Modell verwendet werden. Im Modell der Openstreetmap finden sich die Entitäten dieses Modells und deren gegenseitige Beziehungen jedoch nicht direkt wieder.

Es ist daher zu klären, wo die gesuchten Entitäten und Beziehungen in der Openstreetmap zu finden sind. Gesucht sind die Entitäten *Straße, Gemeinde, Stadtteil* und *Postleitzahl*, sowie die Beziehungen *Gemeinde - Stadtteil, Gemeinde - Straße, Stadtteil - Straße* und *Postleitzahl - Straße*. (siehe Abbildung 21)

Im Folgenden wird unter Bezugnahme auf die thematischen Entitäten aus Abschnitt 4 im einzelnen auf die gesuchten Entitäten und Beziehungen eingegangen. Dabei wird analysiert, welche Datenquellen sich eignen und welche nicht. Dann wird gezeigt, wie sich die gefundenen Daten in das gesuchte Schema integrieren lassen.

Dieser Abschnitt gliedert sich nach den Entitäten aus dem Modell des Referenzdatensatzes. Da Straßen im Mittelpunkt des Datenmodells stehen, wird mit diesen begonnen (Abschnitt 6.1). Dann werden nacheinander Postleitzahlen, Gemeinden und Stadtteile behandelt, wobei insbesondere auf die Beziehung der Straßen zu diesen eingegangen wird. (Abschnitte 6.2, 6.3 und 6.4).

6.1 Straßen

Straßen werden in der Openstreetmap netzwerkartig abgebildet, indem Straßensegmente, wie sie in Abschnitt 4.1 erläutert wurden, in der Datenbank abgelegt werden.

Die Abbildung der Straßen als Netzwerk hat seine Berechtigung, da diese Sicht auf die Straßen insbesondere beim Erstellen von Routinggraphen, aber auch beim Rendern von Karten sinnvoll genutzt werden kann.

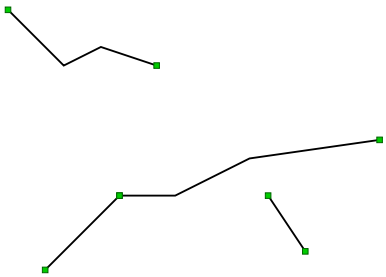


Abbildung 22: Mehrere Streckenzüge bilden eine Straße im postalischen Sinn

Zur Definition von Adressen verwenden Menschen jedoch eine andere Sicht auf dieses Straßennetzwerk, welche man als *postalische Sicht* bezeichnen könnte. Eine Straße im postalischen Sinn fasst mehrere Straßensegmente zusammen, die aus Sicht der Adressbildung nicht zu unterscheiden sind. Ausschlaggebend für die Zusammenfassbarkeit zweier Straßensegmente ist in erster Linie der Name einer Straße.

Straßensegmente werden also anhand ihres Namens zu Straßen im postalischen Sinn zusammengefasst. Der Name einer Straße ist jedoch zweifelsfrei nicht global eindeutig. Deshalb wird als zusätzliches Kriterium die zugehörige Gemeinde herangezogen, sodass nur die Segmente zusammengefasst, die zusätzlich zum Namen auch in ihrer Zugehörigkeit zu einer Gemeinde übereinstimmen. Dieses Vorgehen ist möglich, weil im Weiteren dafür gesorgt wird, dass jeder Straße (bzw. zunächst jedem Straßensegment) eine Gemeinde zugeordnet wird.

Die Praxis zeigt jedoch, dass selbst innerhalb von Gemeinden Straßennamen mehrfach vergeben werden. Dementsprechend sind weitere Kriterien zur Zusammenfassung notwendig. In Frage kämen zunächst Stadtteile und Postleitzahlen der Segmente. Problematisch an beiden Kriterien ist, dass beide im Weiteren nicht flächendeckend ermittelt werden können. Dadurch könnte eine Unterscheidung unterschiedlicher Straßen nach einem solchen Kriterium fehlschlagen.

Stattdessen wird die räumliche Komponente der Segmente in Betracht gezogen. Dazu wird angenommen, dass zusammengehörige Segmente räumlich adjazent sind. Eine Adjazenz im Sinne von direkter Verknüpfung der Streckenzüge der Segmente wird jedoch als zu starkes Kriterium angesehen. Das liegt daran, dass Teile einer Straße nicht notwendigerweise direkt zusammenhängen (Abbildung 22 zeigt eine solche Straße, die sich aus mehreren Streckenzügen zusammensetzt). Stattdessen werden zwei gleichnamige Segmente als zusammengehörig angesehen, wenn ihre Distanz einen gewissen Schwellwert unterschreitet. Die Wahl dieses Schwellwerts ist gewissermaßen willkürlich und wurde in der Implementierung mit 500 Metern konfiguriert. Eine genauere Beschreibung des Algorithmus findet sich in Abschnitt 7.1.7.

Die geometrische Repräsentation der Straßensegmente sind Streckenzüge oder einfache Polygone. Da mehrere solcher Komponenten zusammengefasst werden, ist der geometrische Typ der Straße daher eine Sammlung von verschiedenartigen geometrischen Objekten. Auch wenn hier teilweise polygonale Objekte vorkommen können, handelt es sich vorwiegend um Streckenzüge.

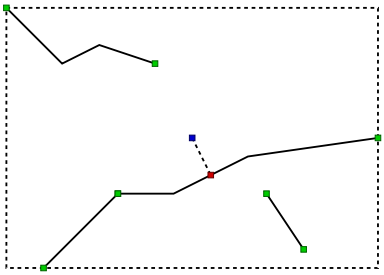


Abbildung 23: Zentroidsbestimmung für Straßen

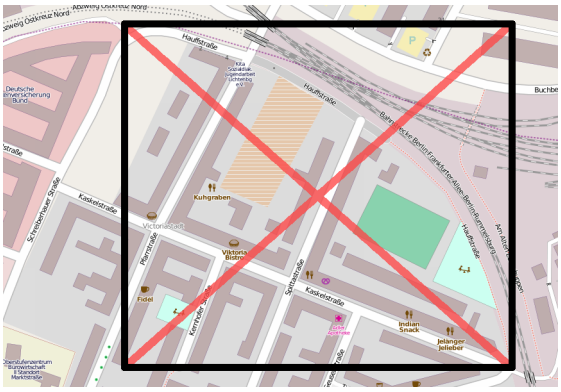


Abbildung 24: Ein ungünstig gelegener arithmetischer Mittelpunkt

Mittelpunkt

Das relationale Schema fordert für Straßen einen geographischen Mittelpunkt. Eine Möglichkeit der Bestimmung eines solchen Mittelpunkts stellt die Berechnung des arithmetischen Mittelpunkts der Bounding-Box dar. Dazu würde zunächst die Bounding-Box der Straße berechnet und ausgehend von dieser deren Mittelpunkt. Wenngleich einfach zu berechnen, kann dieses Vorgehen zu ungünstigen Ergebnissen führen, wenn dieser Mittelpunkt weit von der Straße entfernt liegt. Ein Beispiel hierfür ist die *Hauffstraße* in Abbildung 24. Hier ist die Bounding-Box (schwarzer Rahmen) und ihr Mittelpunkt zu erkennen. Dieser liegt deutlich von der Straße entfernt und schlimmer noch liegt der Punkt fast auf einer anderen, der

Spittastraße.

Es wird daher, ausgehend vom arithmetischen Mittelpunkt der Bounding-Box, ein auf der Straße gelegener Punkt ermittelt. Abbildung 23 zeigt, wie dieser berechnet wird. Der Punkt wird dabei so gewählt, dass es sich um denjenigen Punkt auf der Straße handelt, der am nächsten zum Zentrum der Bounding-Box der Straße gelegen ist. Dazu werden zunächst die Bounding-Box (gestrichelte Linie) und deren Mittelpunkt (blau) bestimmt. Nun wird derjenige Punkt auf der Straße ermittelt, der diesem Punkt am nächsten liegt (rot). In der Implementierung kommt zur Bestimmung des nächsten Punkts auf der Straße eine Bibliotheksfunktion von *JTS* zum Einsatz.

6.2 Postleitzahlen und die Beziehung Postleitzahl - Straße

Postleitzahlen und die Beziehung zu Straßen können über zwei orthogonale Arten modelliert werden. Diese unterscheiden sich darin, von welcher Seite her die Beziehung dargestellt wird und welche Bestandteile der involvierten geographischen Objekte zur Modellierung herangezogen werden:

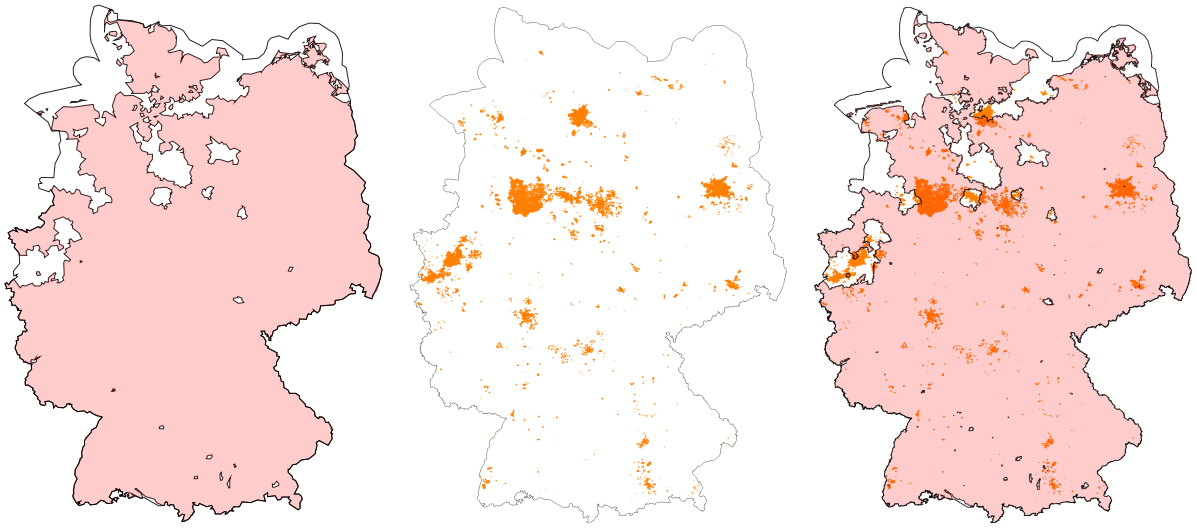
- **Regionsbasiert, nach Postleitzahlregionen:** Einerseits können die Postleitzahlregionen aus Abschnitt 4.2 verwendet werden. Die Menge der Postleitzahlen ist dann durch die vorhandenen Postleitzahlregionen festgelegt und die Beziehung zu Straßen wird dann implizit durch die Topologie bestimmt: Liegt eine Straße innerhalb einer solchen Postleitzahlregion, so wird diese Straße der jeweiligen Postleitzahl zugeordnet. Mittels dieser Regionen können 92,8% der Fläche Deutschlands abgedeckt werden. 89,5% der Straßen kann auf diese Weise eine Postleitzahl zugeordnet werden. Abbildung 25(a) zeigt die Fläche innerhalb Deutschlands, die mit Hilfe diesen Regionen abgedeckt werden kann.
- **Attributbasiert, nach Straßen:** Zum anderen kann einer Straße, wie in Abschnitt 4.1 beschrieben, über Attribute eine Postleitzahl zugeordnet werden. Während die Beziehung in diesem Fall explizit an der Straße modelliert ist, wird die Menge der Postleitzahlen implizit über diese Referenz induziert.

  Hierbei kommen Attribute mit zwei verschiedenen Schlüsseln zum Einsatz, die jedoch die gleiche Aufgabe erfüllen und lediglich aufgrund unterschiedlicher Konventionen nebeneinander existieren. Verwendet werden die Schlüssel *addr:postcode* und *postal_code*, wobei 0,4% bzw. 8,6% der Straßen in Deutschland ein solches Attribut haben. In Abbildung 25(b) ist für jedes Straßensegment, das mit einem der beiden Tags versehen ist, ein Punkt eingezeichnet worden. Hierbei wird deutlich, dass die Verwendung dieser Attribute stark gehäuft auftritt.

Diskussion und Bewertung

Generell ließen sich beide Verfahren nutzen um die gewünschte Information zu extrahieren. Obwohl beide Modellierungen keine vollständige Abdeckung bieten, ist die postleitzahlregionsbasierte mit knapp 93% flächenmäßiger Abdeckung bzw. 90% Abdeckung aller Straßen bereits sehr vollständig und damit praktisch einsetzbar. Prinzipiell könnten beide Datenquellen integriert werden, um eine noch höhere Abdeckung zu erreichen. Eine Überlagerung der beiden Bilder zeigt, dass so auch in Deutschland das Ergebnis noch verbessert werden könnte (Abbildung 25(c)).

Insbesondere in Ländern, wo die Abdeckung mit Postleitzahlregionen weniger gut ist, könnte ein Rückgriff auf die alternative Modellierungsmethode große Vorteile bringen. Konkret betrifft dies vor allem Hamburg und Teile von Nordrhein-Westfalen. Da die erreichte Abdeckung aber insgesamt als zufriedenstellend angesehen werden kann, wird in der Implementierung auf eine Integration beider Quellen verzichtet.



(a) regionsbasiert (b) attributbasiert (c) überlagert

Abbildung 25: Regionen, die Postleitzahlen definieren

6.3 Gemeinden und die Beziehung Gemeinde - Straße

Es wurden drei Möglichkeiten identifiziert, wie Gemeinden und deren Beziehung zu Straßen extrahiert werden können. Die drei Arten der Modellierung werden hier kurz angerissen und in den folgenden Abschnitten wird dann genauer auf die einzelnen Methoden eingegangen:

- Straßensegmente sind mit fremdschlüsselartigen **Sachattributen** versehen, deren Schlüssel den Präfix *is_in* haben. Beispiele hierfür sind etwa die Schlüssel *is_in* oder *is_in:city*. Der Wert des Attributs ist dann der Name administrativer Einheiten, denen ein Straßensegment zuzuordnen ist. Die Beziehung zu Gemeinden ist also an den Straßen selbst modelliert. Die Menge der Gemeinden ließe sich hieraus auch ableiten, als die Menge aller referenzierten Gemeindenamen. Aufgrund geringen Aufkommens und inkonsequenter Benennungsschemata wird schließlich auf die Einbeziehung dieser Informationen verzichtet.
- Eine Teilmenge der administrativen Regionen sind die **Gemeindegrenzen**. Bei dieser Modellierungsmethode definiert sich die Menge der Gemeinden über die vorhandenen Gemeinderegionen. Über ihre räumliche Lage lassen sich Straßen dann den Gemeinderegionen und somit auch den Gemeinden zuordnen. Als präziseste Art der Modellierung der Beziehung Gemeinde ↔ Straße wird primär diese Quelle zur Informationsgewinnung eingesetzt.
- **Ortsmittelpunktsknoten** modellieren ebenfalls Gemeinden. Nicht modelliert ist hier allerdings die räumliche Ausdehnung, da die Gemeinden auf einen Punkt reduziert werden. Daher ist es nicht ohne weiteres möglich, eine räumliche Beziehung zu Straßen herzustellen. Wo die Zuordnung mittels Gemeindegrenzen versagt, wird diese Quelle als Ausweichmöglichkeit verwendet.

Zum Einsatz kommt in der Implementierung eine Hybridlösung, die in erster Linie auf den Gemeindegrenzen aufbaut und im Zweifelsfall auf die Ortsmittelpunktsknoten zurückgreift.

6.3.1 Attribute der Straßensegmente

Zur Anwendung kommen hier Attribute mit den Schlüsseln *is_in*, *is_in:city*, *is_in:municipality*, *is_in:town* und *is_in:village*. Wie in Abbildung 11 zu erkennen war, sind die meisten dieser Attribute jedoch so selten, dass eine Verwendung von vornherein ausscheidet. Abgesehen vom Schlüssel *is_in* sind die Schlüssel dieser

Form an weniger als 0,1% der Straßensegmente vorhanden. Diese Attribute werden von den Betrachtungen daher ausgenommen.

Attribute mit dem Schlüssel *is_in* kommen jedoch an immerhin 0,8% der Straßensegmente vor. Typische Beispiele für die Werte dieser Attribute sind:

```
- "Dresden, Sachsen, Deutschland"
- "Sachsen, Germany, Europe"
- "Steinbach, Baden-Baden, Germany"
- "Wiesbaden-Naurod"
```

Hierbei handelt es sich zumeist um eine durch Kommata getrennte Aufzählung von Namen, deren genauere Bedeutung jedoch nicht offensichtlich ist. Somit ist auch nicht klar, welcher Teil dieser Zeichenkette die Gemeinde wiedergibt, falls diese überhaupt enthalten ist.

Es wäre also eine genauere Analyse notwendig, um diese Informationen im gewünschten Sinne weiterverarbeiten zu können. Auf diese wird jedoch verzichtet, da der Aufwand im Vergleich zum Gewinn (0,8% der Straßensegmente) vergleichsweise hoch sein dürfte.

6.3.2 Ortsmittelpunktsknoten

In Abschnitt 4.4 wurden die sogenannten Ortsmittelpunktsknoten diskutiert. Zwar wurde dort gesehen, dass diese eine Obermenge der Gemeinden darstellen, aber im Weiteren wird dieser Fakt vernachlässigt. Stattdessen wird davon ausgegangen, dass die Menge der Ortsmittelpunktsknoten benutzt werden könnte, um die Menge der Gemeinden zu definieren.

Ermittlung der Beziehung Gemeinde - Straße

Wählt man als Menge der Gemeinden die Ortsmittelpunktsknoten, so stellt sich die Frage, wie man Straßen den Gemeinden zuordnet. Die Ortsmittelpunktsknoten modellieren lediglich die Position einer Gemeinde, nicht ihre räumliche Ausdehnung. Da die Gemeinden also zu Punkten in der Ebene vereinfacht worden sind lässt sich nur eine Annäherung an die Realität für diese Zuordnung finden.

- Vorstellbar ist beispielsweise, dass eine Straße einfach dem räumlich nächstgelegenen Ortsmittelpunktsknoten zugeordnet würde. Das entspricht der Aufteilung der Ebene, die man über ein Voronoi-Diagramm erreicht [Aur91].

  In manchen Fällen ist eine solche Vereinfachung möglich. Abbildung 26 zeigt einen Ausschnitt Sachsen-Anhalts über den ein Voronoi-Diagramm auf Basis der Ortsmittelpunktsknoten gelegt wurde. Besonders auf dem Land, wo Dörfer oft räumlich etwas voneinander entfernt liegen, funktioniert eine Zuordnung dabei augenscheinlich relativ gut: Es ist zu erkennen, dass bei kleinen Orte, die durch viel Landschaft voneinander getrennt sind, eine sinnvolle Zuordnung der entsprechenden Straßen möglich ist.

  Zu erkennen ist aber auch, dass die Zuordnung an anderen Stellen schlecht funktioniert. Beispielsweise betrachte man die Zelle des Voronoi-Diagramms, die für die *Lutherstadt Wittenberg* erstellt wurde: Diese endet im Westen deutlich vor dem Ende der Stadt. Der westlichste Teil der Stadt befindet sich stattdessen in einer Zelle, die dem Dorf *Kienberge* zugeordnet ist.

- Unter Einbeziehung der Kategorien der Ortsmittelpunktsknoten könnte eine Gewichtung in die Zuordnungsfunktion eingebracht werden. Dann würde beispielsweise einer Großstadt ein größeres Gewicht zukommen als einem Dorf. Umsetzbar wäre dies zum Beispiel mit gewichteten Voronoi-Diagrammen wie den *Power-Diagrammen* [Aur91].

- Statt mit einer Zerlegung der Ebene zu arbeiten, könnte einem Ort jede Straße in einem bestimmten Radius zugeordnet werden. Auch hier könnte eine Gewichtung nach Kategorien mit eingebracht

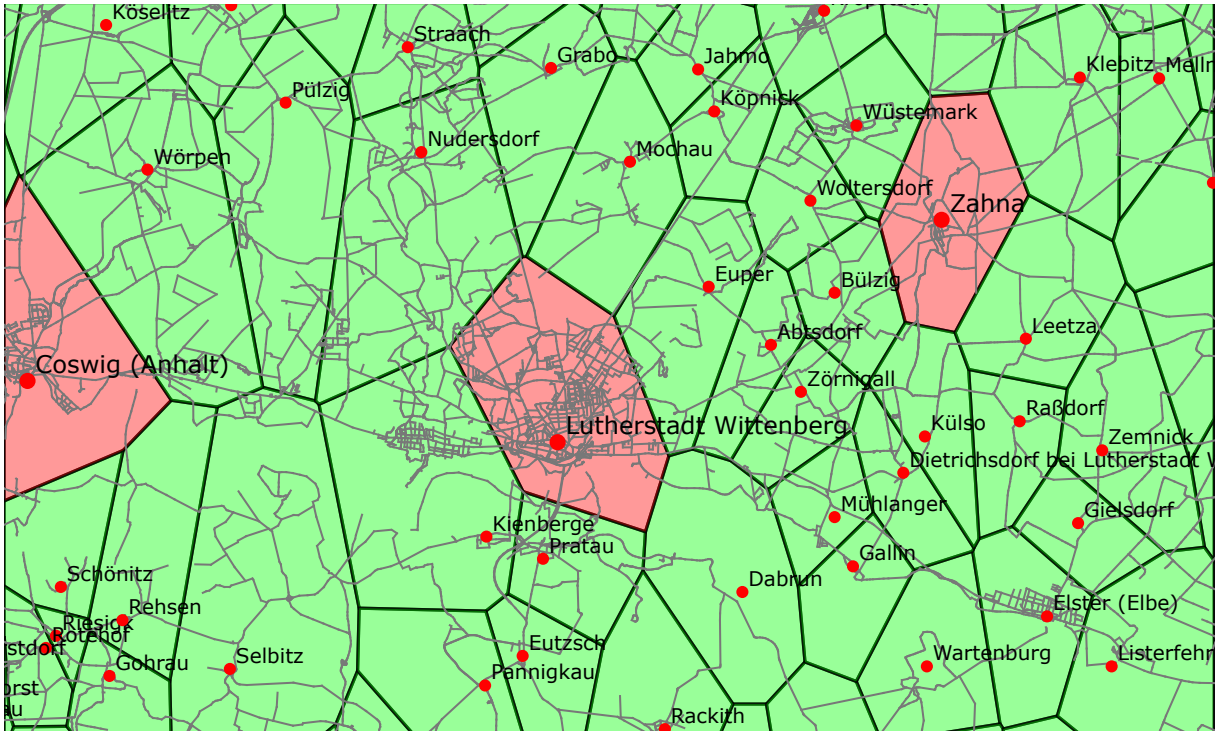


Abbildung 26: Zuordnung von Straßen zu Ortsmittelpunkten in Sachsen-Anhalt bei Wittenberg

werden. Dies könnte das Problem umgehen, dass eine Stadt "abgeschnitten" wird, wie es im obigen Beispiel mit *Wittenberg* passiert. Jedoch bekäme man auch neue Probleme, wie eine mehrfache Abdeckung von Straßen durch Grenzregionen und damit Mehrdeutigkeiten in der Zuordnung von Straßen zu Gemeinden.

6.3.3 Gemeinderegionen

Dies ist die präziseste Modellierungsmethode für Gemeinden, da hier eine exakte Grenze festgelegt wird, die eine Gemeinde ausmacht. Die Beziehung zu Straßen ist dann implizit über die räumliche Lage von Gemeindegrenzen und Straßen gegeben. Problematisch ist jedoch, dass Gemeinden nicht eindeutig als solche identifizierbar sind.

Identifizierung der Gemeinden

Wie in Abschnitt 4.3 gesehen, werden Verwaltungseinheiten in verschiedenen Ebenen bereitgestellt, die in Deutschland folgende Bedeutung haben:

- Ebene 4: Bundesländer
- Ebene 6: Landkreise / Kreisfreie Städte
- Ebene 8: Gemeinden

Das Problem ist hierbei, dass sich die Gemeinden auf diese drei Ebenen verteilen: Auf Ebene 8 finden sich ausschließlich Gemeinden, die beiden Stadtstaaten Berlin und Hamburg befinden sich aber auf Ebene 4 und in Ebene 6 sind sämtliche kreisfreien Städte, und damit die meisten Großstädte, enthalten.

Da aufgrund der Attribute nicht entschieden werden kann, bei welchen Objekten es sich tatsächlich um Gemeinden handelt, werden folgende Möglichkeiten gesehen, mit der Situation umzugehen:

- Behandlung aller Objekte auf den Ebenen 4 und 6, als wären es Gemeinden. Dann hätte man jedoch sämtliche Bundesländer und die Landkreise mit unter den Gemeinden. Bei der Zuordnung der Straßen zu Gemeinden würde dies massiv zu Mehrfachzuordnung führen. Diese Möglichkeit wird daher verworfen.
- Manuelle Selektion der Gemeinden aus den Ebenen 4 und 6. Dies würde den Verarbeitungsprozess schwer wartbar machen, da dafür gesorgt werden müsste, dass Veränderungen an den Daten aus der Openstreetmap korrekt verarbeitet werden. Außerdem würde eine Ausweitung des Prozesses auf andere Länder wahrscheinlich mit großem Aufwand verbunden sein, da auch hier eine manuelle Selektion stattfinden muss. Daher wird auch diese Idee verworfen.
- Entwicklung einer Heuristik, die aufgrund von Faktoren aus der Datenbank ermittelt, ob es sich bei einem Objekt auf Ebene 4 oder 6 um eine Gemeinde handelt. Diese Vorgehensweise wird im Folgenden weiter entwickelt.

Heuristik zur Identifizierung von Gemeinden

Es wurde also eine Heuristik gesucht, um zu bestimmen, ob es sich bei einer administrativen Einheit auf Ebene 4 oder 6 um eine Gemeinde handelt oder nicht. Anders formuliert geht es darum, Gemeinden von Bundesländern bzw. Landkreisen zu unterscheiden.

Die gefundene Heuristik basiert darauf, dass es für Gemeinden noch eine zweite, orthogonale Modellierung in Form der Ortsmittelpunktsknoten gibt. Die Idee ist die, dass sich Gemeinden von Bundesländern und Landkreisen in der Anzahl der enthaltenen Ortsmittelpunktsknoten unterscheiden sollten. Ein Bundesland, das kein Stadtstaat ist, sollte beispielsweise tendenziell viele Ortsknoten enthalten, während ein Stadtstaat eigentlich nur einen enthalten dürfte. Ähnlich verhält es sich mit Land- und Stadtkreisen: Ein Landkreis enthält viele, ein Stadtkreis einen oder wenige Ortsknoten.

Der Algorithmus funktioniert so:

- Bestimme für jedes Objekt A auf Ebene 4 und 6 die Anzahl der enthaltenen Ortsmittelpunktsknoten des Typs *Großstadt* g, *Stadt* s und *Dorf* d.
- Entscheide aufgrund von g, s und d anhand von Schwellwerten, ob es sich bei A um eine Gemeinde handelt oder nicht.

Das Ergebnis der Heuristik lässt sich dabei über die verwendeten Schwellwerte steuern. In der Implementierung wird momentan das folgende Auswahlkriterium verwendet:

$$g + s \leq 1 \tag{1}$$

Eine manuelle Überprüfung der Ergebnisse hat folgendes ergeben:

- Die beiden Stadtstaaten Berlin und Hamburg werden korrekt aus der Menge der Bundesländer ausgewählt.
- Von den 111 kreisfreien Städten in Deutschland werden 106 korrekt als solche identifiziert. Darüber hinaus werden vier falsch positive und drei falsch negative gefunden. Die verbleibenden zwei zu identifizierenden Stadtkreise sind zur Ausführungszeit nicht valide in der Datenbank vorhanden gewesen.

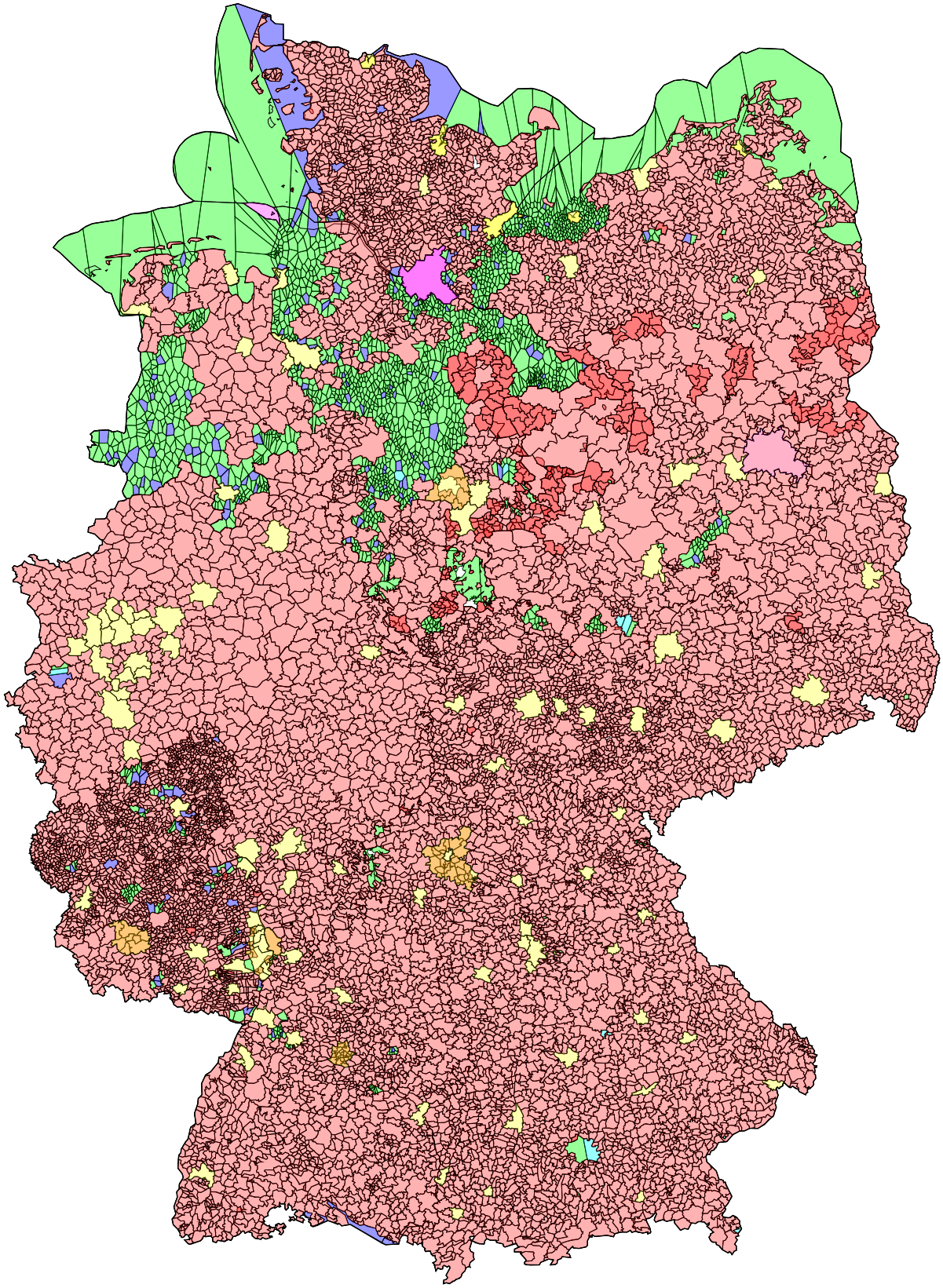


Abbildung 27: Partitionierung in Gemeinden nach Gemeinderegionen (rot, gelb und magenta) und Ortsmittelpunkten (grün, blau und türkis)

6.3.4 Hybridlösung mit Gemeindegrenzen und Ortsmittelpunkten

Da mit der attributbasierten Variante nur eine vergleichsweise geringe Abdeckung erzielt werden kann, wird von der Verwendung dieser Methode abgesehen. Die Verwendung von Grenzregionen wird aufgrund seiner Präzision als sinnvollste Variante eingestuft und wird daher als primäre Lösung eingesetzt. Mit dieser wird eine flächenmäßige Abdeckung von 91,9% erreicht. Abbildung 41 zeigt die Abdeckung, die mit Hilfe der Verwaltungseinheiten erreicht wird. Für die verbleibende Fläche wird anhand der dort vorhandenen Ortsmittelpunktsknoten eine Zerlegung der Ebene in Voronoi-Zellen geschaffen, mit deren Hilfe eine Abdeckung von 100% erreicht wird. Da dieses Vorgehen lediglich als Übergangslösung betrachtet wird, wird die Zerlegung der Ebene in einfache Voronoi-Zellen verwendet, da hierfür ein geeigneter, effizienter Algorithmus gefunden werden konnte.

Das Gesamtergebnis der Zerlegung in Gemeinden ist in Abbildung 27 zu sehen: Gemeinden, die aus administrativen Einheiten hervorgegangen sind, sind rot (Ebene 8, Gemeinden), gelb (Ebene 6, Stadtkreise) oder magenta (Ebene 4, Bundesländer) eingefärbt. Die Gemeinden, die aus Ortsmittelpunkten hervorgegangen sind, sind grün (Dörfer), blau (Städte) oder türkis (Großstädte) eingefärbt. Manche Regionen sind doppelt abgedeckt, was in der Grafik anhand der entsprechenden Mischtöne zu erkennen ist.

Bewertung

Es wurde vom statistischen Bundesamt eine Liste der Gemeinden in Deutschland bezogen. Von den 200 größten Städten Deutschlands konnten 186, also 94%, in Form von Verwaltungsgrenzen als Gemeinden identifiziert und extrahiert werden. Die 14 verbleibenden Gemeinden werden über das Ortsmittelpunktsverfahren ebenfalls abgedeckt.

Auch wenn die ermittelten Gemeindegrenzen nicht mit einem kommerziellen Datensatz abgeglichen wurden, so legt diese einfache Überprüfung dennoch nahe, dass eine sehr umfassende und sinnvolle Abdeckung der Testregion erzielt werden konnte. Unter der Annahme, dass die erfassten Grenzen die Gemeinden akkurat wiedergeben, sollten sich aus den meisten enthaltenen Straßen korrekte Adressen bilden lassen.

6.4 Stadt- / Ortsteile

Stadtteile werden aus den Verwaltungseinheiten abgeleitet. Auf den Ebenen 9 bis 11 befinden sich dort die Stadtteile. Diese werden ausnahmslos und ohne weitere Verarbeitung in den Referenzdatensatz übernommen.

7 Implementierung

Ziel der Implementierung ist die Herstellung einer Komponente für die Adresssuche im Rahmen der Softwarebibliothek des mapsforge-Projekts.

Die Implementierung (Abbildung 28) gliedert sich in diese beiden Teile, die im folgenden separat behandelt werden:

- Herleitung des Referenzdatensatzes. In diesem Vorverarbeitungsprozess werden die Referenzdaten für die Verwendung auf der Zielplattform aus den Ausgangsdaten gewonnen.
- Bereitstellung einer API und einer Benutzerschnittstelle für die Adresssuche auf der Zielplattform.

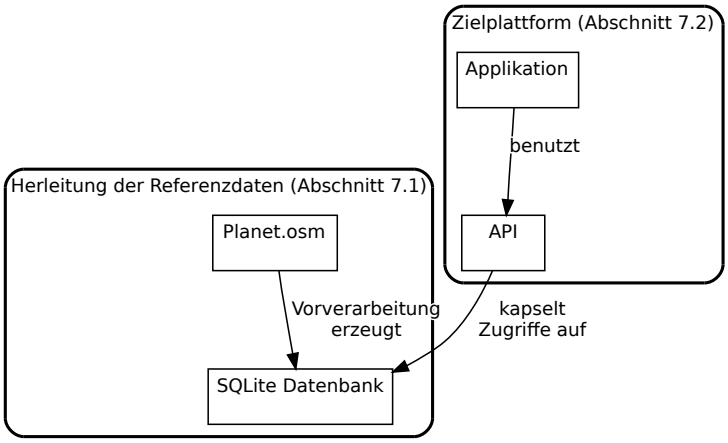


Abbildung 28: Komponenten der Implementierung

7.1 Herleitung der Referenzdaten

Die Herleitung des Referenzdatensatzes hat als Eingabe die *Quelldaten* und eine *Zielregion* und als Ausgabe eine gefüllte Datenbank in dem Schema wie es Abschnitt 5.4 vorgestellt wird. Die folgenden Teilprobleme wurden gelöst:

- Extraktion der in Abschnitt 4 benannten geographischen Objekte.
- Transformation in die Entitäten des Referenzdatensatzes aus Abschnitt 5.3.
- Herleitung der Beziehungen gemäß Abschnitt 6.
- Erstellen des Referenzdatensatzes gemäß dem relationalen Modell aus Abschnitt 5.4.

Zunächst werden in den Abschnitten 7.1.1 bis 7.1.4 generelle Merkmale der Implementierung vorgestellt. Ab Abschnitt 7.1.5 folgt dann die Umsetzung des Transformationsprozesses, deren Teilschritte auf Abbildung 29 gezeigt sind.

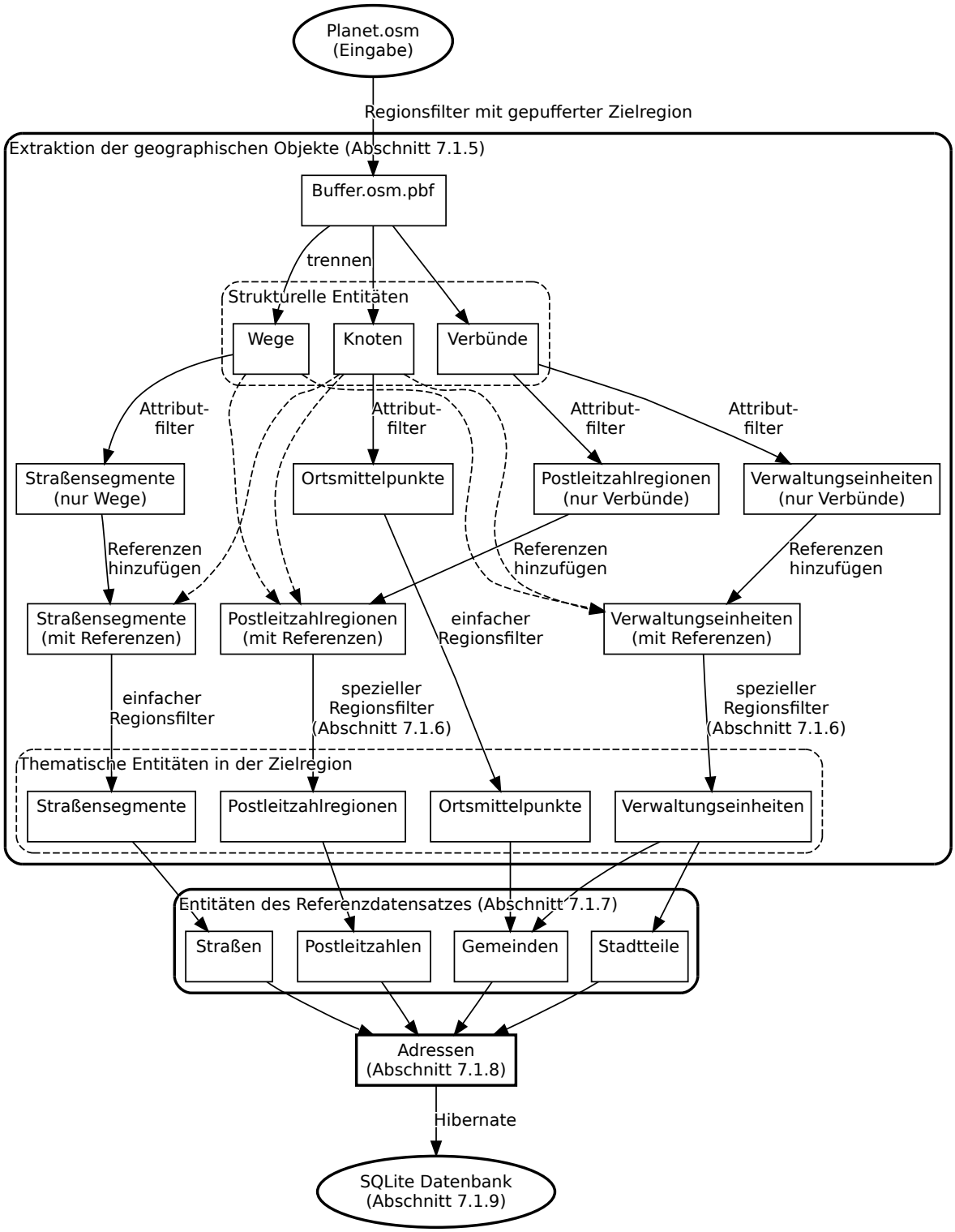


Abbildung 29: Herleitung der Referenzdaten: aus dem Abbild der Openstreetmap-Datenbank wird über mehrere Zwischenschritte eine SQLite-Datenbank erzeugt.

7.1.1 Verarbeitung der Eingabedaten

Als Quelldaten dient die Datenbank des Openstreetmap-Projekts. Diese Datenbank ist in Form einer komprimierten XML-Datei erhältlich. Die Weiterverarbeitung kann somit zunächst nur dateibasiert erfolgen.

Das Projekt stellt mit dem Programm *Osmosis* eine Software zur Verfügung, mit der die Ausgangsdateien datenstromorientiert gelesen, verarbeitet und in verschiedene Formate überführt werden können. Die wichtigsten Ausgabeformate sind:

- PostgreSQL mit PostGIS-Erweiterung.
- Ein Binärformat, das auf *protocol-buffers* aufbaut.

Das Überführen in eine um GIS-Funktionalität erweiterte Datenbank wie PostgreSQL mit PostGIS scheint zunächst die attraktivste Alternative zu sein. Damit könnten alle benötigten Verarbeitungsschritte in einer Datenbankanfragesprache formuliert werden und das Datenbankmanagementsystem würde sich um die effiziente Ausführung der Anfragen kümmern. Erste Tests mit diesem Vorgehen zeigen jedoch ein ernüchterndes Bild:

- Es dauert viele Stunden, eine Region wie Deutschland in die Datenbank zu importieren.
- Die geometrische Funktionalität der in die Datenbank importierten Daten beschränkt sich auf Punkte und Linien. Die für die Verarbeitung benötigten Regionen werden nicht als geometrische Objekte in das Datenbankschema aufgenommen.

Die Arbeit mit dem Binärformat ist im Vergleich zur Datenbank-Variante leichtgewichtig im Bezug auf Ausführungszeiten. Die Daten der kompletten Welt können auf dem Testsystem in 18 Minuten komplett gelesen werden. Trotzdem hat auch diese Herangehensweise seine Nachteile:

- Die Verarbeitung ist datenstromorientiert und damit relativ unflexibel.
- Auch hier fehlt eine Unterstützung von Regionen als geometrischer Typ.

Trotzdem lassen sich gewisse Anfragen sehr effizient bearbeiten: Dazu zählt vor allem die attributbasierte thematische Selektion. Dadurch lässt sich die Größe des zu betrachtenden Datensatzes für ein Teilproblem oftmals schnell reduzieren. Es wurde sich daher für einen dateibasierten Ansatz und damit grundsätzlich für die Arbeit mit Osmosis entschieden.

Tatsächlich passiert die Verarbeitung der Eingabedaten einerseits mit Osmosis wie es in Abschnitt 7.1.2 vorgestellt wird. Dazu werden teilweise vorgefertigte Verarbeitungsmechanismen verwendet, aber es wurden auch neue Verarbeitungsschritte in Form von Plugins implementiert. Andererseits wurden Teile der Verarbeitung imperativ unter Verwendung der in Abschnitt 7.1.3 vorgestellten Techniken umgesetzt.

Ein besonderes Augenmerk liegt dabei immer auf der effizienten Ausführung der notwendigen Operationen. Der gesamte Extraktionsprozess gliedert sich in die Ausführung einer Reihe von kleineren Programmen, die Teilprobleme lösen. Charakteristisch ist daher die Herstellung von Dateien, die Zwischenergebnisse enthalten. Diese Zwischenergebnisse dienen dann als Eingabe für weitere Programme. Dadurch kann oftmals sehr viel *Overhead* vermieden werden, sodass sich insgesamt eine effizientere Ausführung der Vorverarbeitung ergibt.

7.1.2 Arbeit mit Osmosis

Osmosis ist ein in Java verfasstes Programm (bzw. auch eine Softwarebibliothek), welche folgende Bestandteile hat:

- Leseadapter für verschiedene Datei- und Datenbankformate
- Verarbeitungsmethoden
- Schreibeadapter für verschiedene Datei- und Datenbankformate.

Im Weiteren beschränken sich die Ausführungen auf dateibasierte Zugriffe. Das Programm hat eine *Pipeline*-Architektur, mit der sich verschiedene Eingaben mit Verarbeitungsmethoden und Ausgabemethoden kombinieren und zu komplizierteren Gesamtverarbeitungsschritten zusammenfügen lassen. Bezüglich der Eingabedaten arbeitet Osmosis *streamorientiert*, das heißt jede Datei wird sequenziell von vorne nach hinten gelesen und es kann im Datenstrom weder vor- noch zurückgespult werden. In den Dateien liegen die *strukturellen Daten* (vgl. Abschnitt 3.1) in der folgenden Reihenfolge vor:

1. Knoten
2. Wege
3. Verbünde

Wobei die Objekte innerhalb ihres jeweiligen Abschnitts nach ihren *Identifiern* aufsteigend sortiert vorliegen. Diese beiden architektonischen Eigenschaften wirken sich in der Arbeit mit Osmosis stark auf die Möglichkeiten aus, die der Programmierer in der Implementierung neuer Verarbeitungsschritte hat. Dies hat dann besondere Bedeutung, wenn es aufgrund der Datenmenge nicht mehr möglich ist, alle Daten in den Hauptspeicher zu laden.

Es stehen über *Osmosis* die folgenden relevanten Verarbeitungsmethoden zur Verfügung:

- Attributfilter: Diese Verarbeitungsmethode kann zur Extraktion von Themes benutzt werden. Dabei können die Objekte aufgrund ihres strukturellen Entitätstyps und aufgrund ihrer Sachattribute gefiltert werden.
- Regionsfilter: Diese Verarbeitungsmethode filtert Objekte heraus, die räumlich in einer gegebenen Region liegen. Zur Evaluation der geometrischen Prädikate kommt hier die *Java2D*-API zum Einsatz. Im weiteren wird zwischen dem *einfachen* und dem *vervollständigenden* Regionsfilter unterschieden. Eine Differenzierung dieser beiden Modi findet im Beispiel im nächsten Absatz statt.

Daneben kann Osmosis über Plugins erweitert werden, um eigene Datenmanipulationen durchzuführen.

Ein Beispiel für die Auswirkungen der unflexiblen Architektur von Osmosis

Die Implementierung des Regionsfilters ist ein gutes Beispiel für den starken Einfluss, den die Architektur von Osmosis und die des Dateiformats auf das Design der Verarbeitungsmethoden und damit auf die Performance ihrer Ausführung haben.

Der Regionsfilter arbeitet zunächst folgendermaßen (im weiteren als *einfacher* Modus bezeichnet):

- Lies alle Knoten. Für jeden Knoten entscheide, ob dieser in der Zielregion liegt und gib ihn dementsprechend an die Ausgabe weiter. Merke in einer geeigneten Datenstruktur N den Identifier jedes in der Pipeline weitergereichten Knotens.

- Lies alle Wege. Entscheide über die Weiterreichung des Objekts anhand dessen, ob einer der referenzierten Wegknoten in *N* vorhanden ist. Merke alle Identifier der weitergereichten Wege in einer Datenstruktur *W*.
- Lies alle Verbünde. Entscheide über die Weiterreichung anhand des Vorhandenseins der Mitglieder des Verbunds in *N* oder *W*.

Mit diesem Vorgehen lässt sich jedoch nicht sicherstellen, dass alle von Wegen referenzierten Knoten, bzw. alle von Verbünden referenzierten Knoten, Wege und Verbünde in der Pipeline durchgereicht werden (Wegen der Reihenfolge der Objekte nach Entitätstyp). Falls eine derartige Integrität der Ausgabedaten notwendig ist, kann der Regionsfilter entsprechend konfiguriert werden. Dies führt jedoch zu folgendem Verhalten des Regionsfilters (im Weiteren als *vervollständigender* Modus bezeichnet):

- Ermittle wie vorher die durchzureichenden Objekte, aber gib diese noch nicht an die Ausgabe weiter, sondern:
  - Serialisiere *jeden gelesenen* Knoten in einer temporären Datei *SN*.
  - Serialisiere *jeden gelesenen* Weg in einer temporären Datei *SW*.
  - Serialisiere *jeden gelesenen* Verbund in einer temporären Datei *SR*.
- Entscheide am Ende des Datenstroms aufgrund von *SN, SW* und *SR*, sowie der in der Zielregion enthaltenen Knoten, welche Objekte an die Ausgabe weiterzureichen sind.

Diese Verarbeitung führt zu *deutlich* verlängerten Ausführungszeiten, die nicht mehr im akzeptablen Bereich liegen.

7.1.3 Paradigmenwechsel und Osmosis als Bibliothek für Lese- und Schreiboperationen

Osmosis ist in Java geschrieben und bildet grundsätzlich eine Abstraktionsschicht bezüglich des Zugriffs auf die strukturellen Daten der Openstreetmap. Damit kann von der Repräsentation der Daten in verschiedenen Dateiformaten abstrahiert werden. Es bindet den Nutzer jedoch an das Paradigma der datenstromorientierten Verarbeitung für die Implementierung neuer Datenmanipulationsmechanismen.

Um mit iterativen Zugriffsmechanismen auf den Daten operieren zu können, wurden entsprechende Adapter implementiert, die auf Osmosis aufbauen und folgende Zugriffsmuster für einen Datensatz anbieten:

- Datensatz: Diese Methode liest eine komplette Datei in Datenstrukturen, die im Hauptspeicher liegen, und stellt diese dem Nutzer im Anschluss zu Verfügung. Diese Methode eignet sich immer dann, wenn davon ausgegangen werden kann, dass alle Daten im Speicher gehalten werden können.
- Iterator: Mit dieser Methode kann über die strukturellen Objekte einer Datei iteriert werden. Diese Methode bietet sich an, wenn damit zu rechnen ist, dass nicht alle Daten im Speicher gehalten werden können.

Dieser Paradigmenwechsel löst nicht in jedem Fall die Probleme, die durch die Reihenfolge der Objekte gegeben sind. Dennoch erlauben sie eine anwendungs- oder aufgabenzentrierte Entwicklung von Programmen. Im Gegensatz dazu müssen Plugins für Osmosis immer datenstromorientiert geschrieben werden.

7.1.4 Vom strukturellen zum geographischen Modell

Um geometrische Operationen ausführen und Prädikate auf den geographischen Objekten auswerten zu können, ist es notwendig, von den *strukturellen* Objekten zu *geographischen* Objekten zu kommen.

Dazu ist es notwendig, die geometrische Komponente der Objekte als abstrakte Datentypen zur Verfügung zu stellen. Als Softwarebibliothek für geometrische Datentypen wird hier auf die *Java Topology Suite (JTS)* zurückgegriffen. Diese bietet robuste und effiziente Implementierungen der in Abschnitt 2.1 genannten geometrische Operationen [HJ03]. Einzige Ausnahme bildet das Voronoi-Diagramm, welches durch Erweiterung freier Quelltexte[10] von Paul Chew implementiert wurde.

Konkret bedeutet dies, von Knoten, Wegen und Verbünden zu abstrakten Datentypen für Punkte, Streckenzüge und Regionen zu kommen. Die Umwandlung von Knoten in Punkte und von Wegen in Streckenzüge ist trivial. Die Erstellung einer Region aus einem entsprechenden Verbund ist etwas komplizierter:

Der Verbund referenziert eine Menge von Wegen. Diese Wege definieren zusammengenommen eine Menge von geschlossenen Streckenzügen, welche sowohl innere als auch äußere Begrenzungslinien für die einzelnen Polygone der Region bilden. Folgendes ist dabei zu beachten[11]:

- Den Wegen als Mitglieder des Verbunds kann als Rolle einer der Werte "inner" oder "outer" gegeben werden, um zu beschreiben, ob es sich um die Außenlinie eines Polygons handelt oder um die innere Begrenzung eines Lochs im Polygon. Die Angabe solcher Rollen ist allerdings nicht zwingend erforderlich und findet sich nicht an der Mehrheit der untersuchten Regionen. Daher sollte Software, welche die Regionen weiterverarbeiten möchte, nicht von dieser Angabe abhängig sein.
- Ein geschlossener Streckenzug setzt sich nicht zwangsläufig aus genau einem Weg zusammen. Vielmehr lassen sich in der Regel eine ganze Menge von Wegen zu einem geschlossenen Streckenzug zusammensetzen. Die Reihenfolge, in der die Wege zusammenzusetzen sind, ist jedoch strukturell nicht definiert, sondern muss algorithmisch ermittelt werden, indem die verwendeten Knoten betrachtet werden.
- Ein Verbund kann prinzipiell wiederum Verbünde als Mitglieder enthalten. Die von referenzierten Verbünden referenzierten Wege müssen dann ebenfalls zur Konstruktion der Streckenzüge herangezogen werden. Über Rekursion sind im Prinzip auch noch tiefere Verschachtelung möglich. In der Praxis wurden jedoch höchstens Verschachtelungen der Tiefe zwei gefunden.

7.1.5 Extraktion der geographischen Objekte

Folgende Daten sind, wie in Abschnitt 6 erläutert, aus dem Gesamtdatenbestand auszuwählen: Straßensegmente, Verwaltungseinheiten, Postleitzahlregionen und Ortsmittelpunktsknoten. Dabei handelt es sich um mehrere *thematische Selektionen*. Darüber hinaus sollen die Daten auf die Zielregion beschränkt werden, was eine *geometrische Auswahl* darstellt.

Im Prinzip ließe sich diese Aufgabe zweistufig mit Hilfe der vorhandenen Verarbeitungsmethoden von Osmosis lösen: *Regionsfilter* und *Attributfilter* in beliebiger Reihenfolge. Der Regionsfilter müsste jedoch so konfiguriert werden, dass, wie in Abschnitt 7.1.2 beschrieben, dafür gesorgt wird, dass Entitäten vervollständigt werden. Grund hierfür ist, dass in der Zielregion gelegene Regionen nicht zwangsläufig vollständig von dieser eingeschlossen sind. Im einfachen Ausführungsmodus würden dann aber benötigte Knoten oder Wege nicht im Zieldatensatz landen.

Abbildung 30 veranschaulicht das Problem anhand der Zielregion A. Annahme ist hier, dass semantisch B und C *innerhalb* und D *außerhalb* von A liegen:

- Im einfachen Modus des Regionsfilters würde B vollständig extrahiert. Zwei Knoten von C hingegen würden nicht in den Zieldatensatz aufgenommen und die Region könnte im Weiteren nicht mehr korrekt verarbeitet werden.
- Wird hingegen der *vervollständigende* Modus des Regionsfilters angewendet, werden sowohl C als auch D vollständig extrahiert.

[10] http://www.cs.cornell.edu/Info/People/chew/Delaunay.html
[11] http://wiki.openstreetmap.org/wiki/Relation:multipolygon

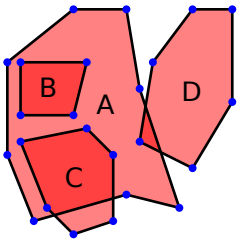


Abbildung 30: Eine Zielregion A und mehrere Eingaberegionen

Ein ähnliches Problem besteht für Wege, die an der Grenze der Zielregion liegen.

Um die Korrektheit der Extraktion zu wahren, müsste also der ineffiziente *vervollständigende* Modus des Regionsfilters ausgeführt werden. Hiermit werden jedoch inakzeptable Laufzeiten erreicht. Daher wurde eine alternative Methode gesucht:

- Einfacher Regionsfilter mit gepufferter Zielregion:
  - Es wird die Puffer-Operation auf die Zielregion angewendet.
  - Die resultierende erweiterte Region wird dann mit Hilfe des Regionsfilters im einfachen Modus aus den Eingabedaten ausgeschnitten.

  Wird der Puffer groß genug gewählt, ist davon auszugehen, dass alle, in der ursprünglichen Zielregion gelegenen Regionen (und ein paar mehr) extrahiert werden. Konkret wurde ein Puffer von 0,4 Grad verwendet. Abbildung 31 zeigt die resultierende Region.
- Nun liegt eine Datei vor, welche die gepufferte Zielregion, also die Zielregion plus einen kleinen zusätzlichen Bereich darum, enthält. Diese Datei wird jetzt in drei Dateien aufgespalten, sodass jede der Dateien nur Objekte je eines der strukturellen Entitätstypen enthält. Dadurch lassen sich spätere Schritte effizienter ausführen.
- Jetzt werden die thematischen Selektionen mittels Attributfilter ausgeführt:
  - Ortsmittelpunktsknoten werden aus den Knoten ausgewählt.
  - Straßensegmente werden aus den Wegen ausgewählt.
  - Verwaltungseinheiten werden aus den Verbünden ausgewählt.
  - Postleitzahlregionen werden aus den Verbünden ausgewählt.
- für Straßensegmente bzw. Verwaltungseinheiten und Postleitzahlregionen müssen noch die referenzierten Knoten bzw. Wege und Knoten aus den entsprechenden Dateien extrahiert werden. Dies ist jetzt vergleichsweise effizient mit einem Iterator über Knoten und Wege möglich.
- Momentan liegen alle ausgewählten Objekte noch für die gepufferte Region vor. Sie müssen also noch auf die tatsächliche Zielregion beschnitten werden. Die Verlagerung dieses Schritt ans Ende der Extraktionskette macht diesen Schritt deutlich effizienter:
  - Die Ortsmittelpunktsknoten können mit einem *einfachen* Regionsfilter ausgeschnitten werden.
  - Für die Straßensegment wird ein *vervollständigender* Regionsfilter eingesetzt. Da dieser auf vergleichsweise wenig Daten arbeitet, fällt die ineffiziente Implementierung nicht mehr besonders ins Gewicht.
  - Verwaltungseinheiten und Postleitzahlregionen werden mit einem speziellen Regionsfilter ausgeschnitten, der in Abschnitt 7.1.6 erläutert wird.

7.1.6 Spezieller Regionsfilter

Der spezielle Regionsfilter ist als Osmosis-Plugin realisiert, der eine räumliche Selektion von Regionen ausführt, die zu einem bestimmten Grad von einer gegebenen Zielregion Z überdeckt sind.

Eine Region A wird genau dann von dem Plugin weitergegeben, wenn die Schnittfläche $A \cap Z$ einen flächenmäßigen Anteil an A hat, der einen Schwellwert d überschreitet. In der Praxis wurden mit einem Wert für d von 0,9 zufriedenstellende Ergebnisse erzielt.

Motivation für den speziellen Regionsfilter sind Situationen wie die in Abbildung 30 gezeigte. In diesem Fall würden mit diesem speziellen Filter B und C vollständig extrahiert. Die Region D würde hingegen nicht, aber auch nicht teilweise extrahiert.

Relevant ist dies beispielsweise bei der Extraktion der Gemeinden in der Testregion Deutschland. Einerseits sollen alle Gemeinden aus Deutschland extrahiert werden, selbst wenn einige Knoten der Gemeinderegion außerhalb der Staatsgrenze Deutschlands modelliert wurden. Andererseits soll vermieden werden, dass Gemeinden in den Datensatz aufgenommen werden, die eigentlich im Ausland liegen. Hier liegt der Unterschied zum *vervollständigenden* Regionsfilter denn dieser extrahiert solche Regionen aufgrund einzelner, im Inland modellierter Knoten.

7.1.7 Transformation in die Entitäten des Referenzdatensatzes

Die Transformation der thematischen Entitäten der Openstreetmap in die Entitäten, die im Referenzdatensatz gebraucht werden, gliedert sich in folgende Teilschritte:

1. Straßensegmente → Straßen. Innerhalb einer Gemeinde werden die Straßensegmente wie in Abschnitt 6.1 beschrieben zu logischen Einheiten zusammengefasst, wobei als Kriterien der Zusammenfassung Name und Distanz eingesetzt werden.

   Im Folgenden wird der Algorithmus vorgestellt, der zur Gruppierung der Straßensegmente zu Straßen im postalischen Sinne verwendet wird.

   Die Segmente einer Gemeinde werden zunächst über den Namen zusammengefasst und dann über die Entfernung in Gruppen von Segmenten aufgeteilt. Dazu wird der Algorithmus, der in Listing 1 schematisch dargestellt ist, verwendet.

```
for (name : names) {
    segments = getSegmentsWithName(name)
    Graph graph = new Graph();
    for (segment : segments){
        graph.addNode(segment);
    }
    for (a : segments) {
        for (b : segments without a){
            distance = getDistance(a, b);
            if (distance < threshold){
                graph.addEdge(a,b);
            }
        }
    }
    Set<Set<Segment>> groupsOfSegments = graph.getPartition()
    Set<Street> streets = new StreetSet(groupsOfSegments)
}
```

Listing 1: Gruppierung von Straßen nach ihrer Entfernung

   Für jeden vorhandenen Namen werden zunächst alle vorhandenen Segmente ermittelt. Zwei Segmente sollen dann zusammengefasst werden, wenn die Distanz zwischen den beiden Segmenten das Limit *threshold* unterschreitet. Außerdem soll die Zusammenfassung transitiv erfolgen, das heißt wenn a und b zusammengefasst werden und b und c zusammengefasst werden, dann sollen auch a

und c zusammengefasst werden. Dieses Kriterium wird über einen einfachen Graphenalgorithmus umgesetzt: Alle Segmente werden als Knoten in einen Graphen eingefügt. Für je zwei Segmente a, b wird ermittelt, ob diese aufgrund ihrer Distanz zusammengefasst werden sollen. Ist dies der Fall, wird eine Kante zwischen a und b eingefügt. Ist der Algorithmus fertig, so stellt die Partitionierung des Graphen die gewünschte Zerlegung in Mengen von Segmenten dar.

2. Verwaltungseinheiten → Gemeinden. Hier sind einerseits thematische Selektionen notwendig und andererseits kommt die in Abschnitt 6.3.3 entwickelte Heuristik zum Einsatz.

   Zunächst werden attributbasiert die administrativen Einheiten der Ebene 8 als Gemeinden und die Einheiten der Ebenen 4 und 6 als Kandidaten für die Heuristik ausgewählt. Durch deren Einsatz werden die Regionen aus den Ebenen 4 und 6 ausgewählt, die höchstens einen Stadt- oder Großstadtknoten enthalten.

   Hierzu wird pro Region ein mit dem Wert 0 initialisierter Zähler geführt. Dann wird für jeden Stadtknoten ermittelt, in welchen Regionen dieser liegt und die entsprechenden Zähler werden inkrementiert. Anschließend werden alle Regionen als Gemeinde ausgewählt, deren Zähler einen Wert kleiner oder gleich 1 hat.

   Zur effizienten Implementierung des Algorithmus wird ein R-Baum eingesetzt, in den die Regionen eingefügt werden. Damit ist es effizient möglich, zu einem Stadtknoten die enthaltenden Regionen zu ermitteln. Obwohl auch *JTS* einen R-Baum zur Verfügung stellt, wurde aus Effizienzgründen die speziellere Bibliothek *Java Spatial Index (JSI)*[12] eingesetzt, da mit dieser deutlich schnellere Ausführungszeiten erreicht werden konnten.

3. Ermittlung des nicht-abgedeckten Bereichs: Aus den in Schritt 2 ermittelten Gemeinden wird zunächst der von Gemeinden abgedeckte Bereich durch Vereinigung sämtlicher Gemeinderegionen ermittelt. Die Differenz dieser Vereinigung von der Eingaberegion bildet dann die Region, für die mit Hilfe der Ortsmittelpunktsknoten durch die Bildung von Voronoi-Zellen die restlichen Gemeinderegionen ermittelt werden.

   Die Vereinigung hochaufgelöster Gemeinderegionen stellt eine relativ kostspielige Operation dar. In Deutschland werden derzeit gut 11 000 solcher Gemeinderegionen vereinigt. Bei solchen Eingabegrößen fällt diese Operation gemessen an der Gesamtlaufzeit des Vorverarbeitungsprozesses stark ins Gewicht.

   Zunächst wurde ein naiver Algorithmus implementiert, der die Eingaberegionen sequenziell vereinigt. Dabei wächst die Komplexität in der Regel mit jeder Iteration, sodass immer mehr Zeit pro Vereinigung benötigt wird.

   Deshalb wurde ein weiterer Algorithmus implementiert, der sich die Dichte der gegebenen Regionen zu Nutze macht. Damit ist gemeint, dass in der Regel viele der Eingaberegionen eine gemeinsame Grenze haben. Der Algorithmus bekommt als Eingabe eine Menge von Regionen R und liefert die Vereinigung all dieser Regionen:

   (a) Wähle eine beliebige Region aus R. Entferne diese aus R und nenne diese Region c.

   (b) Wähle und entferne aus R alle Nachbarregionen S, die sich mit c schneiden.

   (c) Bilde sequenziell die Vereinigung d von c mit allen Elementen aus S.

   (d) Setze $c = d$ und fahre mit (b) fort.

   Wenn sich in Schritt (b) keine Nachbarn von c finden lassen, wird c in eine Menge von Teilergebnissen aufgenommen und der Algorithmus beginnt von vorn. Ist R leer, werden alle bisherigen Teilergebnisse vereinigt. Diese Vereinigung liefert dann das Ergebnis und der Algorithmus terminiert.

   Die Effizienzsteigerung gegenüber dem naiven Algorithmus wird dadurch erreicht, dass die Komplexität (im Sinne der Anzahl von Ecken und Kanten) der Zwischenergebnisse gesenkt wird, weil durch die Vereinigung von benachbarten Regionen Ecken und Kanten wegfallen. Die Ausführungszeit für die Operation konnte mit diesem Algorithmus von 4 Stunden und 16 Minuten auf nur 27 Minuten reduziert werden.

[12] http://jsi.sourceforge.net/

4. Ortsmittelpunkte → Gemeinden. Zunächst werden alle Ortsmittelpunkte ausgewählt, die innerhalb der bisher nicht abgedeckten Region R liegen, die in Schritt 3 berechnet wurde. Dazu wird lediglich der Überdeckungstest von JTS benötigt. Anschließend wird für diese Ortsmittelpunkte ein Voronoi-Diagramm gebildet. Damit die entstehenden Regionen im weiteren Vorgehen nicht von den in Schritt 2 gebildeten Gemeinderegionen unterschieden werden müssen, wird jede Region des Voronoi-Diagramms mit R geschnitten.
5. Bilde Gesamtheit der Gemeinden. Die Menge der Gemeinden ergibt sich aus der Vereinigung der Gemeinden aus den Schritten 2 und 4.
6. Postleitzahlregionen. Die Postleitzahlregionen stehen durch eine thematische Selektion zur Verfügung. In der Praxis konnten für eine feste Postleitzahl oft zwei Regionen gefunden werden. In solchen Fällen wurden diese zur Vereinfachung der Daten zu einer Gesamtregion vereinigt. Diese Vereinfachung lässt sich in den Abbildungen erkennen: Abbildung 32 zeigt alle extrahierten Postleitzahlregionen und Abbildung 33 zeigt die Regionen, die sich ergeben, wenn Regionen, die die gleiche Postleitzahl modellieren vereinigt werden.

7.1.8 Herstellung der Beziehungen

Zur Herstellung der Beziehungen der Zielobjekte müssen primär die Beziehungen zwischen den Straßen und den anderen Objekte hergestellt werden. Darüber hinaus werden noch die Beziehungen zwischen Gemeinden und Stadtteilen hergestellt.

Folgende Aufgaben sind zur Herstellung der Beziehungen nötig:

- Straßenbeziehungen
  - Finde alle beinhaltenden Gemeinden.
  - Finde alle beinhaltenden Stadtteile.
  - Finde alle beinhaltenden Postleitzahlregionen.
- Gemeinde - Stadtteil
  - Finde für jeden Stadtteil die beinhaltende Gemeinde

Technisch muss hier jeweils das Prädikat der Überdeckung für jedes Paar von Straße / Region evaluiert werden. Implementiert wurde die Evaluation dieser Prädikate mit JTS unter Einsatz eines R-Baums zur Effizienzsteigerung.

Für die Beziehung Gemeinde - Stadtteil muss ebenfalls das Prädikat der Überdeckung evaluiert werden. Dazu wird analog zum speziellen Regionsfilter aus Abschnitt 7.1.6 die Überdeckung mit einem Schwellwert von 0,9 ermittelt.

7.1.9 Relationales Datenmodell

Die in den vorherigen Abschnitten ermittelten Zielobjekte und deren Beziehungen werden in eine Datenbank eingefügt.

Dazu wurden entsprechende Java-Klassen modelliert, die dann unter Einsatz von *Annotationen* zur Verwendung mit dem Objekt-relationalen Werkzeug *Hibernate* konfiguriert wurden. Das entsprechende relationale Modell wurde dann automatisiert aus diesen Klassen erzeugt.

Dieser Schritt wurde unternommen, um einerseits das Datenbankschema in Zukunft leicht anpassen zu können, andererseits aber auch, um ohne viel Aufwand verschiedene Zielplattformen unterstützen zu können:

- für die Android-Bibliothek wurde eine dateibasierte *SQLite*-Datenbank befüllt, da diese auf dem Betriebssystem nativ unterstützt wird.
- eine Portierung auf PostgreSQL eröffnet zum Beispiel die Implementierung von performanten Webservices.

7.2 Zielplattform

Zielplattform ist das *Android*-Betriebssystem. Abschnitt 7.2.1 stellt die entwickelte API zum Zugriff auf die Referenzdaten aus der dateibasierten SQLite-Datenbank vor. Abschnitt 7.2.2 geht dann darauf ein, wie darauf basierend eine rudimentäre Benutzerschnittstelle zur Integration in die Applikation *AdvancedMapViewer* erstellt wurde. In Abschnitt 7.2.3 wird dann auf die Kompatibilität zur Geokodierungsschnittstelle von *Android* eingegangen.

7.2.1 API zur Datenbank

Generell könnte auch von Android aus über das *Hibernate-Mapping* auf die vorhandene Datenbank zugegriffen werden. Dann würden jedoch *JDBC* und *Hibernate* als Middleware zum Einsatz kommen, was auf dem mobilen Gerät aufgrund des entstehenden Overheads nicht unbedingt wünschenswert ist.

SQLite ist jedoch eine fest in Android integrierte Datenbank, weshalb eigene Zugriffsmethoden auf diese Art von Datenbank angeboten werden. Durch die Verwendung dieser Mechanismen ist JDBC gar nicht zum Datenbankzugriff nötig, wie es sonst in der Java-Welt üblich ist. Vermutlich versprechen die Entwickler sich hiervon verbesserte Zugriffszeiten.

Da es auf der Plattform so üblich ist, wurde der Verwendung der internen Zugriffsmethoden der Vorzug gegeben. Dies bedeutet bezüglich des Entwicklungskomforts zunächst einmal einen Rückschritt von objektorientierter zu relationaler Sicht auf die Objekte in der Datenbank.

Um dennoch objektorientiert auf der Zielplattform mit den Daten hantieren zu können, wurden entsprechende Adapterklassen entworfen, welche die Interaktion mit der Datenbank vor dem Nutzer verbergen.

Folgende Methoden bieten Einstiegspunkte, um Referenzen auf Objekte zu erhalten:

```
public static List<SqCity> getCities(SQLiteDatabase db, String needle, int limit);

public static List<SqPostcode> getPostcodes(SQLiteDatabase db, String needle, int limit);
```

Listing 2: Zugriff auf Gemeinden oder Postleitzahlen

Ausgehend von einer Stadt oder Postleitzahl steht dann die folgende Methode zur Verfügung, um Straßen zu ermitteln:

```
public List<SqRoad> getRoads(String needle, int limit);
```

Listing 3: Zugriff auf Straßen von Gemeinden oder Postleitzahlen aus

Über den Parameter *needle* wird bestimmt, welche Objekte gefunden werden. Die erhaltene Zeichenkette wird von der API intern zum Abgleich der Namensspalte der jeweiligen Relation verwendet. Dabei kommt der *LIKE*-Operator zum Einsatz, sodass der Nutzer der API über Verwendung der SQL-Wildcard "%" die Ergebnismenge beeinflussen kann. Darüber hinaus lässt sich über den Parameter *limit* bestimmen, wie viele Ergebnisse höchstens geliefert werden sollen, um eine Beschränkung der Ressourcen zu erlauben.

7.2.2 Benutzerschnittstelle

Auf Grundlage der API wurde ein grafische Benutzerschnittstelle implementiert. Diese erlaubt in Anlehnung an die Nutzungsszenarien aus Abschnitt 5.3 das Auffinden einer Gemeinde oder einer Straße.

Dazu wird der Nutzer zunächst zur Eingabe einer Stadt oder einer Postleitzahl in einem einfachen Textfeld gebeten. Dabei wird er über ein *AutoCompleteTextView* bei der Eingabe unterstützt, indem schon bei der Eingabe eines einzelnen Buchstabens die 20 ersten Treffer einer API-Anfrage zur direkten Auswahl gestellt werden. Umgesetzt wurde dies, indem die vom AutoCompleteTextView geforderte Schnittstelle *ListAdapter* durch Erweiterung der Klasse *BaseAdapter* auf Basis der API implementiert wurde.

Hat der Nutzer Gemeinde oder Postleitzahl über Auswahl eines Objekts aus der Vervollständigungsliste gewählt, kann im nächsten Textfeld der Name einer Straße eingegeben werden. Auch hier wird analog zur vorherigen Suche über eine automatische Vervollständigung Unterstützung gegeben, sodass das Textfeld eventuell schon nach der Eingabe weniger Zeichen den gewünschten Suchbegriff enthält.

In jedem Fall ist momentan das Bedienen des *Suchen*-Knopfs notwendig. Daraufhin wird dem Nutzer nochmals eine Liste der Suchergebnisse präsentiert. In dieser Ansicht werden dann auch detailliertere Informationen zur jeweiligen Straße dargestellt, sodass im Zweifelsfall die gewünschte Auswahl getroffen werden kann. Dazu werden, in Klammern hinter dem Namen der Straße, sowohl Stadtteile als auch Postleitzahlen angegeben.

Angenommen der Nutzer wählt zunächst als Gemeinde *Berlin* und sucht dann nach der *Schillerstraße*, so wird ihm die folgende Liste zur Auswahl präsentiert:

- Schillerstraße (Lichterfelde, Steglitz-Zehlendorf) (12207)
- Schillerstraße (Buckow 1, Neukölln) (12529, 12353)
- Schillerstraße (Tempelhof-Schöneberg, Lichtenrade) (12305)
- Schillerstraße (Niederschönhausen, Rosenthal, Pankow) (13158, 13156)
- Schillerstraße (Zehlendorf, Steglitz-Zehlendorf) (14163, 14129)
- Schillerstraße (Charlottenburg-Wilmersdorf, Charlottenburg) (10627, 10625, 10623)
- Schillerstraße (Wilhelmsruh, Pankow) (13158)
- Schillerstraße (Bohnsdorf, Treptow-Köpenick) (12526)

7.2.3 Kompatibilität zur Geokodierungsschnittstelle von Android

Android bietet dem Nutzer eine Schnittstelle zur Geokodierung und umgekehrten Geokodierung an: Zentrale Klasse ist hier der *Geocoder*[13], welcher die folgenden zentralen Methoden zum Erhalt von *Address*-Objekten[14] bereitstellt:

```
List<Address> getFromLocation(double latitude, double longitude, int maxResults);

List<Address> getFromLocationName(String locationName, int maxResults);
```

Listing 4: Zugriff auf Objekte des Typs *Addresss*

Über diese Methoden kann einerseits über eine Koordinate und andererseits über eine beliebig formatierte Zeichenkette eine Liste von möglichen Adressen bezogen werden. Die Verwendung dieser Klasse benötigt

[13] http://developer.android.com/reference/android/location/Geocoder.html
[14] http://developer.android.com/reference/android/location/Address.html

jedoch ein *Backend*, um Ergebnisse zu liefern. Aus der Dokumentation ist zunächst jedoch nicht zu ersehen, wie dieses Backend konfiguriert oder ausgetauscht werden kann.

Sowohl *Geocoder* als auch *Address* sind Klassen, nicht Schnittstellen, weshalb eine austauschbare Implementierung ohne Weiteres nicht möglich ist. Über das Backend ließe sich diese Problematik aber eventuell lösen.

Insbesondere ist aber auch die Implementierung eines intelligenteren Verarbeitungsalgorithmus nötig, um die Anforderungen dieser Klassen erfüllen zu können: Die Umwandlung einer beliebigen Zeichenkette, als Eingabe, in eine geographische Referenz, als Ausgabe, ist mit Hilfe typischer Konzepte der Geokodierung zu lösen, die allerdings nicht Teil dieser Arbeit sind.

8 Schluss

8.1 Zusammenfassung

In dieser Arbeit wird ein Geokodierer für Adressen für das Mobilgerätebetriebssystem *Android* entwickelt, der auf den Daten der *Openstreetmap* basiert. Zu diesem Zweck wird ein geokodierter Referenzdatensatz für Adressen aus einem Abbild der Datenbank der Openstreetmap erzeugt. Dessen Entwicklung steht als zentrale Komponente des Geokodierers im Mittelpunkt dieser Arbeit.

In der vorliegenden Form kann der Referenzdatensatz dazu verwendet werden, strukturierte Suchanfragen nach gängigen Suchmustern zu beantworten. Konkret lässt sich, durch Angabe einer Adresse in Form von Gemeinde oder Postleitzahl in Kombination mit einem Straßennamen, eine Koordinate ermitteln, die auf der gesuchten Straße liegt.

Zu diesem Zweck wurde eine einfache API auf der Zielplattform implementiert, die bei der Anwendungsentwicklung verwendet werden kann. Darauf basierend konnte eine benutzerfreundliche grafische Schnittstelle für den *AdvancedMapViewer* erstellt werden, die den Nutzer durch automatische Vervollständigung seiner Eingaben unterstützt. Das Format der Daten im Referenzdatensatz wurde dabei für den Einsatz auf dem mobilen Gerät optimiert, sodass schnelle Zugriffszeiten und geringer Speicherverbrauch erreicht werden konnten.

Adressen werden in dieser Arbeit in erster Linie durch Straßen, Gemeinden und Postleitzahlen modelliert. Für diese Komponenten einer Adresse konnten im untersuchten Gebiet ausreichende Datenmengen gefunden werden. Zusätzlich werden, wo vorhanden, auch Stadtteile zur Modellierung herangezogen. Diese können verwendet werden, um mit der Situation umzugehen, dass Straßennamen in manchen Gemeinden mehrfach vergeben worden sind. Vernachlässigt werden hingegen Hausnummern, weil davon auszugehen ist, dass höchstens 5% aller tatsächlich vorhandenen Hausnummern aus den Ausgangsdaten hergeleitet werden könnten.

Als Testregion wurde die Bundesrepublik Deutschland gewählt. Es wurde gezeigt, dass sowohl für Gemeinden, als auch für Postleitzahlen für mehr als 90% der untersuchten Fläche hochaufgelöst erfasste Regionen in der Datenbank der Openstreetmap vorhanden sind. Diese stellen, in Kombination mit den Straßen, die wichtigsten Komponenten dar, um Adressen zu bilden.

Überall dort, wo für Gemeinden keine genauen Grenzen vorliegen, wurden die weniger genauen Ortsmittelpunktsknoten eingesetzt, um auch aus den verbleibenden Straßen noch möglichst sinnvolle Adressen zu bilden. Obwohl damit zu rechnen ist, dass dadurch auch fehlerhafte Adressen entstehen, wird angenommen, dass über dieser Vereinfachung deutlich mehr richtige als falsche Adressen in die Datenbank aufgenommen werden. Mögliche Fehler werden ignoriert, da davon ausgegangen wird, dass langfristig flächendeckend hochaufgelöste Daten verfügbar sein werden.

Durch die räumliche Beschränkung auf die Bundesrepublik war es möglich, die ermittelten Daten teilweise manuell zu überprüfen, sodass eine bestimmte Gewissheit über die Qualität hergestellt werden konnte.

Betrachtet wurden hierzu die 200 größten Städte Deutschlands. Hier konnte in 94% der Fälle eine Gemeinderegion ermittelt werden, welche das der Stadt zugehörige Gebiet genau festlegt. Setzt man eine vollständige Erfassung des Straßennetzwerks voraus, kann somit davon ausgegangen werden, dass zumindest für diese wichtigen Teilgebiete umfassende Adressinformationen ermittelt werden können. Durch Hinzunahme der Ortsmittelpunktsknoten konnte auch für die restlichen Städte das zugehörige Gebiet, zumindest näherungsweise, bestimmt werden.

Auch für die Postleitzahlen konnte die Abdeckung genauer überprüft werden: 91% aller von der Post vergebenen Postleitzahlen konnten in Form von geographischen Regionen ermittelt werden.

Der Referenzdatensatz wird in einem Vorverarbeitungsprozess hergeleitet, welcher im Rahmen dieser

Arbeit in Java implementiert wurde. Dieser wird mit einer zu bearbeitenden Zielregion parametrisiert und erzeugt aus einem vollständigen Datenbankabbild der Openstreetmap eine SQLite-Datenbank. Für die Zielregion Deutschland läuft dieser Prozess auf dem Testsystem derzeit in 4,5 Stunden ab.

Im Rahmen der Implementierung wurden viele technische und konzeptionelle Probleme in Umgang und Weiterverarbeitung der Rohdaten der Openstreetmap gelöst. Dazu zählt vor allem die Überführung sämtlicher struktureller Entitäten in abstrakte geometrische Datenstrukturen der Java-Bibliothek *JTS*. Ebenfalls wichtig sind die entstandenen Plugins für Osmosis, sowie die Adapter, die eine imperative Verwendung der Osmosis-Bibliothek möglich machen.

8.2 Ausblick

Diese Arbeit löst in erster Linie ein recht allgemeines Problem: Die Extraktion eines Referenzdatensatzes zur Geokodierung aus der Openstreetmap. Der Einsatz dieses Datensatzes auf mobilen Geräten stellt einen, aber nicht den einzigen Anwendungsfall dar. Im Gegenteil gibt es eine Reihe anderer Anwendungen, für die diese Daten von Interesse sind. Ein naheliegender Anwendungsfall wäre der eines Webservices zur Routenplanung oder ganz allgemein zur Adresssuche.

Die kleinste adressierbare Einheit ist derzeit die Straße. Sollte der Abdeckungsgrad der Hausnummern in den Ausgangsdaten steigen, kann das Modell des Referenzdatensatzes entsprechend erweitert werden. Offen bleibt jedoch, wie dies tatsächlich durchgeführt werden könnte.

Zur Ermittlung der Gemeinden aus den Verwaltungsgrenzen kommt derzeit eine Heuristik zum Einsatz. Generell wäre es wünschenswert, dass diese Informationen nicht erst abgeleitet werden müssen, sondern von den Nutzern in die Hauptdatenbank aufgenommen, und damit direkt zugreifbar gemacht wird. Dazu wäre es jedoch zunächst nötig, geeignete Attribute zu definieren, um diese Kategorisierung zu ermöglichen.

Prinzipiell müsste der Referenzdatensatz sich auch relativ gut eignen, um *Reverse Geocoding*, d.h. die Zuordnung einer Adresse zu einer Georeferenz umzusetzen. Hierzu könnten beispielsweise alle Straßen in einem geometrischen Index abgelegt werden, sodass die nächstgelegene Straße und damit auch Gemeinde und Postleitzahl ermittelt werden können.

Eine Implementierung der Geokodierungsschnittstelle von Android bleibt offen. Unter Einsatz bekannter Methoden der Geokodierung sollte diese Komponente auf Grundlage des Referenzdatensatzes implementiert werden.

Die Bereitstellung aller geometrischen Bestandteile in abstrakten Datentypen ermöglicht prinzipiell den Aufbau einer geographischen Datenbank, beispielsweise auf Basis von PostgreSQL und PostGIS, in der alle geometrischen Merkmale der Openstreetmap auswertbar sind. Die Überführung ließe sich dabei mittels *Well-Known-Binary* oder *Well-Known-Text*-Repräsentationen der geometrischen Komponenten realisieren, die sowohl von *JTS* als auch von *PostGIS* geschrieben bzw. gelesen werden können.

Literatur

[Ame09] Christof Amelunxen. An approach to geocoding based on volunteered spatial data. Master's thesis, Paris Lodron-Universität Salzburg, 06 2009.

[Aur91] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[DFB03] Clodoveu A. Davis, Frederico T. Fonseca, and Karla A. V. Borges. A flexible addressing system for approximate geocoding. In *GeoInfo*, 2003.

[GWK07] D.W. Goldberg, J.P. Wilson, and C.A. Knoblock. From text to geographic coordinates: the current state of geocoding. *URISA Journal*, 19(1):3, 2007.

[HJ03] L. Harrie and M. Johansson. Real-time data generalisation and integration using Java. *Geoforum Perspektiv*, pages 29–34, 2003.

[Nei08] Pascal Neis. Location based services mit openstreetmap daten. Master's thesis, Fachhochschule Mainz, 08 2008.

[Peu76] T.K. Peucker. A theory of the cartographic line. *International yearbook of cartography*, 16:134–143, 1976.

[RSV02] Philippe Rigaux, Michel Scholl, and Agnès Voisard. *Spatial Databases with application to GIS.* Morgan Kaufmann, San Francisco, CA, 2002.

[RT08] Frederik Ramm and Jochen Topf. *OpenStreetMap. Die freie Weltkarte nutzen und mitgestalten.* Lehmanns Media, Berlin, 2008.

[Vet10] Christian Vetter. Fast and exact mobile navigation with openstreetmap data. Diploma thesis, Karlsruhe Institute of Technology, 03 2010.

A Abbildungen für Deutschland

- Zielregion und gepufferte Zielregion: Abbildung 31
- Postleitzahlregionen: Abbildung 32
- Postleitzahlregionen, gleichnamige vereinigt: Abbildung 33
- Postleitzahlregionen, insgesamt: Abbildung 34
- Administrative Einheiten
  - Ebene 6: Abbildung 35
  - Ebene 7: Abbildung 36
  - Ebene 8: Abbildung 37
  - Ebene 9: Abbildung 38
  - Ebene 10: Abbildung 39
  - Ebene 11: Abbildung 40
- Gemeindegrenzen, insgesamt: Abbildung 41
- Über Gemeindegrenzen nicht abgedeckte Region: Abbildung 42
- Orstmittelpunktsknoten: Abbildung 43

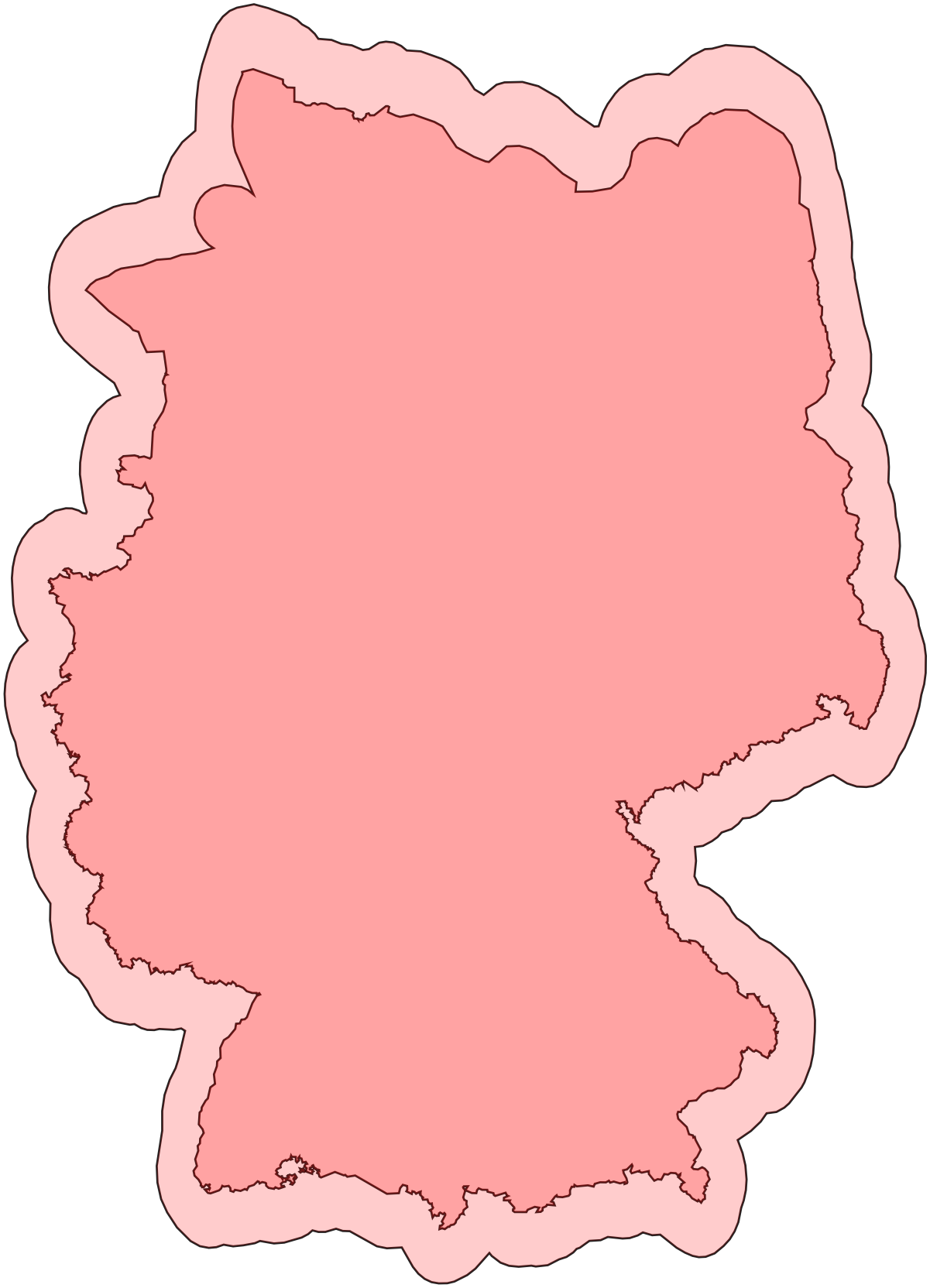


Abbildung 31: Die Staatsgrenze der Bundesrepublik Deutschland und eine um 0.4 Grad erweiterte Region

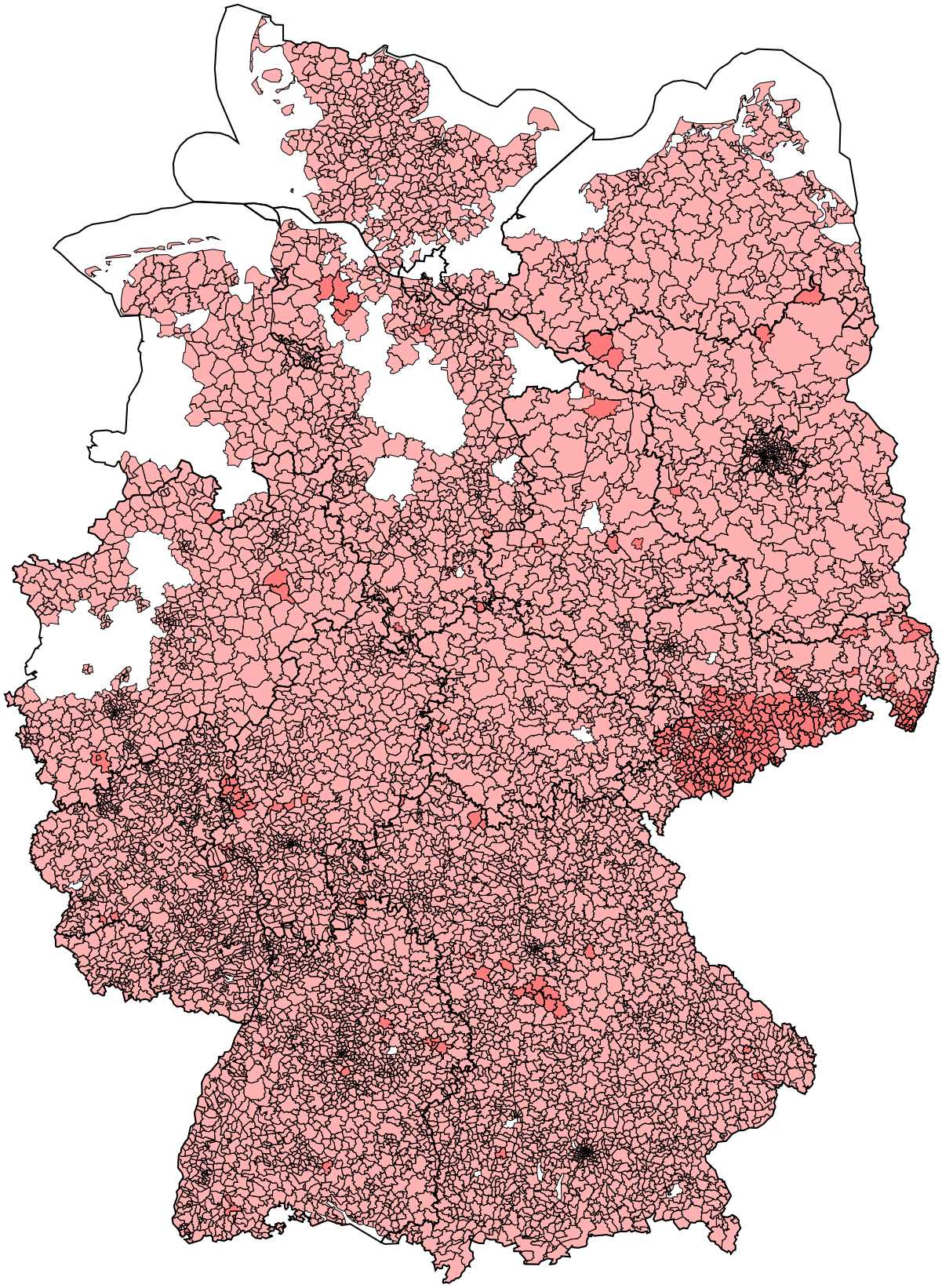


Abbildung 32: Postleitzahlregionen

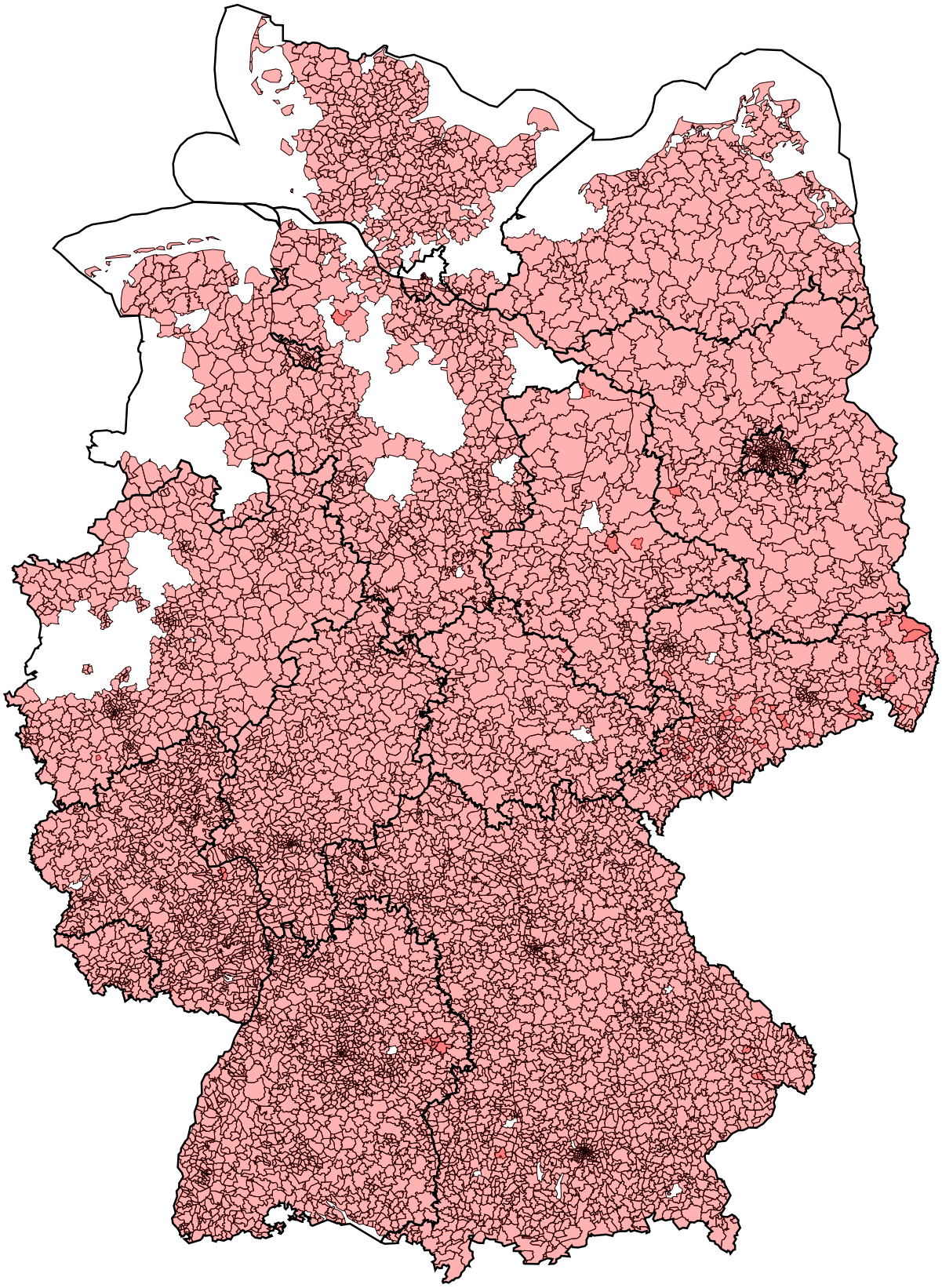


Abbildung 33: Postleitzahlregionen, gleichnamige vereinigt

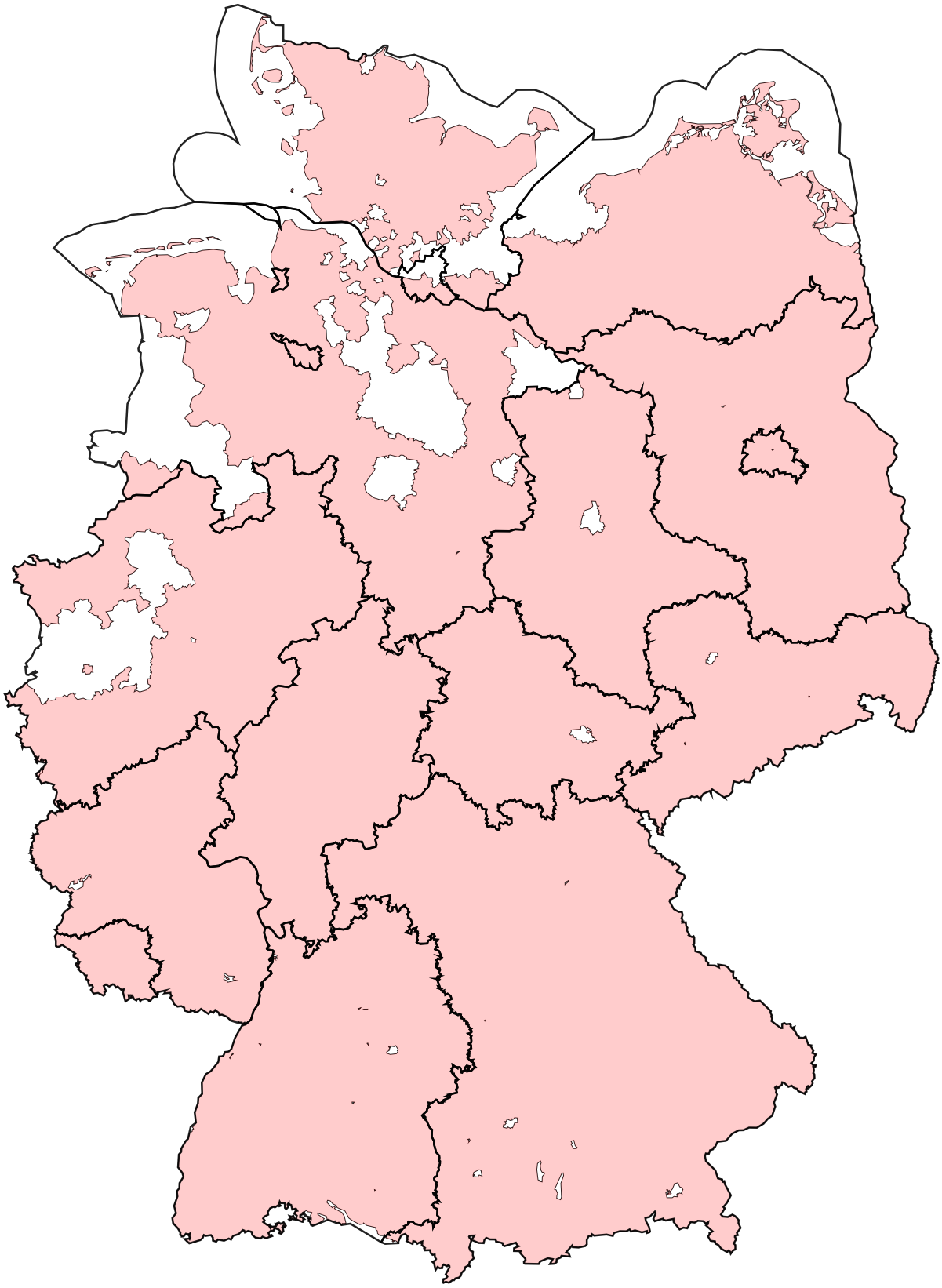


Abbildung 34: Über Postleitzahlregionen erfasstes Gebiet

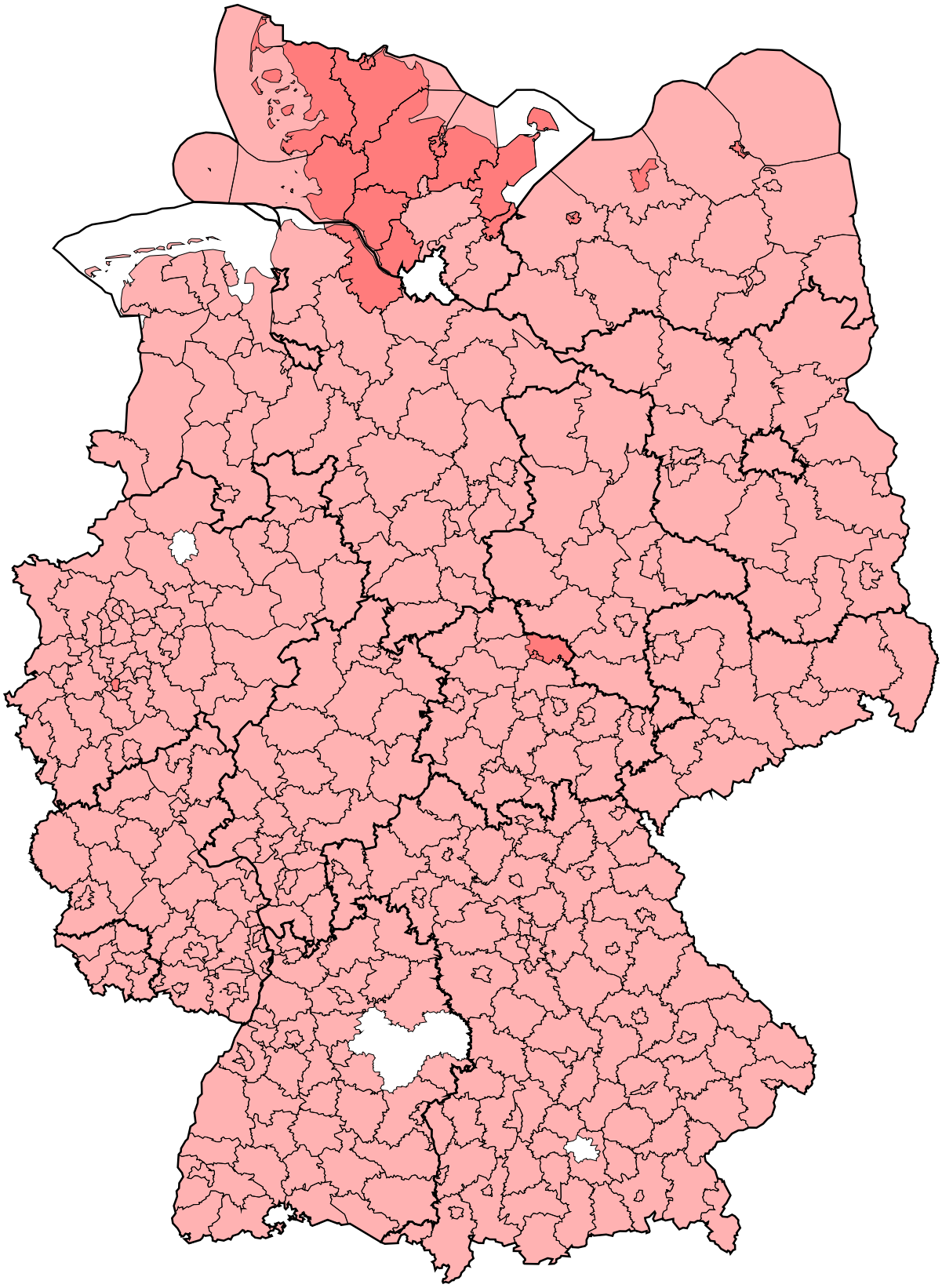


Abbildung 35: Administrative Einheiten auf Ebene 6

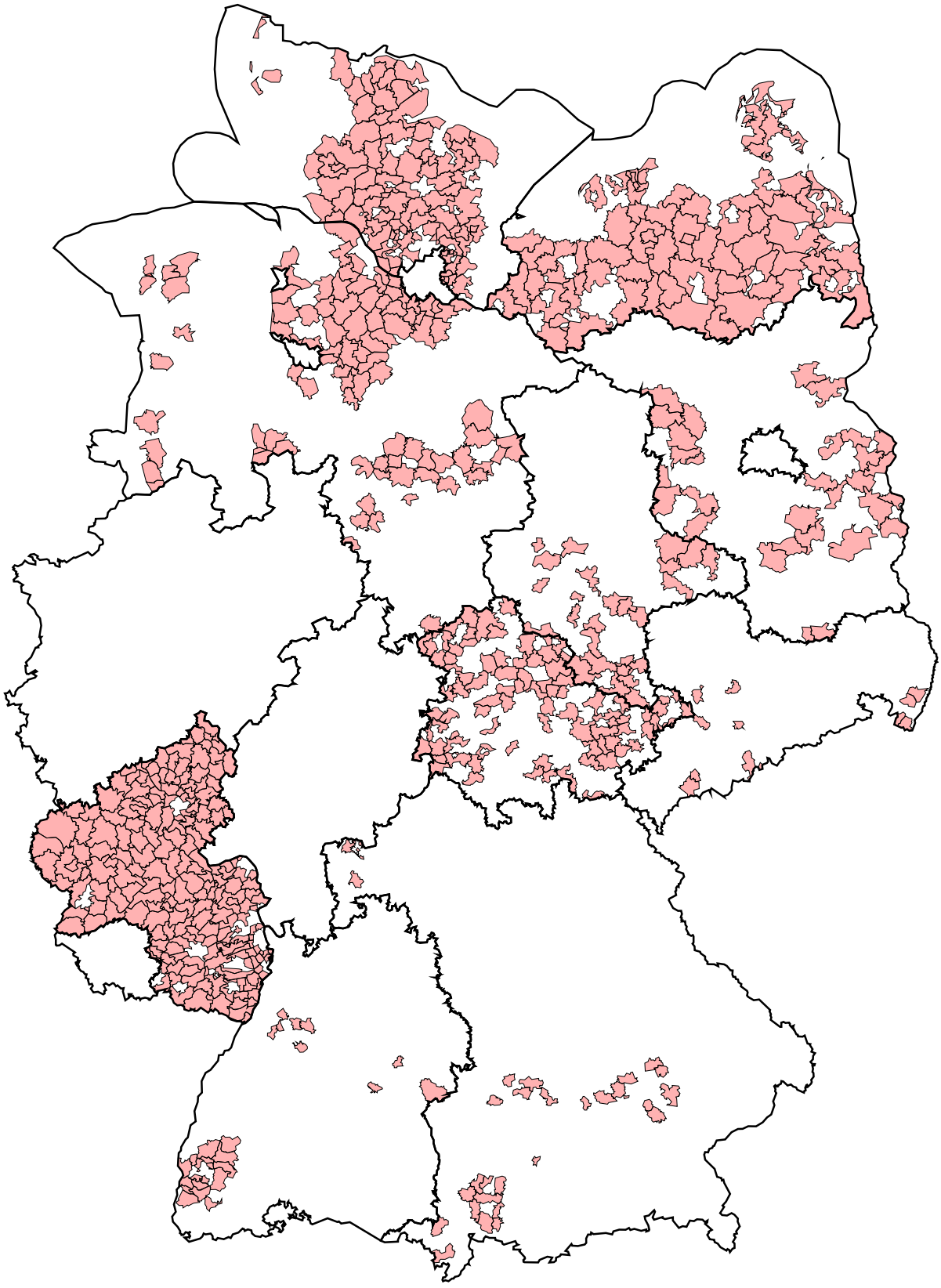


Abbildung 36: Administrative Einheiten auf Ebene 7

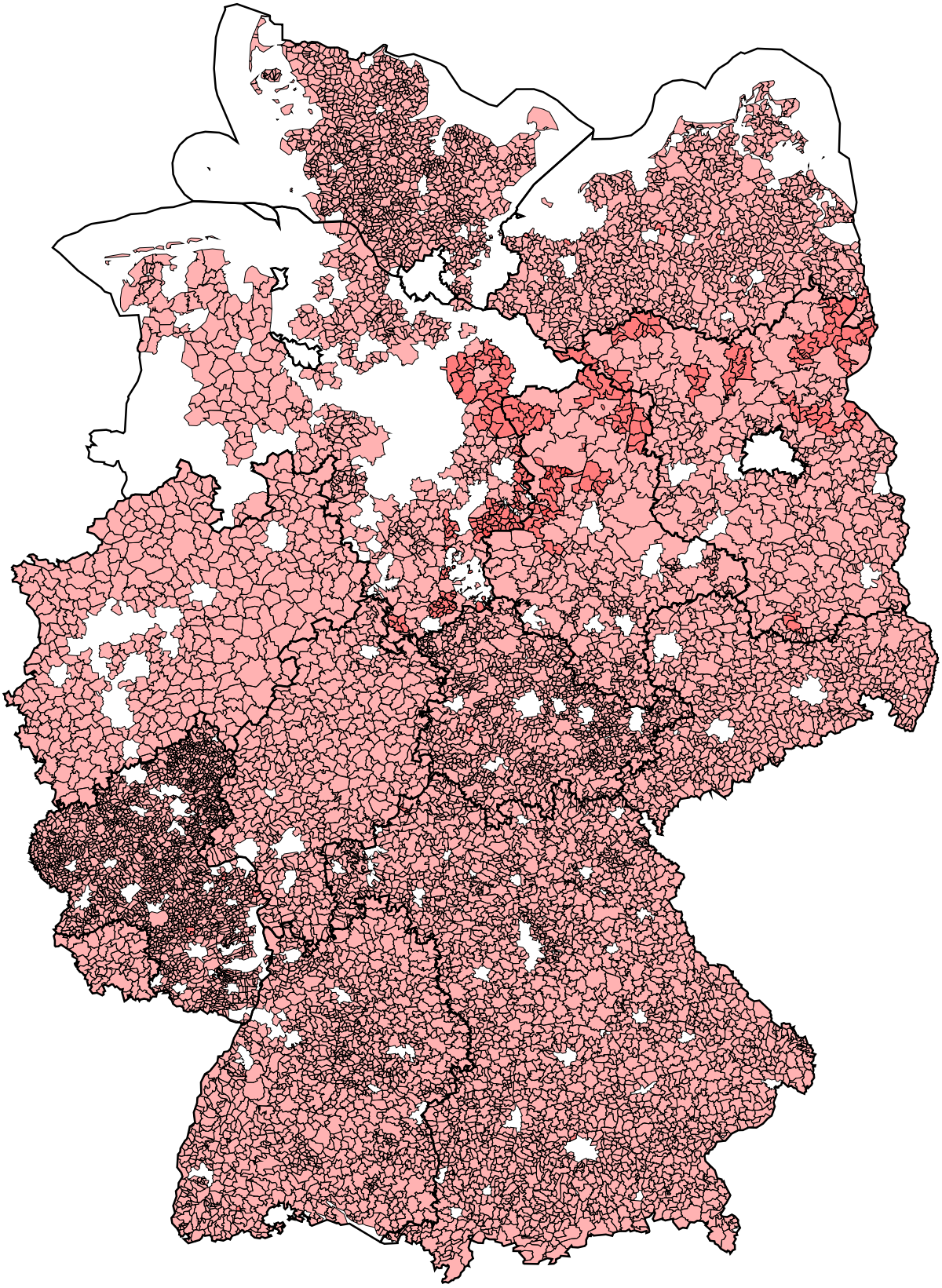


Abbildung 37: Administrative Einheiten auf Ebene 8

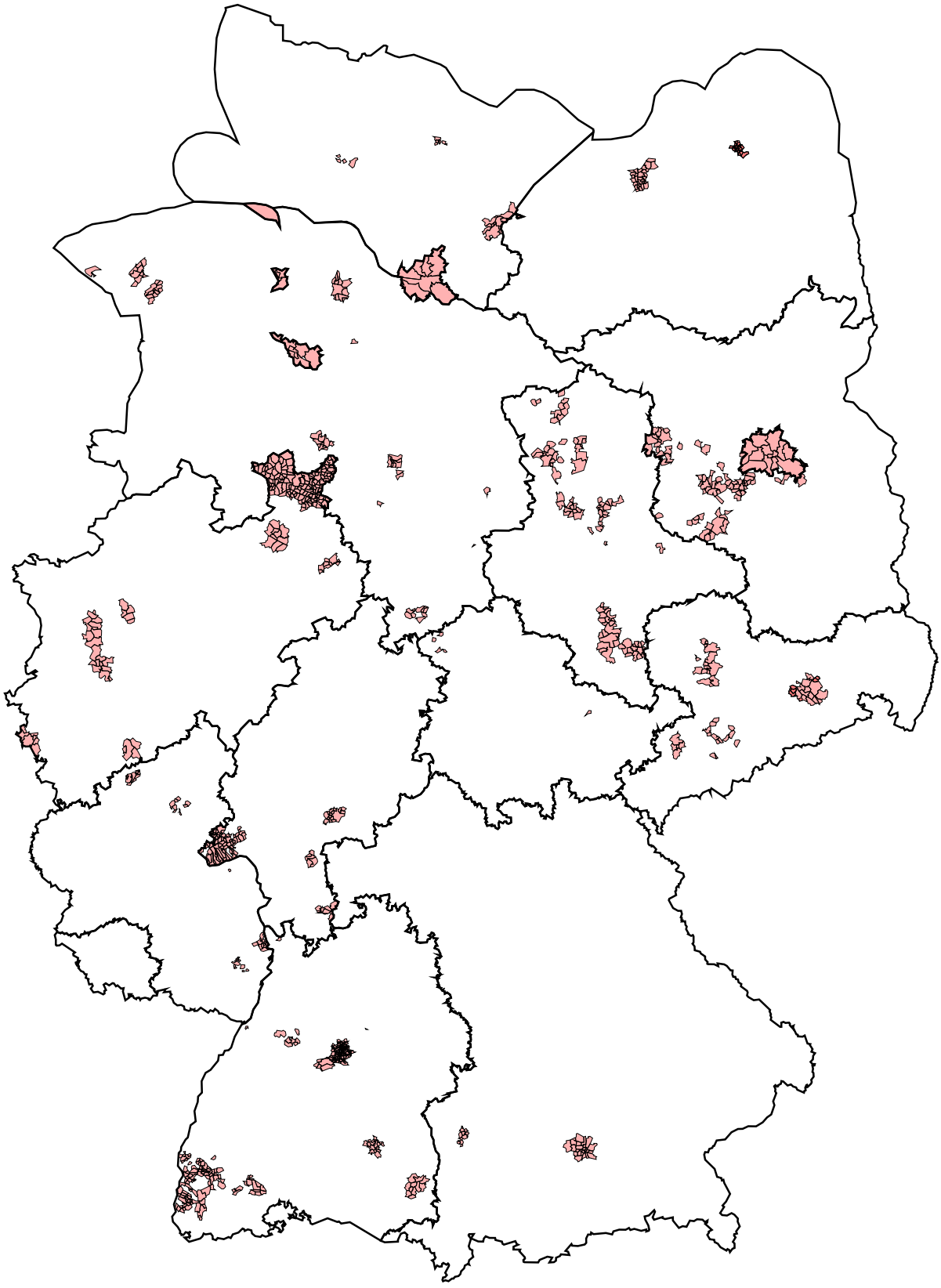


Abbildung 38: Administrative Einheiten auf Ebene 9

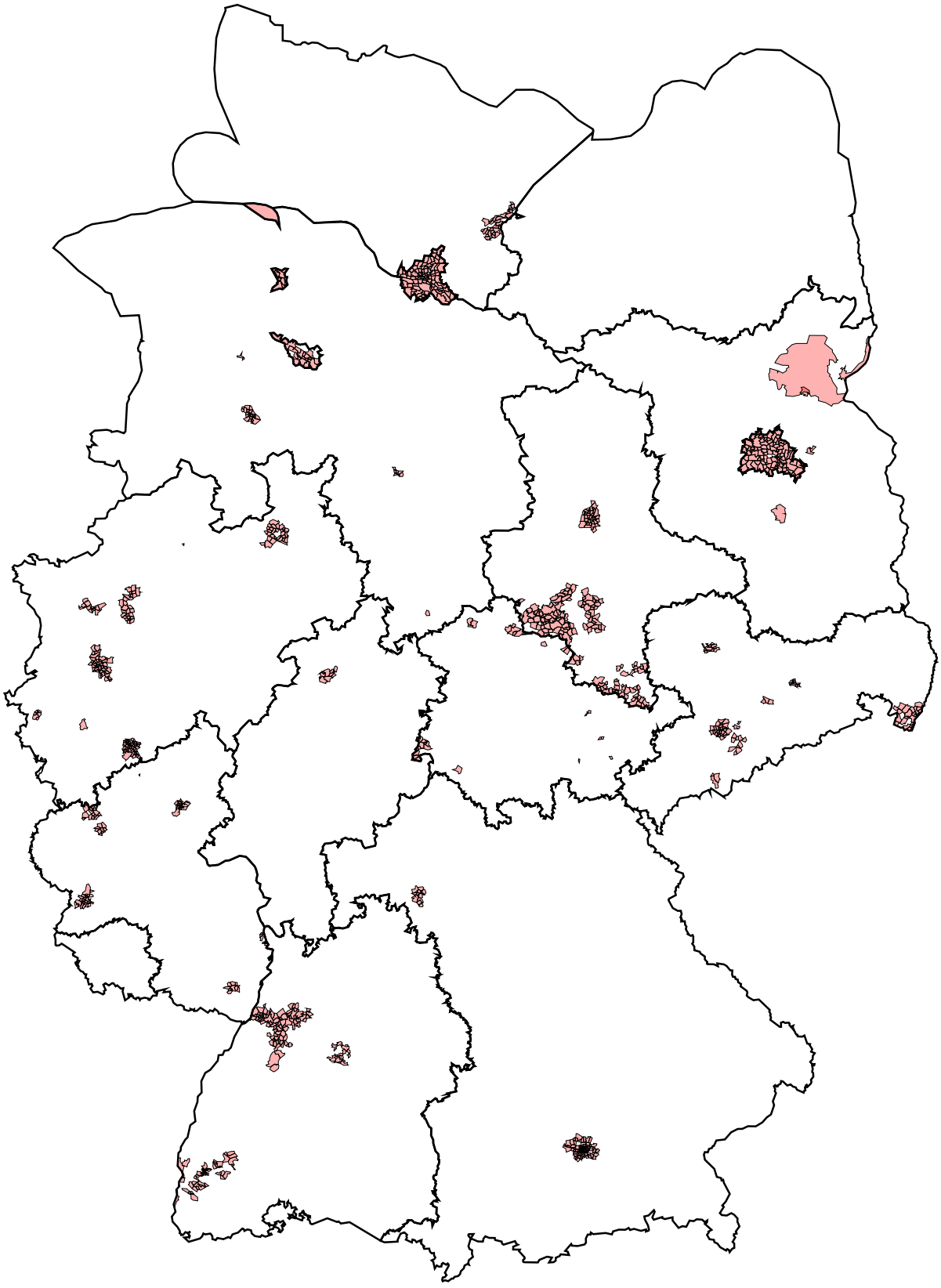


Abbildung 39: Administrative Einheiten auf Ebene 10

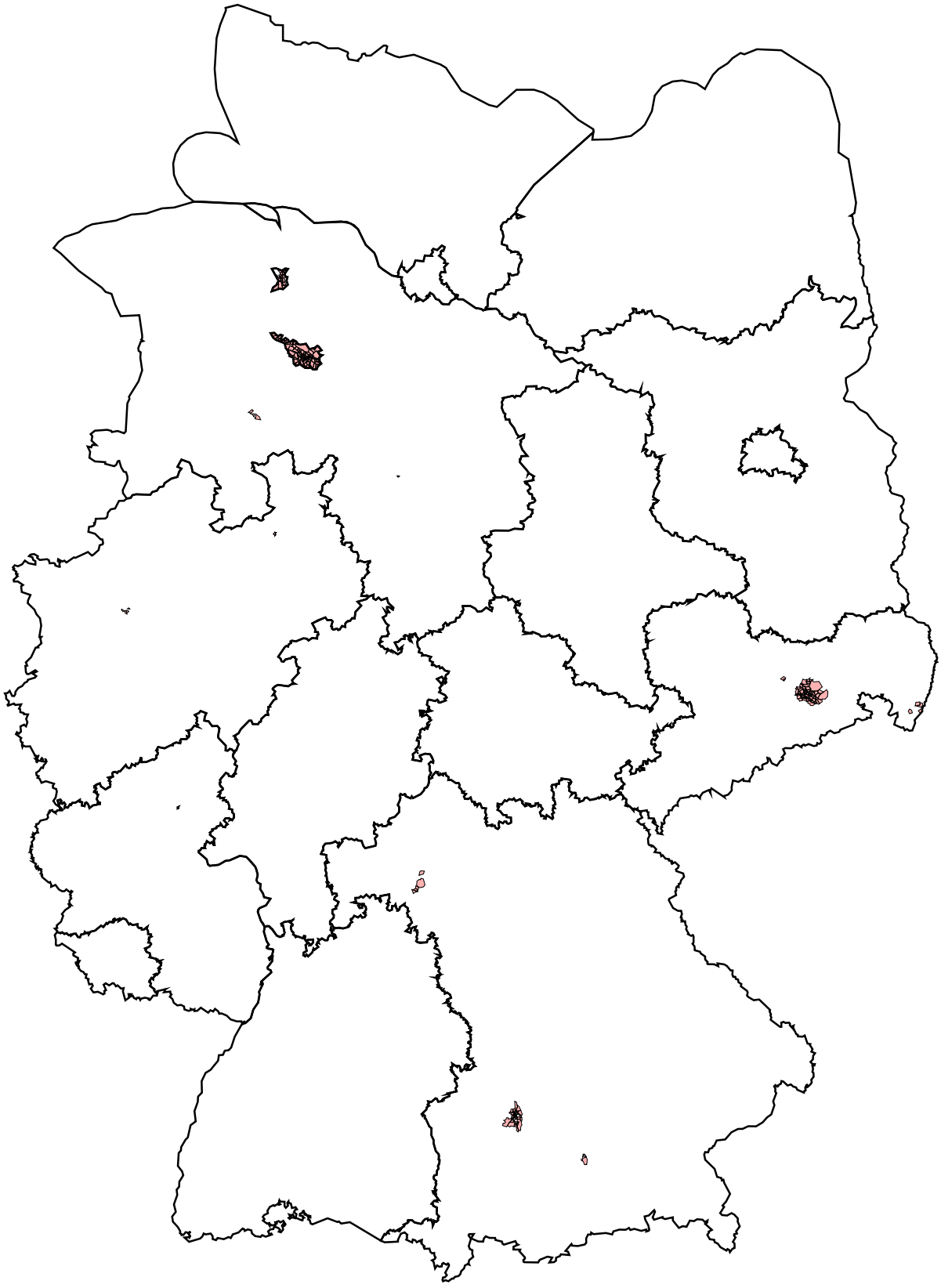


Abbildung 40: Administrative Einheiten auf Ebene 11

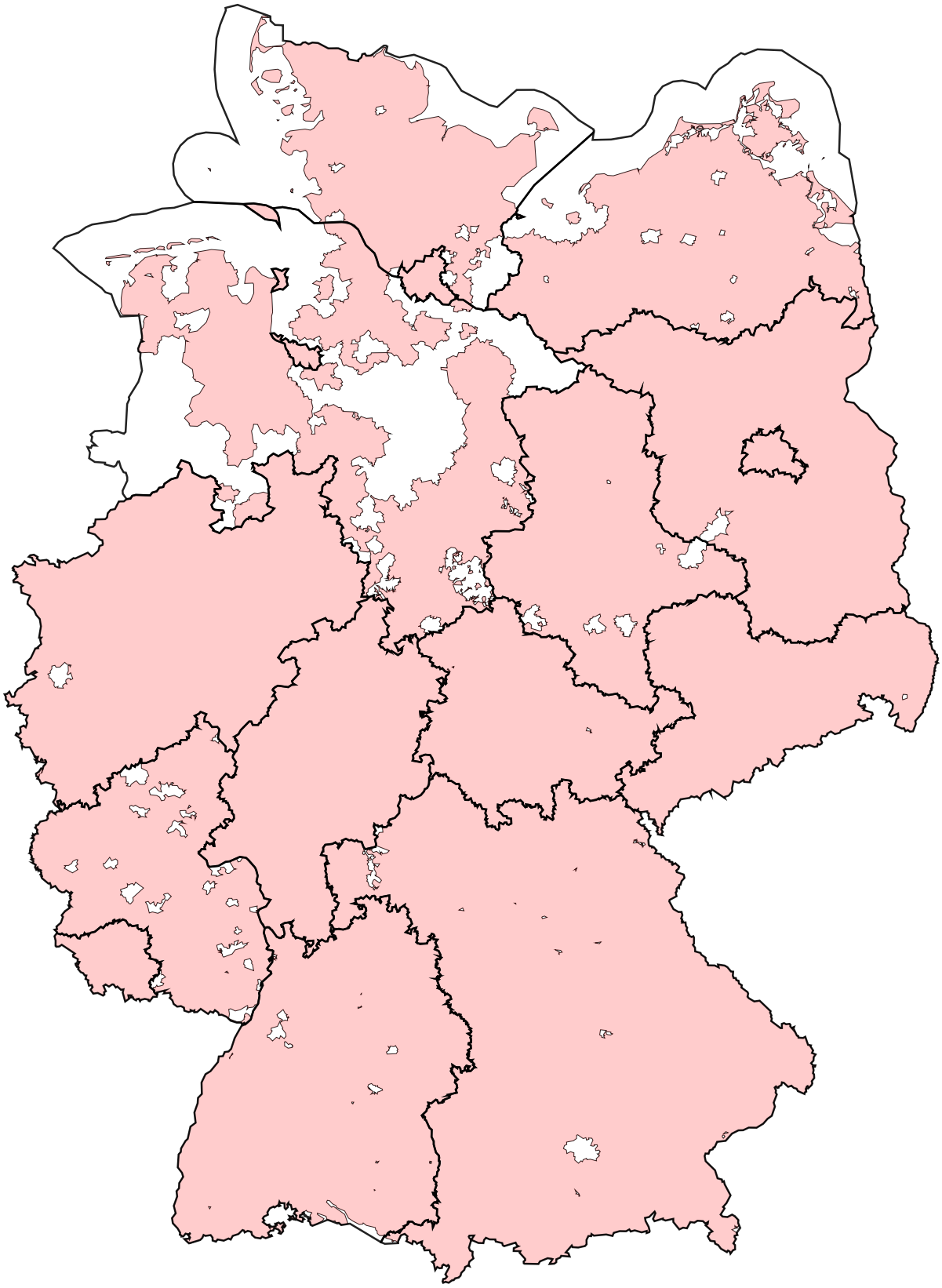


Abbildung 41: Über Gemeindegrenzen erfasstes Gebiet

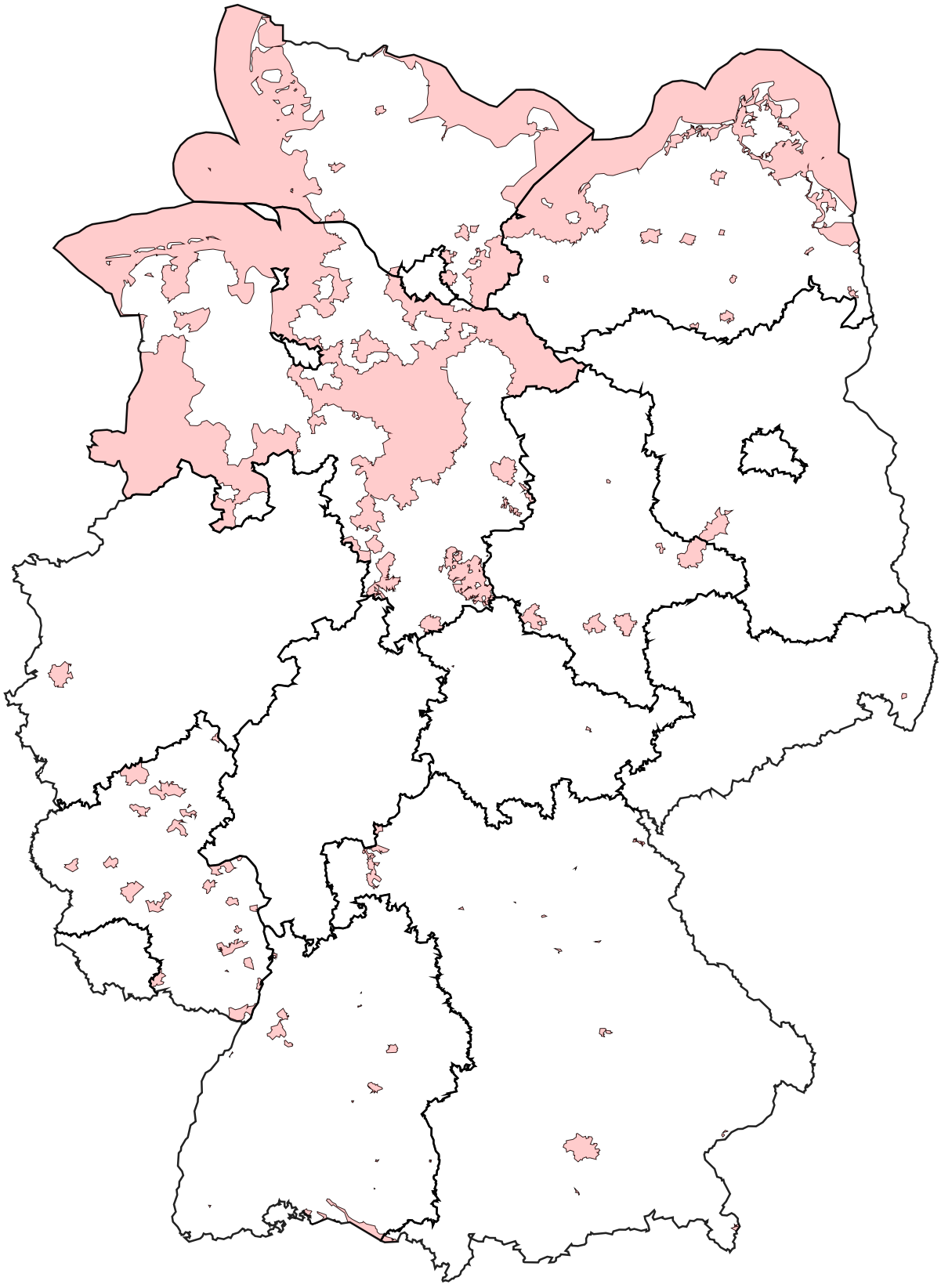


Abbildung 42: Über Gemeindegrenzen *nicht* erfasstes Gebiet

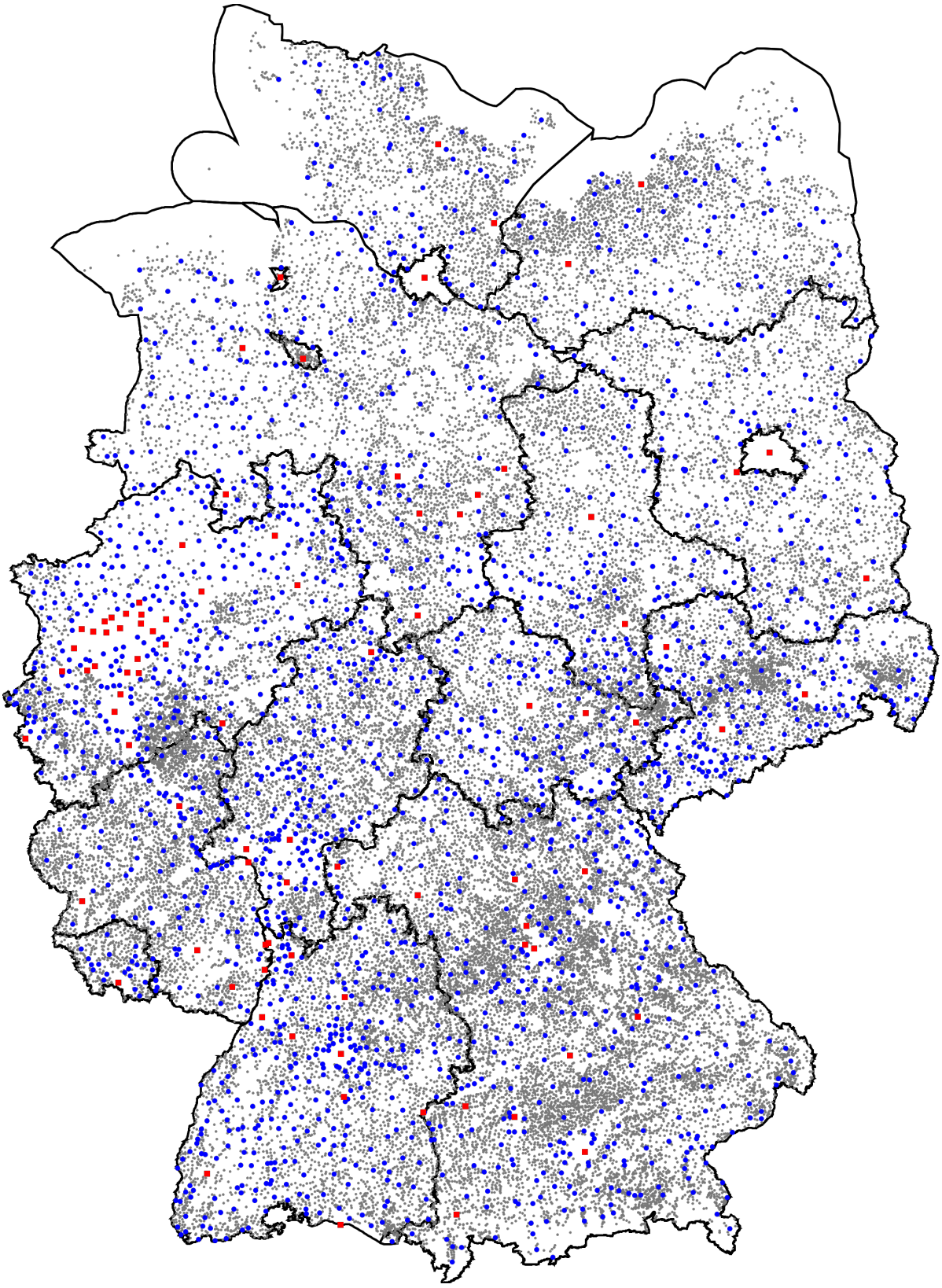


Abbildung 43: Ortsmittelpunkte: Großstädte, Städte und Dörfer