

# Hochschule für Technik Stuttgart

## Bachelor-Arbeit

Als PL nach der SPO-2012

Ausgeführt für die Bachelor-Prüfung im  
Wintersemester 2015/16

**Entwicklung eines Datenmodells zur  
Fußgängernavigation auf Basis von  
OpenStreetMap-Daten**

Erstprüfer und Betreuer:

Zweitprüfer:

Prof. Dr.-Ing. Volker Coors

Dipl.-Ing. Frederik Ramm

## Zusammenfassung

In der vorliegenden Arbeit wurde ein Extraktionsalgorithmus entwickelt, der ein Fußgängerouting basierend auf OpenStreetMap-Daten ermöglicht. Durch Extraktion von Kartendaten werden mögliche Fußgängerwege konstruiert und deren Topografie in einer Datenbank abgelegt. Die Wegfindung auf Basis der gewonnenen Daten wird mit der Routingengine pgRouting realisiert. Im Vordergrund dieser Arbeit steht die Extraktion von Bürgersteiggeometrien und Geometrien zur Überquerung von Straßen.

Um eine spätere Personalisierung des Routings, z.B. eine Beschränkung auf barrierefreie Routen, zu ermöglichen, wurde eine einheitliche Attributierung der Daten angestrebt. Hierfür wurden die bereits bestehenden OSM-eigenen Attribute übernommen und neu erzeugte Geometrien mit Attributen versehen.

Der Algorithmus wurde anhand von Datensätzen in der Stuttgarter Innenstadt und in Biberach an der Riß getestet. Hierbei konnte gezeigt werden, dass das Routing auf dem erzeugten Routinggraphen für neun von zwölf Testrouten plausible Ergebnisse liefert. Im Anschluss wird die Algorithmik an sich sowie Anwendung der erstellten Software für die Verbesserung der Kartendaten diskutiert.

# Abkürzungen

|         |   |
|---------|---|
| API     | Application Programming Interface                 |
| GDAL    | Geospatial Data Abstraction Library               |
| GEOS    | Geometry Engine Open Source                       |
| GIS     | Geoinformationssystem                             |
| GNSS    | Global Navigation Satelite System                 |
| GPX     | Global Positioning System Exchange Format         |
| HFT     | Hochschule für Technik                            |
| ID      | Index / Identifikationsnummer                     |
| KFZ     | Kraftfahrzeug                                     |
| KIT     | Karlsruher Institut für Technologie               |
| NavTech | Navigation Technologies (bis 2004, heute: NAVTEQ) |
| OGC     | Open Geospatial Consortium                        |
| OGR     | OGR Simple Features Library                       |
| OSGEO   | Open Source Geospatial Foundation                 |
| OSM     | OpenStreetMap                                     |
| OSRM    | Open Source Routing Engine                        |
| PedRO   | Pedestrian Routing on OpenStreetMap               |
| RAM     | Random-Access Memory                              |
| SOTM    | State of the Map                                  |
| SQL     | Structured Query Language                         |
| SRTM    | Shuttle Radar Topography Mission                  |

# Inhaltsverzeichnis

|  |           |
|--|-----------|
| <b>Abkürzungen</b>                             | <b>II</b> |
| <b>1 Einleitung</b>                            | <b>1</b>  |
| 1.1 Anwendungsgebiete . . . . .                | 2         |
| 1.2 Eingrenzung der Arbeit . . . . .           | 2         |
| 1.3 Zielsetzung und Validierung . . . . .      | 3         |
| <b>2 Stand der Technik</b>                     | <b>4</b>  |
| 2.1 Navigationssystem . . . . .                | 4         |
| 2.2 Routing in OpenStreetMap . . . . .         | 6         |
| 2.3 Fußgängerrouting . . . . .                 | 7         |
| 2.4 Indoor-Mapping in OpenStreetMap . . . . .  | 8         |
| 2.5 Multimodales Routing . . . . .             | 8         |
| <b>3 Idee</b>                                  | <b>9</b>  |
| 3.1 Verwendete Technologien . . . . .          | 9         |
| 3.1.1 Osmium . . . . .                         | 9         |
| 3.1.2 PosgreSQL und PostGIS . . . . .          | 10        |
| 3.1.3 pgRouting . . . . .                      | 11        |
| <b>4 Konzept</b>                               | <b>12</b> |
| 4.1 Testgebiete . . . . .                      | 13        |
| 4.2 Attributierung in OpenStreetMap . . . . .  | 14        |
| 4.3 Attribute in der Datenextraktion . . . . . | 15        |
| 4.4 Datenmodell . . . . .                      | 16        |
| <b>5 Implementierung</b>                       | <b>18</b> |
| 5.1 Extraktionsalgorithmen . . . . .           | 18        |
| 5.1.1 Direktextrakte . . . . .                 | 18        |
| 5.1.2 Bürgersteige . . . . .                   | 20        |
| 5.1.3 Überwege . . . . .                       | 21        |
| 5.1.4 Kontrastverfahren . . . . .              | 22        |
| 5.2 Komplexitätsanalyse . . . . .              | 24        |

|   |           |
|---|-----------|
| <b>6 Evaluation</b>                             | <b>26</b> |
| 6.1 Erfolgreiche Testrouten . . . . .           | 26        |
| 6.2 Spitze Winkel . . . . .                     | 27        |
| 6.3 Fehlerhafte Überwege . . . . .              | 29        |
| <b>7 Diskussion</b>                             | <b>30</b> |
| 7.1 Beurteilung Kontrastverfahren . . . . .     | 30        |
| 7.2 Krümmungsanalyse . . . . .                  | 30        |
| 7.3 Fehler in den Ausgangsdaten . . . . .       | 30        |
| 7.4 Fazit . . . . .                             | 31        |
| <b>Literatur</b>                                | <b>32</b> |
| <b>A Relevante Attribute der OSM-Geometrien</b> | <b>36</b> |
| <b>B Testgebiete</b>                            | <b>39</b> |
| <b>C Speicherbedarf der Objekte in PedRO</b>    | <b>41</b> |
| <b>D UML-Diagramm</b>                           | <b>42</b> |

# 1 Einleitung

Die einfachste und älteste Möglichkeit, das Wegfindungsproblem zu bewältigen, ist es, einen Ortskundigen nach dem Weg zu fragen. Mit OpenStreetMap (OSM) steht heute eine große Menge ortsgebundener Informationen zur Verfügung. OSM sind Kartendaten, also eine Abbildung der Wirklichkeit auf einen bestimmten Raum auf der Erdoberfläche. Diese Informationen werden weltweit von freiwilligen Ortskundigen gesammelt. Gleichzeitig wachsen die technischen Möglichkeiten durch mobile Endgeräte, die in der Lage sind, Karten darzustellen, Wegbeschreibungen sprachlich zu vermitteln und Positionsbestimmungen durchzuführen. Aus dem Umfang der frei verfügbaren Karteninformationen und den Möglichkeiten der modernen Informationstechnologie ergeben sich eine Vielzahl von Fragestellungen bezüglich der Verarbeitung dieser Informationen.

Kommerzielle Navigationslösungen für Fußgänger, wie Google Maps, leisten hilfreiche Dienste. Eine Adresse oder eine Sehenswürdigkeit zu finden gelingt erfahrungsgemäß gut. Der große Kartenanbieter hat zudem den Vorzug, Bewegungsdaten im großen Stil auswerten zu können. So lassen sich zum Beispiel sehr aktuelle Stauinformationen berechnen. Durch die Auswertung der GNSS-Positionen mehrerer Nutzer kann empirisch ermittelt werden, wo sich Fußgänger bewegen. Dadurch können mögliche Fußwege definiert werden. Aber auch Routing-Dienste, die auf den Kartendaten von OSM basieren, sind für viele Anwendungen ausreichend. Sie beziehen neben den KFZ-Wegen noch die Fußgängerwege ein und routen anhand dieser Geometrien.

Die vorliegende Arbeit ist ein Versuch, gegenüber den bestehenden Systemen, präziser zu sein. Im Vordergrund steht nicht die Frage, wie ein bestimmter Ort zu erreichen ist, sondern, wo bewegt sich jemand, um ein bestimmtes Ziel zu erreichen. Um diese Frage zu beantworten, muss die jeweilige Straßenseite berücksichtigt werden, auf der sich der Fußgänger bewegt. Da in den OSM-Daten in der Regel keine Bürgersteiggeometrien vorhanden sind, wird ein Algorithmus entwickelt, der diese konstruiert. Werden Straßenseiten berücksichtigt, kommen auch Straßenüberquerungen neue Bedeutung zu. Die im Zuge der Arbeit entwickelte Software synthetisiert diese Informationen aus den vorhanden Kartendaten und erzeugt daraus einen Routinggraphen.

## 1.1 Anwendungsgebiete

Das Interesse für Fußgängernavigationssysteme ist vielseitig. Zunächst gibt es den klassischen Endverbraucher, der eine einzelne Route sucht und mithilfe einer Software möglichst bequem und verständlich zu einem Ort gelotst werden möchte. Im Freizeitbereich gibt es immer mehr Lösungsangebote, sei es für Touristen oder für Sportler. Ein touristisches Bedürfnis ähnelt dem Traveling-Salesman-Problem (vgl. Wikipedia, 2016b). Er will möglichst viel von einer Stadt sehen und hat favorisierte Orte, die er besucht haben will. Ein Bergsteiger will vielleicht möglichst sportlich von A nach B oder will einen berühmten Klettersteig nicht verpassen.

Aber auch in der administrativen Ebene der Stadtverwaltung werden Routinglösungen benötigt. Geht es um den sicheren Schulweg der Schüler einer bestimmten Schule oder die Erreichbarkeit von Bushaltestellen, versagen klassische Routingmaschinen schnell, da sie zu unpräzise an den vorhandenen Verkehrswegen der KFZ entlang navigieren. Wird für diesen Zweck auf OSM zurückgegriffen, ist eine genauere Analyse der Kartendaten notwendig.

## 1.2 Eingrenzung der Arbeit

Innerhalb des Funktionszusammenhangs eines Navigationssystems liegt der Fokus der Arbeit auf der automatischen Generierung des Routinggraphen. Dazu wird ein Datenmodell entwickelt, das ein Routing ermöglicht und das die für die Navigation relevanten Informationen abbilden kann. Das Modell bezieht sich auf eine planare Projektion der Erdoberfläche und wird keine übereinanderliegenden Ebenen berücksichtigen, wie es beispielsweise ein Indoor-Routing benötigt.

Die Datengrundlage bilden Kartenauszüge aus OSM. Diese stehen für jeden beliebigen Ort kostenlos zu Verfügung. Die Daten verfügen jedoch nicht über alle relevanten Geometrien. Teilweise können die Geometrien direkt aus den Daten übernommen werden. Diese werden in routingfähige Geometrien umgewandelt. Die anderen Geometrien, die Bürgersteige und Überwege, müssen heuristisch erzeugt werden. Es wird versucht, diese Geometrien möglichst realitätsnah zu synthetisieren. Selektiert werden die Daten anhand des OSM üblichen Taggings (OSM Community, 2015). Eine Analyse der Vollständigkeit und Qualität der Attributierung soll in diesem Rahmen nicht erfolgen. Dennoch können während der Arbeit mit Testdatensätzen

Schwachstellen auffallen, sowohl in der Praxis des Straßenmappings, als auch in den offiziellen Vereinbarungen darüber. Ob und in wie weit die entwickelte Software die Qualität von OSM-Daten steigern kann, wird im Anschluss diskutiert.

Die Algorithmik des eigentlichen Routings ist nicht Thema der Arbeit und soll mit pgRouting realisiert werden. Für die Testrouten stellt pgRouting den Dijkstralgorithmus (vgl. Lang, 2012, S. 138f) zur Wegfindung bereit, welcher in den Dimensionen der Testgebiete ausreichend performant ist.

### 1.3 Zielsetzung und Validierung

Das erste Ziel ist, einen routingfähigen Graphen zu erzeugen. Dafür werden die Daten in ein Format gebracht, das pgRouting versteht. Fehlerhafte Geometrien sollten im Graphen möglichst vermieden werden. Diese wirken sich teilweise direkt auf das Routing aus, indem das gesuchte Ziel nicht oder nur über Umwege erreichbar ist. Teilweise wirken sich Geometriefehler, wie selbst schneidende Linien, lediglich auf die Distanz des Weges aus. Ein weiteres Ziel ist eine funktionierende Attributierung. Dabei sollen die Informationen, wie Straßename und -typ, im AusgabefORMAT erhalten bleiben. Des Weiteren soll zu jeder Geometrie die Länge gespeichert sein. Die Attribute ermöglichen eine personalisierte Zielführung. Exemplarisch sollen hierfür mindestens zwei Präferenzmuster umgesetzt werden.

Wünschenswert, auch hinsichtlich einer Interoperabilität der Software, ist die Implementierung der Software als OGC-konformen Location Service (OGC, 2015). Erweiterungen der entwickelten Software PedRO<sup>1</sup> (Pedestrian Routing on Openstreetmap - Arbeitstitel) sind in vielerlei Hinsicht denkbar. Beispielsweise sind in OSM die beleuchteten Straßen getaggt. Diese Daten sind in OSM selten vollständig. Trotzdem wäre diese Information natürlich für ein Fußgängerouting interessant.

Zur Überprüfung der Ergebnisse dient ein Vergleich verschiedener Routen mit den Ergebnissen Navigationsanbieters OpenRouteService. Des Weiteren werden die Routen vor Ort auf ihre Begehbarkeit überprüft. Dabei soll die Mehrheit der Wege begehbar sein und einige optimiert gegenüber einem bestehenden Routingsystem. Insgesamt ist das Ziel, einen Routinggraphen zu erzeugen, der ein hohen Realitätsbezug

---

<sup>1</sup><https://github.com/Nathanael-L/pedro>

aufweist.

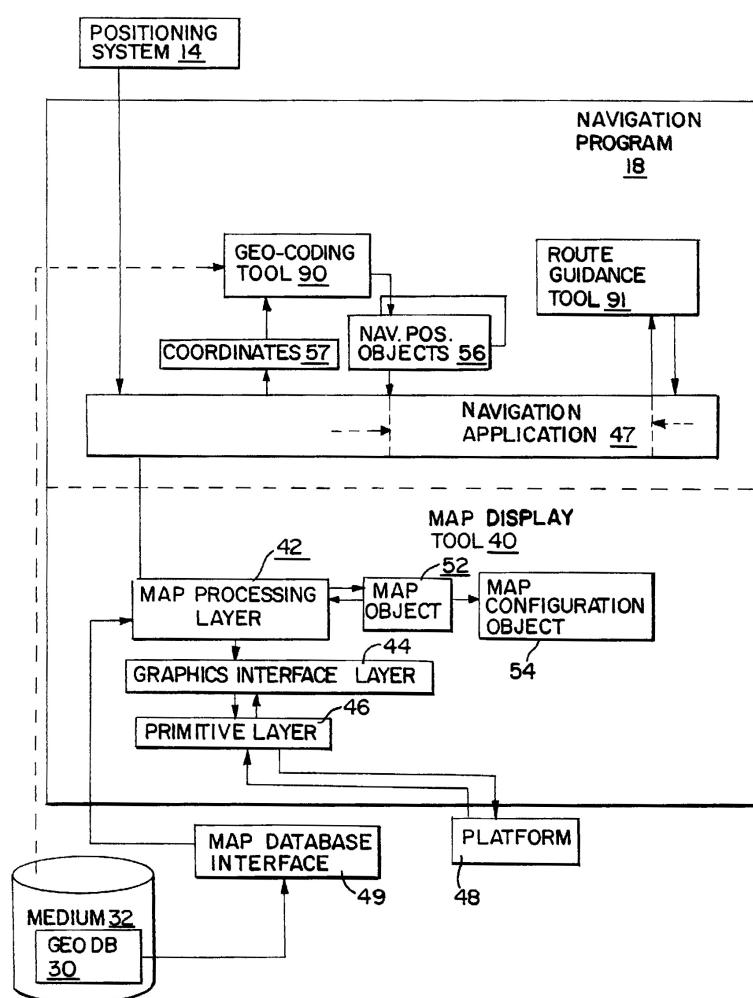
## 2 Stand der Technik

Das Thema des Routings und der Navigation umfasst viele Aspekte. Im Folgenden sollen Navigationssysteme allgemein und die Fußgängernavigation im Speziellen skizziert werden – welche Komponenten umfassen die Systeme, welche Besonderheiten spielen im Fußgängerouting eine Rolle und welche Erkenntnisse wurden bisher erzielt.

### 2.1 Navigationssystem

Die Spezifikation von NavTech (Navigation Technologies Corporation) in dem Patent „Method and System for Map Display in Navigation Applications“ formulierte Mc Donough el al. (2000) ein Navigationssystem folgendermaßen. Abbildung 1 zeigt ein Flussdiagramm aus dem Patent, das die Komponenten einer Navigationslösung und deren Zusammenhänge darstellt. Das System besteht aus zwei Teilen: das *navigation software programm* (oben) und das *map display tool* (unten). Beide beziehen ihre Daten aus einer Geodatenbank. Die Navigationssoftware ermittelt anhand einer Positionsbestimmung und dem Geocoding die aktuelle Positionen des Systems. Die relevanten Geoobjekte werden von der Datenbank abgefragt und topologisch gespeichert. Ein Routingalgorithmus *route guidance tool* ermittelt den gesuchten Weg. Die Eingangsparameter des Algorithmus sind die Start- und Endkoordinaten. Anhand der topologischen Information ermittelt der Algorithmus den kürzesten bzw. optimalen Weg und gibt ihn als Routingpfad aus. Der Pfad enthält Geometrieinformationen und Sachdaten, z.B. Straßennamen für die Wegbeschreibung und wird vom *map display tool* zur Zielführung verwendet. Diese verwendet in den meisten Fällen eine Wegbeschreibung, eine Visualisierung auf einer Karte oder eine Kombination aus beiden. Das *map display tool* bezieht die Daten von der Geodatenbank zur Darstellung der Karte. Der *map processing layer* ist über eine Schnittstelle mit der Geodatenbank verbunden und erzeugt die graphische Darstellung. Über weitere Schichten wird die Benutzerinteraktion ermöglicht. Dies betrifft die Anpassung der

FIG. 6



**Abbildung 1:** Grundprinzip eines Navigationssystems nach dem Patent der Navigation Technologies Corporation (2000)

Darstellung sowie die Erstellung von Abfragen an das Navigationssystem (Mc Donough et al., 2000, S. 6f).

Um die Zielführung zu überwachen, ist an dieser Stelle eine permanente Positionsbestimmung notwendig. Anhand der ermittelten Position kann die Zielführung mit Hilfe des Routingalgorithmus korrigiert werden.

## 2.2 Routing in OpenStreetMap

Eine verbreitete Routingengine im OSM Umfeld ist die Open Source Routing Engine (OSRM). Erstmals vorgestellt wurde sie auf der State of the Map (SOTM) 2010 (Vetter and Luxen, 2011) und wurde am KIT Karlsruhe entwickelt. Die Performanz der Maschine ist hoch (Project OSRM, 2015), so lassen sich Routen durch ganz Europa serverseitig in unter 50 ms berechnen (Luxen and Vetter, 2011, vgl. Tabelle S. 514). Das Hauptaugenmerk von OSRM liegt nicht nur auf der Generierung des Routinggraphen, sondern auf der Bereitstellung eines performanten Algorithmus. Daher integriert die offizielle Projektseite zur Zeit nur eine PKW-Routenplanung. Andere Navigationslösungen, die auf OSM aufbauen, sind beispielsweise OpenRouteService (ORS Community, 2008), Graphhopper (GraphHopper Community, 2012) und BRouter (BRouter Community, 2013). Diese haben eine umfangreiche Benutzeroberfläche mit einer Vielzahl von Profilen und einen großen Funktionsumfang. Sie integrieren Höhenprofile, können GPX-Pfade exportieren oder stellen touristische Funktionalitäten bereit. Für das Fahrradrouting sind sie inzwischen besonders gut geeignet. Das liegt einerseits daran, dass in der OSM-Szene ein großes Interesse für die Fahrradnavigation besteht und daher Radwege sehr gut gemapped sind. Andererseits lassen sich mit den Höhendaten der Shuttle Radar Topography Mission (SRTM), die für den Fahrradfahrer ausreichend aufgelöst sind, sehr interessante Berechnungen anstellen. Das Fußgängerouting der genannten Realisierung funktioniert. Es extrahiert die Straßen, die laut OSM für Fußgänger begehbar sind, jedoch werden weder Flächen überquert noch Gehwege im Graphen generiert.

## 2.3 Fußgängerrouting

Die Fußgängernavigation stellt an das Routing besondere Ansprüche im Vergleich zu anderen Navigationssystemen. Sind motorisierte Fahrzeuge, der Schienenverkehr und prinzipiell auch Fahrräder immer an linienförmige Straßen gebunden, so ist der Fußgänger in mehrerer Hinsicht freier. Er bewegt sich in drei bis vier Freiheitsgraden und ist dabei weniger an Wege gebunden. Zu Fuß lässt sich zum Beispiel ein Wald, je nach Steigung und Vegetationsdichte, abseits von befestigten Wegen durchqueren. Fußgängerspezifische Aspekte sind:

- Der optimale Weg ist individueller. So gibt es persönliche Einschränkungen, die beispielsweise das Treppengehen erschweren. Auch könnte eine Navigationslösung Präferenzen beachten, wie „nachts nur auf beleuchteten Straßen“ oder „bevorzugt durch die Altstadt gehen“. In diesem Zusammenhang wird von Personalisierung gesprochen.
- Das lineare Routing kann um ein flächenhaftes ergänzt werden.
- Das Finden der Start- und Endpunkte im Routinggraphen gewinnt aufgrund niedriger Geschwindigkeiten an Bedeutung.
- Höhen können berücksichtigt werden. Dies spielt besonders für das Indoor-Routing eine Rolle.
- Die Auflösung der SRTM-Daten von ca. 30 m reichen für einige Fragestellungen nicht mehr aus. Denkbar wäre eine Berücksichtigung der Randsteinhöhe oder die Steigung der Treppenstufen.
- Die Zielführung und dessen Überwachung durch eine Positionsbestimmung, kann vor allem in Innenräumen nicht oder nicht ausreichend mit GNSS gelöst werden.
- Die Wegbeschreibung und auch die Visualisierung muss anders gedacht werden als bei einer straßengebundenen Navigation. Um den Fußgänger anzuleiten, werden häufig Landmarken eingesetzt. Landmarken sind wiedererkennbare Objekte, wie auffällige Gebäude, die von einem bestimmten Standort aus sichtbar sind.

Bauer et al. (2014, S. 408-413) veröffentlichten ein Verfahren, das Routing auf Flächen ermöglicht und optimiert. Sie verwendeten dafür ebenfalls OSM-Daten. Ein weiterer Aspekt, der in der Arbeit angesprochen wird, ist das Finden des Start- und Endknotens im Routinggraphen. „Bei Standardnavigationsprogrammen ist es meist [...] nicht möglich einen selbstbestimmten Start- und Endpunkt koordinatengenau zu wählen, da der nächste im Routinggraph befindliche Punkt als Start- oder Endpunkt verwendet wird“ (Bauer et al., 2014, S.408). Dies muss bei der Erzeugung des Graphen berücksichtigt werden.

## 2.4 Indoor-Mapping in OpenStreetMap

Das Kartieren von Innenräumen ist in OSM eine noch sehr junge Praxis, die sich noch im Wachstum befindet. Zum jetzigen Zeitpunkt gibt es knapp 40000 Ways mit dem Tagging *indoor=yes* - im Vergleich dazu gibt es 5,8 Mio Ways mit *highway=footway* (TagInfo, 2016). Es existieren bereits technische Umsetzungen für Systeme, Innenräume auf der Karte darzustellen. So können mit Open Level Up (OLU Community, 2016) die einzelnen Etagen eines Gebäudes dargestellt werden. Im OSM-Wiki (OSM Community, 2016) sind unter anderem folgende Ziele der Innenraumkartierung aufgeführt: die Darstellung von Fluchtwegkarten, Indoor-Navigation und 3D-Visualisierung.

In Innenräumen gewinnen Landmarken aufgrund der eingeschränkten oder veränderten Sicht eine neue Rolle. Auch die Darstellung der dritten Dimension auf dem Endgerät ist vor allem in geschlossenen Räumen besonders hilfreich. OSM hat sich auf die Erfassung flacher Geometrien beschränkt. Dennoch können auch mit diesen Daten 3D-Modelle generiert werden. Goetz (2012) entwickelt ein webbasiertes Datenmodell zur Visualisierung von OSM-Innenräumen.

## 2.5 Multimodales Routing

Multimodales Routing stellt einen Trend in der aktuellen Forschung dar (Bernhard et al., 2012) (Erdmann, 2013) (Kuntzsch, 2015) (Stark and Torlach, 2003). Vorwiegend Gemeinden und Verkehrsverbünde haben großes Interesse an dieser Entwicklung. Dabei geht es einerseits um den Abbau des Individualverkehrs und um

die damit verbundene Reduzierung des CO<sub>2</sub>-Ausstoßes (vlg. Bernhard et al., 2012). Andererseits besteht großes Interesse, Menschen mit eingeschränkter Mobilität zu unterstützen (Bohner-Degrell and Köhler, 2014) (Geue et al., 2014) (Schuster, 2012). Für ein multimodales Routing ist ein ausgereiftes, parametrisierbares Fußgängerouting unabdingbar.

## 3 Idee

Die Idee ist es, ein Fußgängerouting aufgrund der OSM-Daten zu ermöglichen, ohne dass dazu die Bürgersteige in den Daten enthalten sein müssen. Ein Programm erzeugt aus einem OSM-Datensatz einen Routinggraph, der Fußege, Bürgersteige und Straßenüberquerungen enthält. Die konstruierten Wege werden anhand des Taggings, also anhand der vorhandenen Attribute der Verkehrswegegeometrien, konstruiert.

Die verwendeten Kartendaten sind für viele weitere Anwendungen eines Fußgängeroutings nützlich. Beispielsweise können Points of Interest als Landmarken in das System einbezogen werden. Daten des öffentlichen Nahverkehrs können verwendet werden, um ein multimodales Routing zu ermöglichen. Der entscheidende Vorteil von OSM in diesem Zusammenhang ist, dass ein einzelner Datensatz alle relevanten Informationen enthält und keine zusätzlichen Daten herangezogen werden müssen.

### 3.1 Verwendete Technologien

Im Folgenden werden die Technologien vorgestellt, die für die Umsetzung der Datensynthese und für das anschließende Routing zum Einsatz kommen.

#### 3.1.1 Osmium

Für eine effiziente Verarbeitung der Ausgangsdaten kommt die Osmium-Bibliothek zum Einsatz. Sie wurde von Jochen Topf in der Geofabrik entwickelt. „Osmium ist ein schnelles und flexibles C++- und Javascript-Toolkit für das Arbeiten mit OSM-Daten.“ (Topf, 2012, Minute 4:26) Verwendet wird sie beispielsweise von TagInfo, einem Angebot über umfangreiche Statistiken der OSM-Daten (TagInfo, 2016). Die Datenmenge von fast 60 GB wurde auf dem TagInfo-Server innerhalb von 2 Stunden

berechnet (2012).

Alternativ zu der Verwendung von Osmium wird vielfach auf datenbankbasierende Verarbeitungsmethoden zurückgegriffen. Der Flaschenhals dieses Vorgehens sind dabei die häufigen Zugriffe auf die Festplatte. Dies ist besonders in Anbetracht der stetig wachsenden Datenmenge von OSM bedeutsam und führt dazu, dass einige Anwendungen nicht mehr performant genug sind, um beispielsweise tägliche Berechnungen über die gesamten Daten durchzuführen.

Der entscheidende Vorteil der Osmium-Bibliothek besteht im Streaming-Prinzip. Die Daten werden während des Einleseprozesses gefiltert und in den Arbeitsspeicher geschrieben. Die Verarbeitung der relevanten Daten geschieht entweder während oder im Anschluss des Leseprozesses. Dadurch reduzieren sich die langsamten Zugriffe auf die Festplatte auf das Lesen der Eingabedaten und auf das Schreiben der Ausgabedaten. Bei Ersterem handelt es sich um ein sequenzielles Lesen, wofür keine Sprünge auf dem physikalischen Datenträger nötig sind (vgl. Topf, 2012).

Osmium ist als erweiterbares Baukastensystem zu verstehen. Für die drei Grunddatentypen von OSM - Node, Way, Relation - können Handler geschrieben werden. Beim Einlesen der Daten erzeugt jede Entität des jeweiligen Datentyps einen Callback. Der Handler entscheidet anhand der Attribute (Tags) des Objektes, ob es für die weitere Verarbeitung von Bedeutung ist und kann alle vorhandenen Informationen abfragen. Einige dieser Handler sind bereits vorhanden. Von Bedeutung für die Arbeit ist der *LocationHandler*, der alle Koordinaten des Datensatzes in den RAM lädt.

Die Bibliothek integriert unter anderem Schnittstellen zu OGR (GDAL, 2015) und GEOS (OSGEO, 2015). OGR ist eine zur GDAL Familie gehörende API, die ein Umwandeln von Geodaten in Formate wie Shape (ESRI), PostGIS und SQLite ermöglicht. GEOS stellt einige GIS-typische Geometrieoperationen zur Verfügung. Beide Bibliotheken kommen in der vorliegenden Arbeit für die Erzeugung neuer Wegegeometrien zum Einsatz.

### 3.1.2 PostgreSQL und PostGIS

Die routingfähigen Geometrien sowie die zugehörigen Sachdaten werden in eine PostgreSQL-Datenbank geschrieben. „PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and

a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness“ (PostgreSQL Community, 2016). Das Datenbanksystem ist sehr ausgereift und kommt auch mit großen Datenmengen zurecht. Außerdem sind die für den Anwendungsfall notwendigen Erweiterungen vorhanden. Mit der PostGIS-Erweiterung erhält die Datenbank umfangreiche GIS-Funktionalitäten. Sie ermöglicht das Speichern aller Simple Features und von georeferenzierten Rasterdaten. „Das Projekt implementiert die Simple Feature Access-Spezifikation des Open Geospatial Consortium“ (Wikipedia, 2016a), erweitert PostgreSQL um eine Vielzahl von Geometrieoperationen und ermöglicht das Anlegen von räumlichen Indizes. Damit ist gemeint, dass die Indizes in einer Baumstruktur gespeichert sind, so dass sich die Laufzeit für das Finden einer Geometrie logarithmisch zur Datenmenge verhält.

### 3.1.3 pgRouting

Der wesentliche Partner im Zusammenspiel mit PedRO ist pgRouting. Es stellt die Routingengine dar, die den Hauptteil der Arbeit leistet, die zu entwickeln kompliziert und zeitaufwändig wäre. In pgRouting sind einige Routingalgorithmen integriert (vgl. pgRouting Community, 2016a). Die Technologie ist gut dokumentiert und eine Einführung erleichtert der Einstieg (pgRouting Community, 2016b). pgRouting ist, wie PostGIS, eine Erweiterung für PostgreSQL. Um eine routingfähige Datenbank zu erstellen, müssen zunächst die beiden Erweiterungen eingebunden werden (vgl. Listing 1).

**Listing 1:** Erweiterungen in PostgreSQL laden

```
CREATE EXTENSION postgis;
CREATE EXTENSION pgrouting;
```

Der nächste Schritt besteht darin, den Routinggraphen mittels PedRO in die Datenbank zu schreiben. Es wird die Tabelle *ways* erstellt, die für pgRouting leicht angepasst werden muss, um dieses die Topologie berechnen zu lassen (vgl. Listing 3). Dem Datensatz werden die Spalten *source* und *target* hinzugefügt. Darin werden Start- und Endpunkte der Geometrien über eine Nummer miteinander verknüpft.

**Listing 2:** Tabelle anpassen und Topologie rechnen

```
ALTER TABLE ways RENAME COLUMN geom TO the_geom;
ALTER TABLE ways ADD COLUMN "source" integer;
ALTER TABLE ways ADD COLUMN "target" integer;
SELECT pgr_createTopology( 'ways' , 0.00001 , 'the_geom' , 'gid' );
```

Nun ist die Datenbank in der Lage, zwischen zwei Knoten anhand ihrer ID zu routen. In der Praxis wird dagegen meist zwischen zwei Koordinaten geroutet, bzw. zwischen zwei geocodierten Orten. Im Rahmen des erwähnten Workshops wird eine Funktion namens *fromAtoB* vorgestellt. Wird diese in der Datenbank erzeugt, kann mit einer Zeile zwischen zwei Koordinaten geroutet werden:

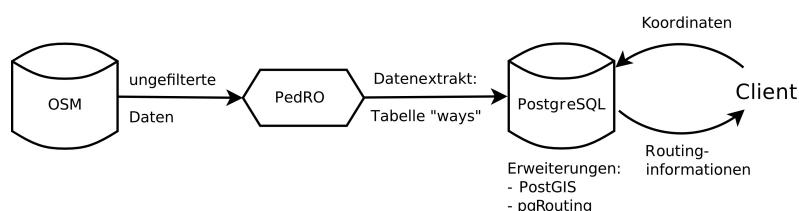
**Listing 3:** Tabelle anpassen und Topologie berechnen

```
CREATE TABLE rte_hbf_hft AS SELECT * FROM pgr_fromAtoB( 'ways' ,
9.181 , 48.784 , 9.173 , 48.781);
```

Erzeugt wird die Route in einer neuen Tabelle. Für jedes Linienstück zwischen zwei Knoten kreiert pgRouting einen Eintrag in der Tabelle mit den entsprechenden Geometrien und Sachdaten.

## 4 Konzept

Abbildung 2 zeigt den grundsätzlichen Funktionszusammenhang zwischen der eigenen Software PedRO und den anderen Technologien. Die Eingabe ist ein OSM-Datensatz, der innerhalb der Software mithilfe der osmium-Bibliothek eingelesen wird. Im Arbeitsspeicher werden die Koordinaten aller Nodes, die entnommenen und die synthetisierten Wege gespeichert. Dadurch entsteht eine hohe Auslastung der Arbeitsspeichers (vgl. Kapitel 5.2).



**Abbildung 2:** Schematische Darstellung der Datenverarbeitung des OSM-Datensatzes und der Ausgabe des Routinggraphen in die Ausgabetafel.

Aus den Rohdaten werden die relevanten Geometrien mit ihren Sachinformationen gelesen und weiterverarbeitet. Erzeugt wird eine Topographie, die die routingfähigen Geometrien enthält. Die Ausgabe ist eine Datenbanktabelle<sup>2</sup>, mit der pgRouting in

<sup>2</sup>Aus Performanzgründen wird hier ein Umweg in Kauf genommen. Zunächst wird die Ausgabe in das Shape-Format geschrieben und in einem Zwischenschritt mit shp2pgsql in die Datenbank eingelesen.

der Lage ist, das Routing durchzuführen. Aus der Topografie wird die notwendige Topologie zu ermittelt. Als Clients sind verschiedene Akteure vorstellbar. Dies können SQL-Abfragen aus einem GIS heraus sein, oder ein Script, das in einer Web-Umgebung eingebunden ist. Denkbar wäre auch eine Erweiterung für einen gängigen OSM-Editor, um das Routingverhalten in den Daten zu untersuchen und gegebenenfalls Fehler aufzudecken.

## 4.1 Testgebiete

Für das Testen des Routinggraphen sind im Vorfeld Testgebiete definiert worden. Der erste Datensatz liegt im urbanen Raum in Stuttgart. Hier ist davon auszugehen, dass die Daten vollständiger sind als im ländlichen Raum. Des Weiteren wird das Programm in Biberach an der Riß, einer Kleinstadt in Oberschwaben sowie in Großschaufenhausen, einem kleinen Dorf im Kreis Biberach, getestet. Hier sind deutliche Unterschiede in der Quantität der Daten erkennbar, aber auch bezüglich des Taggings sind Qualitätsunterschiede zu erwarten (vgl. Anhang B).

In diesen Gebieten sind zwölf Testrouten definiert, von denen drei im Folgenden näher beschrieben und analysiert werden. Die erste Testroute verläuft durch den Stadtgarten vor der Hochschule. In den Ausgangsdaten sind alle befestigten Fußwege vorhanden. Anhand der Testroute wird geprüft, ob die Datenextraktion und das Routing entlang der Kanten in der Datenbank grundsätzlich funktioniert. Ein Zielpunkt ist nahe der Fußwege platziert, ein weiterer an der Friedrichstraße.

Das zweite Testgebiet erstreckt sich von der Mensa des Campus Stadtmitte über das Lindenmuseum in die Sattlerstraße (vgl. Abbildung 1). In dem Gebiet liegen zwei Kreuzungen. Auf der ersten größeren Kreuzung sind die Fußgängerüberwege in den Daten als Ways enthalten und müssen sinnvoll mit den konstruierten Bürgersteigen verbunden werden. Auf der nächsten, kleineren Kreuzung sind nur Nodes auf den KFZ-Straßen als Überwege getagged.

Außerdem ist das Problem der doppelten Fußwege in der Abbildung 5 deutlich zu sehen, dessen Lösungsansatz in Kapitel 5.1.4 beschrieben ist. Vor der Mensa verläuft in nördlicher Richtung ein Fußweg, der mit einem konstruierten Bürgersteig entlang der Straße konkurrieren würde. Hier muss angenommen werden, dass der spekulierte Bürgersteig vor Ort nicht existiert. Eine weitere Herausforderung stellt die Tatsache, dass für die Straßen, die sich auf dem Hegelplatz kreuzen, beide Fahrtrichtungen se-

parat gemappt sind. Jedoch fehlt eine Aussage in den Tags, auf welcher Seite der Fahrbahn ein Bürgersteig existiert. Sie sind lediglich als Einbahnstraße deklariert. Die dritte Testroute verläuft über den Marktplatz in Biberach. Interessant ist an dieser Stelle, wie sich das Routing auf dieser Fläche verhält. Einerseits ist die Umrandung des Platzes in den Daten enthalten, andererseits sind Fußwege darin kartiert. Insgesamt tauchen im urbaneren Raum aufgrund der Dichte der Daten andere Fragestellungen auf. Hinzu kommen Wege, die außerhalb der Ortschaften verlaufen. Ist es zum Beispiel möglich, von einem Dorf aus ein zweites über Feld- und Waldwege zu erreichen?

## 4.2 Attributierung in OpenStreetMap

Ein OSM-Datensatz besteht aus drei Hauptklassen: Nodes, Ways und Relations.<sup>3</sup> Die Nodes sind die einzigen Objekte, die geografische Informationen in Form von Länge und Breite besitzen. Ein Way kann sowohl Linien- als auch Flächenobjekte abbilden und besteht aus mehreren Nodes.

Relationen sind Gruppierungen von mehreren Nodes oder Ways. Beispielsweise können mehrere Straßenstücke zu einer Buslinie, mehrere Flusssegmente zum Rhein oder mehrere Grenzlinien zu einer Stadtgrenze zusammengefügt werden. Für das Routing haben Relationen in aller Regel keine Bedeutung. Daher werden sie in der vorliegenden Arbeit nicht berücksichtigt.

Die Definition der Objektart ist über das Tagging geregelt. Jedem Objekt können Tags in Form von Schlüssel-Wert-Paaren zugeordnet werden. Für Straßen wird das *highway*-Tag benutzt und der Wert gibt an, was für eine Art von Straße es ist. Zusatztags können spezielle, für Fußgänger relevante Informationen enthalten. Beispielsweise verfügt ein Fußweg über das Tagging *highway=pedestrian, surface=cobblestone*, welches beschreibt, dass der Straßenbelag aus Kopfsteinpflaster besteht. Die Tabellen in Anhang A geben einen Überblick über das für die Arbeit relevante Tagging. Die Klassifizierung der Straßen in *primary*, *secondary* und *tertiary* stellt für das Extraktionsverfahren eine große Herausforderung dar. Für ein weltweit gültiges Schema

---

<sup>3</sup>Eine Übersetzung der Klassennamen ist besonders für den Node schwierig, da die wörtliche Übersetzung Knoten nur in Ausnahmefällen richtig ist und der geometrisch behaftete Begriff des Punktes dem Bedeutungsinhalt eines Nodes nicht gerecht wird. Daher sollen die originalen Bezeichner *Node*, *Way*, *Relation* im Text verwendet werden.

werden die Straßen in OSM nach ihrer Bedeutung für ihren Ort klassifiziert. Hauptverkehrswege werden als *primary* bezeichnet und je nach Größe der Ortschaft werden weitere wichtige Verbindungsstraßen abgestuft in *secondary* und *tertiary*. Wenn in einem Land die Verkehrswege einer klaren Klassifizierung unterliegen, ist es einfacher zu beurteilen, wie die Straßen vor Ort aussehen. In Deutschland sind die Hauptverkehrswege in den großen Städten meist mehrspurig und nicht überquerbar, auf dem Land dagegen kann in aller Regel von ruhigen Straßen mit Bürgersteig ausgegangen werden, die sich auch ohne Ampel oder Zebrastreifen leicht überqueren lassen. Eine allgemeingültige Aussage zur Kartierung der Straßen weltweit kann jedoch nicht getroffen werden.

Die Reihenfolge der Nodes in einem Way spielt bei bestimmten Objektarten eine Rolle. Einbahnstraßen müssen in richtiger Reihenfolge kartiert sein, aber auch relative Ortsangaben wie *sidewalk=left* sind abhängig von der Richtung der Basisgeometrie. Dies spielt im Rahmen der Datenextraktion eine wichtige Rolle.

### 4.3 Attribute in der Datenextraktion

Grundsätzlich werden die eingehenden Daten nach zwei Kategorien sortiert. Jeder begehbarer Way wird als *PedestrianRoad* klassifiziert – Straßen, die vermutlich einen Bürgersteig besitzen, als *VehicleRoad*. *VehicleRoads* sind Straßen, die vorwiegend für motorisierte Fahrzeuge existieren. Außerdem werden Nodes gekennzeichnet, die innerhalb der Straße einen Überweg markieren. Zu *PedestrianRoads* gehören neben Fußwegen noch Spielstraßen, Feldwege und Treppen. In Fußgängerzonen sind meist linienförmige Wege eingezeichnet, die ebenfalls in den Graphen mit aufgenommen werden. Dies hat den Grund, dass sie von den anderen Fußwegen nur mit größerem Aufwand unterschieden werden können. Außerdem wird angenommen, dass sie oft günstige Verbindungswege auf den Flächen darstellen. Von den Kraftfahrzeugstraßen wird jede berücksichtigt, die nicht autobahnartig ist. Zudem wird nach dem *footway-*, *tunnel-* und *bridge*-Tag gefiltert. In Tunnels und auf Brücken wird davon ausgegangen, dass, falls ein Fußweg existiert, dieser separat gemapped ist.

Die beschriebene Klassifizierung dient ausschließlich der Extraktionsalgorithmitik innerhalb der Software. In der Datenbank werden die Liniengeometrien einerseits nach der Begehbarkeit attributiert. Dies geschieht in der Spalte *Type* mit Schlüsselworten wie *steps*, *dirtroad* und *sidewalk*. Andererseits wird die Originalattributierung aus

OSM in der Spalte *OSM-Type* gespeichert. Die datenbankseitige Attributierung dient der Personalisierung der Wegberechnung und ermöglicht eine sprachbasierte Wegbeschreibung.

Überwege sind entweder als linienförmige Objekte oder als Nodes auf den KFZ-Wegen gemappt. Die Art des Überweges ist im *crossing*-Tag beschrieben. Es wird unterschieden zwischen Fußgängerampel, Straßenmarkierungen, Verkehrsinsel und unkontrolliertem Überweg. Zudem kann im *crossing-ref*-Tag die Art des Überweges detaillierter angegeben werden (Tabelle 2). Ways, die einen Überweg abbilden, haben neben dem *highway*-Tag noch das Tagging *footway=crossing* (vgl. OSM Community, 2015).

## 4.4 Datenmodell

Die Ausgabetafelle *ways* (Tabelle 2) enthält alle Informationen, die für die folgenden Prozesse des Routings und der Navigation von Bedeutung sind. Einerseits sind das die Geometrien, andererseits die Sachinformationen, die für die Wegbeschreibung verwendet werden und die eine Personalisierung ermöglichen. In der Spalte *name* wird von den Direktextrakten der OSM-Name eingefügt und für erzeugte Geometrien der Name der Straße, entlang derer der Bürgersteig oder die Kreuzung konstruiert wurde. Die Spalte *type* enthält den Typ des Weges. Für die übernommenen Geometrien steht dort der Wert des *highway*-Tags. Sind die Geometrien dagegen konstruiert, ist der Typ entweder *sidewalk*, *osm-crossing*, *frequent-crossing* oder *risk-crossing*. Für die Bürgersteige werden zudem in der Spalte *osm-type* der Typ der Straße, entlang die der Bürgersteig konstruiert ist, hinterlegt. Für die *osm-crossings* wird dort die Überquerungsart gespeichert. Für alle restlichen Geometrien bleibt die Spalte *osm-type* leer. Damit Straßen unabhängig von den Überwegen in den Ausgangsdaten überquert werden können, werden die Bürgersteige in regelmäßigen Abständen miteinander verbunden. Diese Überquerungen werden als *frequent-crossing* bezeichnet.

**Tabelle 2:** prRouting-Tabelle *ways*

| Spalte   | Typ                   |
|----------|-----------------------|
| gid      | integer               |
| class_id | numeric(10,0)         |
| type     | character varying(20) |
| osm_type | character varying(14) |

**Tabelle 2:** prRouting-Tabelle *ways*

| Spalte   | Typ                       |
|----------|---------------------------|
| length   | numeric                   |
| name     | character varying(40)     |
| osm_id   | character varying(14)     |
| the_geom | geometry(MultiLineString) |
| source   | integer                   |
| target   | integer                   |

Eine Personalisierung kann durch zwei Arten erfolgen. Entweder werden bestimmte Wege, wie Treppen, aus dem Routinggraphen ausgeschlossen oder es werden bestimmte Wege bevorzugt. Die Personalisierung kann SQL-seitig durch eine geschachtelte Abfrage realisiert werden, indem nicht auf der Tabelle *ways*, sondern auf einer Unterabfrage geroutet wird. Gibt es dauerhafte Profile bestimmter Personalisierungen, können abgeleitete Tabellen erstellt werden. Dies reduziert die Laufzeit der Abfrage, geht jedoch zu Lasten der Flexibilität. Die Tabellen der Profile müssen jedes mal neu ermittelt werden, wenn sich im Routinggraphen etwas ändert. Um bestimmte Wege zu bevorzugen, ist es möglich die Distanz des Weges in der Spalte *length* für bestimmte Wegtypen rechnerisch zu manipulieren. Werden OSM-eigene Überwege bevorzugt, kann die Distanz für die *frequent-crossings* um ein Vielfaches erhöht werden.

Listing 4 und Listing5 zeigt Umsetzungsbeispiele zur Erstellung abgeleiteter Profiltabellen.

**Listing 4:** Umsetzung der Personalisierung: Treppen vermeiden

```
CREATE TABLE no_steps AS
SELECT * FROM ways WHERE type <> 'steps';
```

**Listing 5:** Umsetzung der Personalisierung: offizielle Überwege bevorzugen

```
CREATE TABLE save_crossing AS
SELECT gid , class_id , type , osm_type , length * 10 , name ,
       osm_id , the_geom , source , target
FROM ways
WHERE type = 'frequent-crossing' OR type = 'risk-crossing'
UNION
SELECT * FROM ways
WHERE type <> 'frequent-crossing'
AND type <> 'risk-crossing';
```

## 5 Implementierung

Dieses Kapitel befasst sich mit der Umsetzung des Konzeptes. Das Ergebnis ist das Konsolenprogramm PedRO. Implementiert ist die Software in C++, da die verwendete Osmium-Bibliothek für C++ verfügbar ist. Zunächst wird die konkrete Implementierung von vier wesentlichen Algorithmen beschrieben, um im Anschluss die Laufzeit und der, damit zusammenhängende, Speicherbedarf der Software zu diskutieren. Einen Überblick über PedRO liefert das UML-Diagramm im Anhang D.

### 5.1 Extraktionsalgorithmen

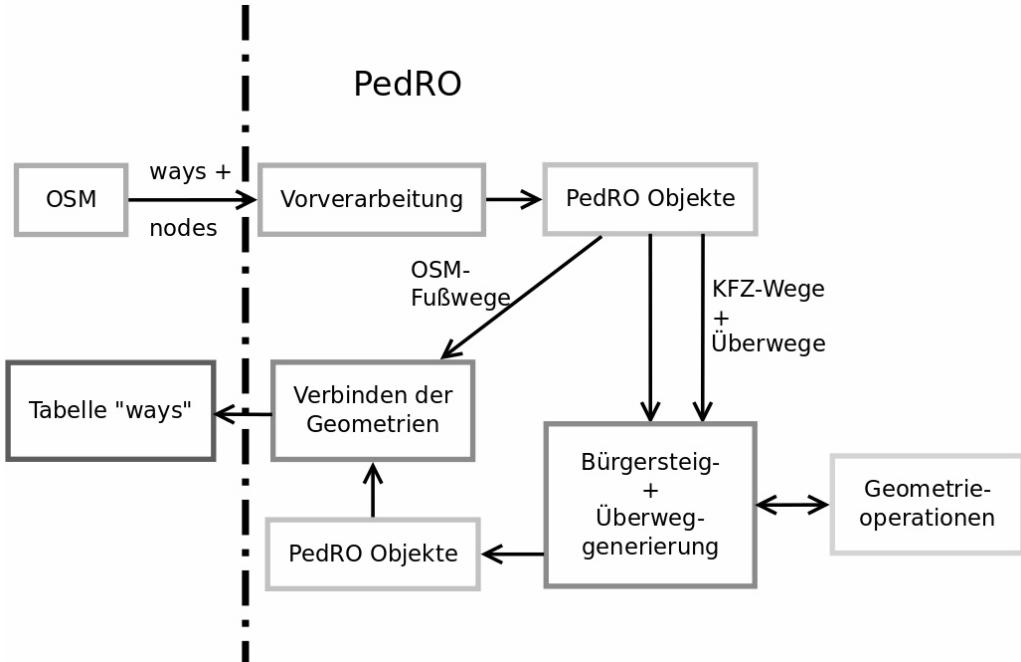
Abbildung 3 zeigt das Funktionsprinzip von PedRO. Die OSM-Daten werden zweimal eingelesen. Im ersten Durchlauf werden die Geometrien der Ways vorbereitet, um aus ihnen im zweiten Durchlauf die PedRO-eigenen Objekte zu erzeugen. Alle Straßengeometrien sind an den Kreuzungen und Abbiegungen zerlegt, damit aus ihnen eine verwertbare Topologie erzeugt werden kann. Die KFZ-Wege und die Überwege werden für die Generierung der Bürgersteiggeometrien und der Überweggeometrien verwendet. Hierfür sind verschiedene Geometrieoperationen notwendig, die teilweise aus der GEOS-Bibliothek stammen, teilweise selbst erstellt sind.

Aus den erzeugten Geometrien und den aus den OSM-Daten übernommenen Attributen werden weitere Objekte erzeugt. Um alle Geometrien miteinander zu verbinden, werden die erzeugten Geometrien mit denen der vorhandenen Fußwege verschnitten. Der letzte Schritt ist der Datenbankexport in die Tabelle *ways*, auf der pgRouting das Routing rechnen kann.

Im Folgenden wird auf vier wesentliche Algorithmen näher eingegangen.

#### 5.1.1 Direktextrakte

Alle Geometrien, die in OSM als Fußwege getagged sind, werden direkt aus den Daten übernommen. Osmium stellt eigene Geometrieerzeuger (GeometryFactory), die OGR- und GEOS-Geometrien erzeugen können, zur Verfügung. Außerdem können Tags, wie der Name des Weges, OSM-ID und weitere Zusatzinformationen, bequem



**Abbildung 3:** Übersicht über die Algorithmik innerhalb der Extraktionssoftware PedRO.

aus den Daten gelesen werden. Das einzige Hindernis, das routingfähigen Geometrien im Weg steht, ist die Tatsache, dass die Liniensegmente an den Schnittpunkten mit anderen Wegen nicht unterbrochen sind. Ein Routingalgorithmus muss von jedem Knoten die nächsten Nachbarnknoten kennen. Zwar erstellt pgRouting mit *pgr\_createTopology* zwei eindeutige Knoten am Anfang und Ende jeder Liniengeometrie, jedoch werden Schnittpunkte zweier Geometrien nicht ohne weiteres erkannt. In dem Verarbeitungsprozess werden die Knoten mit einer ID versehen und in jedem Eintrag der Tabelle *ways* in den Spalten *source* und *target* vermerkt. Aus diesem Grund müssen die Linien an ihren Schnittpunkten mit anderen Geometrien unterteilt werden. Während des Leseprozesses der OSM Daten können diese Wege ohne geometrische Verschneidungsoperation zerlegt werden. Das liegt daran, dass sich die Linien in Nodes schneiden.

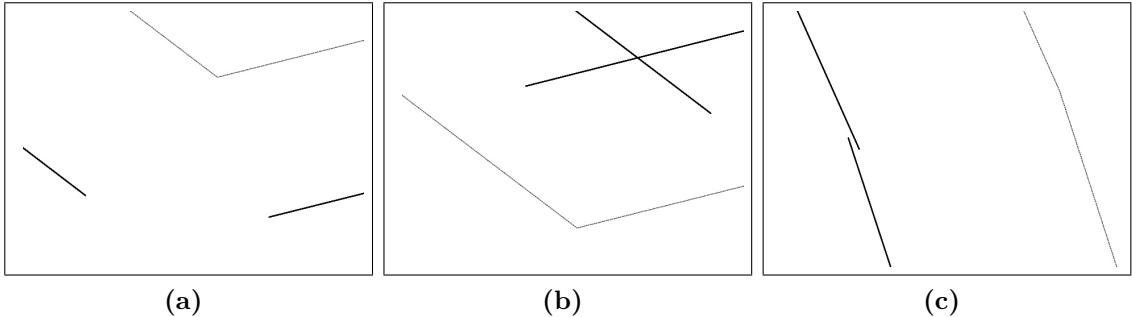
Die Zerlegung erfolgt in zwei Schritten. Zuerst werden die Nodes aller (Fußgänger-)Ways abgefangen. In einem Knotenverzeichnis werden jeder ID eines Nodes die zugehörigen IDs der Ways zugeordnet, in denen der Node vorkommt. Jeder NodeID werden die benachbarten IDs in einem Verzeichnis zugeordnet. Kreuzungspunkte sind diejenigen Nodes, denen mehr als zwei Nachbarnodes zugeordnet sind. In einem Zwischenschritt wird über das Verzeichnis iteriert. Für jeden Weg, der Kreuzungs-

punkte enthält, werden die IDs der Nodes zugewiesen. Nun können während eines zweiten Lesevorgangs die Geometrien zerlegt werden. Alle Ways, die nicht in der Map erscheinen, werden direkt in die Datenbank importiert. Die anderen werden an den Schnittpunkten zerschnitten; die Teilstücke bekommen jeweils dieselben Attribute.

### 5.1.2 Bürgersteige

Während die OSM-Daten eingelesen werden, wird für jeden Way vom Typ *VehicleRoad* ein Ereignis aufgerufen. Dort wird zunächst ein *VehicleRoad*-Objekt erzeugt, in dem die Geometrie und die Sachdaten gespeichert werden. Zu den Sachdaten zählt der Name der Straße, der Straßentyp, die Anzahl der Spuren und die Informationen über die Bürgersteige. Der Umfang dieser Sachdaten ist abhängig von der Qualität der Daten. Zur Erzeugung eines Verzeichnisses aller Nodes, wird über die einzelnen Nodes jedes Linienobjektes iteriert. In dem Verzeichnis wird jeder Node-ID die Nachbar-IDs und eine Referenz auf das zugehörige *VehicleRoad*-Objekt zugeordnet. Damit die folgenden Generierungsschritte fehlerfrei funktionieren, werden die Nachbarnodes anhand der Orientierung (Winkel zur Nordrichtung) im Uhrzeigersinn geordnet. Im Anschluss an den Leseprozess wird zur Erzeugung der Bürgersteiggeometrien über das Verzeichnis iteriert. Ausgehend von den Nodes werden parallele Liniensegmente zu den Nachbarnodes erzeugt. Damit die Geometrien nur einmal erzeugt werden, wird die Verbindung anhand beider IDs als abgeschlossen markiert. Dies ist außerdem für den nächsten Schritt, dem Verbinden der Segmente, wichtig. Die Segmente zeigen in eine andere Richtung, wenn sie in einem früheren Schritt erzeugt wurden.

Mit der Generierung der parallelen Segmente entsteht auf der konvexen Seite der Kurve eine Lücke zwischen den Segmenten (Abb. 4a). Dagegen schneiden sich die Segmente auf der konkaven Seite entweder (Abb. 4b) oder sie verfehlten sich sehr knapp (Abb. 4c). Zunächst wird der Winkel zwischen den Segmenten bestimmt. Ist dieser größer als  $180^\circ$ , wird einem Liniensegment die Verbindung zur Nachbarkoordinate des nächsten Segmentes hinzugefügt. Bei einem Winkel kleiner-gleich  $180^\circ$  wird zunächst geprüft, ob sich die Segmente schneiden. Gibt es einen Schnittpunkt, werden die Startkoordinaten der Segmente durch den Schnittpunkt ersetzt. Ansonsten wird der Mittelwert aus den beiden Startkoordinaten gebildet und als Verbindungs-



**Abbildung 4:** Fallunterscheidung für das Verbinden der parallelen Liniensegmente. (a) Lücke auf der konvexen Seite der Kurve. (b) Segmente schneiden sich auf der konkaven Seite. (c) Segmente schneiden sich auf der konkaven Seite nicht.

punkt verwendet. Hat ein Node lediglich einen Nachbarn, werden die Enden beider Bürgersteige miteinander verbunden. Dadurch reduziert sich die Zahl offener Enden im Routinggraphen. Es wird davon ausgegangen, dass es in der Regel möglich ist, eine Sackgasse an ihrem Ende zu überqueren.

### 5.1.3 Überwege

Fußgängerüberwege können als fertige Fußwege in den Daten enthalten sein. Dann werden sie, wie in Kapitel Direktextrakte beschrieben, in die Datenbank übernommen. In den meisten Fällen ist dagegen ein Node innerhalb der überquerenden Straße als Überweg markiert. Um diese Nodes zu einem späteren Zeitpunkt identifizieren zu können, ist ein vorbereitender Einleseprozess notwendig. Jeder Überweg-Node wird zusammen mit der Überquerungsart in ein Verzeichnis gespeichert. Bei der Erzeugung der Bürgersteige wird für jeden Node geprüft, ob er einen Überweg darstellt. Ist dies der Fall, wird ein Überweg des entsprechenden Typs erzeugt.

Zusätzlich zu den offiziellen Überwegen werden regelmäßige Straßenüberquerungen in den Graphen mit aufgenommen. Diesen liegt die Annahme zugrunde, dass jede Straße unabhängig von einem Fußgängerüberweg überquert werden kann. In ruhigen Gegenden ist dies unumgänglich, um beispielsweise eine abzweigende Straße zu überqueren. Diese Überwege werden mit einer größeren Distanz versehen, damit der Routingalgorithmus sich im Zweifelsfall für den offiziellen Überweg entscheidet. Zudem werden die Überwege für größere Straßen mit mehreren Spuren als *risk-crossing*

markiert, damit diese Überquerungen im Rahmen der Personalisierung herausgefiltert werden können.

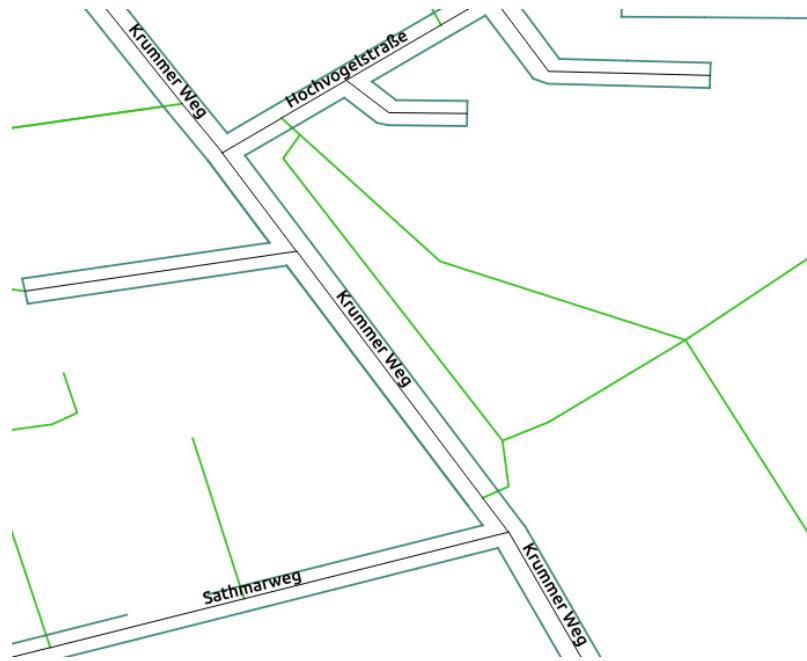
#### 5.1.4 Kontrastverfahren

Abbildung 5 veranschaulicht, dass die erzeugten Bürgersteiggeometrien in Konkurrenz zu den vorhandenen Fußwegen stehen können. In der Mitte der Abbildung am Krummer Weg verläuft ein Fußgängerweg parallel zum Bürgersteig. Hier wird davon ausgegangen, dass der Bürgersteig vor Ort nicht existiert. Zwei Situationen sind in solchen Fällen am wahrscheinlichsten. Einerseits kommt es vor, dass Bürgersteige in OSM separat kartiert sind. Andererseits gibt es häufig die Situation, dass neben einer Straße, meist getrennt durch einen Grünstreifen, ein Fußweg vorhanden ist. Dort ist ebenfalls anzunehmen, dass an der Straße kein zusätzlicher Bürgersteig vorhanden ist.

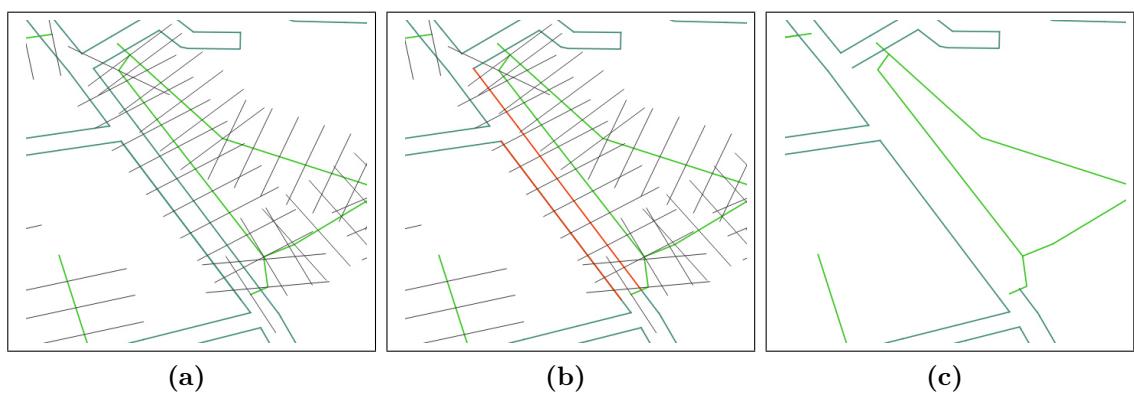
Um solche Situationen aufzudecken, wird eine Kontrastanalyse eingesetzt, die die erzeugten Geometrien mit den vorhandenen Wegen vergleicht. Verwendet wird ein Ansatz, den Mapbox zum Korrigieren der OSM-Daten aufzeigt (Channell, 2015). Jedes Liniensegment der Fußgängerwege aus den Ausgangsdaten wird in regelmäßigen Abständen zerlegt. An diesen Unterpunkten werden Orthogonale erstellt (Abb. 6a) und in einem R-Baum gespeichert. Ein R-Baum ist ähnlich dem B-Baum und eignet sich zur balancierten Speicherung räumlicher Daten (vgl. Wikipedia, 2016c). Er ist in der GEOS-Bibliothek implementiert.

Im R-Baum wird die Bounding Box der Orthogonale sowie ein Pointer auf das Geometrieobjekt abgelegt. Eine Anfrage auf den Baum mit einem Bürgersteigsegment gibt einen Vektor aller Orthogonale zurück, die sich möglicherweise mit dem Segment schneiden. Dies erfolgt durch den Vergleich der Begrenzungsrahmen, die bei Liniensegmenten jedoch keinen Speichervorteil bieten. Dennoch ist die Datenstruktur des Baumes sehr vorteilhaft, da die Objekte nach ihrem räumlichen Index sortiert sind. Mit deutlich aufwendigeren Verschneidungsfunktionen für jedes Bürgersteigstück und den benachbarten Orthogonale kann die Zahl der Schnittpunkte gezählt werden. Der Vergleich dieser Anzahl mit der Länge des Liniensegmentes liefert ein Verhältnis. Ein Faktor legt nun fest, ob der entsprechende Bürgersteig eine große Ähnlichkeit aufweist (Abb. 6b).

Die Stellschrauben dieses Algorithmus sind die Distanz der Untersegmente zuein-



**Abbildung 5:** Problem konkurrierender Geometrien. OSM Fußweg (grün), generierter Bürgersteig (blau), KFZ-Straße (schwarz)



**Abbildung 6:** Kontrastverfahren. Bürgersteige (blau), Fußwege (grün). (a) Orthogonale nach Segmentierung der Fußwege (schwarz). (b) Verschneidung der Orthogonalen mit den Bürgersteigen (rot). (c) Ergebnis: Bürgersteig mit kleinster Distanz wird ausgeschlossen.

ander (Segmentierungsfaktor), die Länge der Orthogonalen (Bürgersteigabstand) und der Faktor, der bestimmt, wie viele Orthogonalen pro Segmentierungsfaktor einen Bürgersteig mindestens schneiden müssen (Kontrastfaktor). Erstere Distanz darf nicht zu groß gewählt werden, damit kleinere Wegstücke nicht zerlegt werden. Am Startpunkt jedes Liniensegmentes wird eine Orthogonale gebildet, so dass mindestens eine Orthogonale besteht. Bei engen bzw. detailreich kartierten Kurven führt dies zu Verfälschungen. Andererseits wirkt sich der Segmentierungsfaktor quadratisch antiproportional auf den Aufwand des Algorithmus aus und sollte daher nicht zu klein gewählt werden. Die Länge der Orthogonalen sollte einem maximalen Erfahrungswert entsprechen. Tests ergaben für eine Entfernung von zehn Meter akzeptable Ergebnisse. Der Kontrastfaktor muss unter 1.0 gewählt werden, da durch eine abweichende Neigung beider Linienstücke zueinander weniger Schnittpunkte entstehen. Channell hat hierfür einen Faktor von 0.7 empfohlen, wobei sein Vorgehen leicht von dem Beschriebenen abweicht.

Um zu vermeiden, dass die gegenüberliegende Straßenseite von der Ausdünnung betroffen ist, wird der Algorithmus zweimal durchlaufen. Im ersten Durchlauf wird für jede Orthogonale die Distanz zu dem Schnittpunkt gespeichert, der am nächsten liegt. So können für den zweiten Durchlauf nur diejenigen Bürgersteige aussortiert werden, die als nächstes zum Fußweg liegen (Abb. 6c).

## 5.2 Komplexitätsanalyse

Der Speicherbedarf und somit auch die Laufzeit von PedRO ist von mehreren Faktoren abhängig. Allgemein sind sie abhängig von der Funktionsdichte der Eingabedatei. Einerseits hat die Anzahl ( $k$ ) der Linienstücke der Verkehrswege einen Einfluss auf die Komplexität, andererseits spielt die Gesamtlänge ( $l$ ) des Straßennetzes eine wichtige Rolle.

Am Anfang werden alle Nodes des Datensatzes in den Speicher geladen, damit zum späteren Zeitpunkt die Koordinaten der Nodes zu Verfügung stehen. Benötigt werden jedoch nur die Nodes der Verkehrswege. Für die KFZ-Wege und für die Fußwege werden Knotenverzeichnisse erstellt. Die Größe dieser Verzeichnisse ist abhängig von  $k$ . Die Anzahl der Fußwegobjekte entspricht der Anzahl der OSM-eigenen Fußwege, zerlegt an deren Knotenpunkte. Da die Bürgersteige und Überquerungen in regelmäßigen Abständen erzeugt werden, ist die Menge der erzeugten Objekte

abhängig von der Länge der KFZ-Wege und dem Grad der Frequentierung. Für die Unterteilung wurde ein Abstand von 50 Metern gewählt. Also ist die Menge der Bürgersteige als auch Überwege:

$$\text{Anzahl Bürgersteige} = \frac{l(\text{VehicleRoads})}{50m}$$

$$\text{Anzahl (regelmäßiger) Überquerungen} = \frac{l(\text{VehicleRoads})}{50m}$$

Ähnliches gilt für das Kontrastverfahren, bei dem die *PedestrianRoads* alle 10m zerlegt und orthogonale Liniensegmente erzeugt werden:

$$\text{Anzahl der Orthogonalen} = \frac{l(\text{PedestrianRoads})}{10m}$$

An mehreren Stellen im Programmverlauf wird durch Verzeichnisse und Listen iteriert. Zunächst wird durch die Knotenverzeichnisse der Fußwege iteriert, um die PedRO-Objekte der Fußwege zu erzeugen. k entspricht hier der Anzahl der Wege zwischen den Knoten. Zur Generierung der Bürgersteige wird durch das Knotenverzeichnis der KFZ-Wege iteriert. Beide Verzeichnisse sind verhältnismäßig klein, da die regelmäßigen Unterteilungen noch nicht stattgefunden haben. Als nächstes wird das Kontrastverfahren angewendet. Hierbei wird über die Bürgersteig-Objekte iteriert und mit den Orthogonalen verglichen. Die Orthogonalen befinden sich in einem R-Baum, wodurch sich die relativ große Menge der Geometrien logarithmisch auf die Laufzeit auswirkt. Dennoch müssen zur Erstellung des Baumes die einzelnen Objekte erzeugt werden, was sich auf die Laufzeit auswirkt.

Der letzte Arbeitsschritt, bevor die Daten in die Datenbank geschrieben werden, ist die Verschneidung aller Geometrien miteinander. Um Laufzeit zu sparen, werden die zahlenmäßig größeren Mengen der Bürgersteige und Überquerungen in zwei weiteren R-Bäumen abgelegt. Iteriert wird hierbei über die *PederstrianRoads*.

$$\text{Anzahl Fußwege} = \text{Anzahl Knotenpunkte des Fußwegenetzes}$$

Daraus ergibt sich, dass die Prozesse mit der höchsten Komplexität diejenigen sind, die abhängig von der Länge des Straßennetzes sind. In der Sprache der O-Notation (Lang, 2012, S. 368-371) ergibt sich:

$$T(n) = O(n)$$

Wobei  $n$  die Anzahl der 50m-Straßenstücke der KFZ-Wege plus die Anzahl der 10m-Straßenstücke der Fußwege ist. Daraus ergibt sich:

$$T(n) = O\left(\frac{l}{\text{Frequentierung}}\right)$$

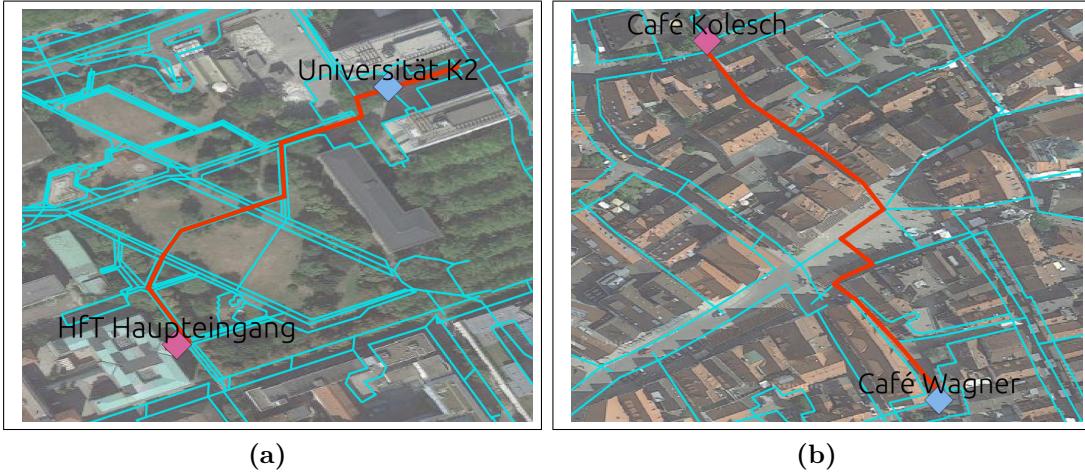
## 6 Evaluation

Grundsätzlich lässt sich sagen, dass die Extraktion soweit funktioniert, dass das Routing für neun von zwölf Testrouten plausible Ergebnisse liefert. Der Übergang von den konstruierten Geometrien auf Fußwege funktioniert dort, wo sich beide Geometrien schneiden. Bürgersteige werden in einem definierten Abstand zur Straße konstruiert und sind miteinander verbunden. Kreuzungspunkte auf der Straße werden als solche erkannt, und ein Überweg wird erzeugt. Diese sind zwar nur teilweise rechtwinklig zum Bürgersteig, das spielt für die Entfernungsmessung und die Wegfindung jedoch kaum eine Rolle. Das Kontrastverfahren erfüllt teilweise seinen Zweck, führt andererseits jedoch auch zu Fehlern im Routinggraphen. Spitze Straßenwinkel und zu detaillierte Kurvenkartierungen können ungewollte Nebeneffekte verursachen. An einigen Stellen entstehen Inseln im Routinggraphen und können darin zu Fehlverhalten führen. Im Folgenden soll vertiefend auf diese Schwierigkeiten eingegangen werden.

### 6.1 Erfolgreiche Testrouten

Die erste Testroute durch den Stadtgarten in Stuttgart (Abbildung 7a) verläuft ausschließlich über die Fußwege der Ausgangsdaten. Aus dem Ergebnis lässt sich schließen, dass der Routinggraph erfolgreich in die PostgreSQL-Tabelle exportiert wurde und pgRouting den Dijkstra-Algorithmus rechnen kann. Die zweite Route in Biberach verläuft über den Marktplatz (Abbildung 7b). Hier ist zu sehen, dass der Platz entlang den kartierten Fußwege überquert wird. Das Ergebnis ist zufriedenstellend, die Überquerung des Platzes ließe sich jedoch durch ein Flächenrouting verbessern. Die letzte Route führt vom Königsbau über den Schillerplatz zum Marktplatz in Stuttgart (Abbildung 8a). Im Vergleich zu der Route in Biberach ist hier der Routinggraph etwas dichter. Das Routing verläuft über den Schillerplatz relativ gerade. Ebenfalls ist zu erkennen, dass ein Gebäude durchquert wird. Dies ist

der Zugang zum Schillerplatz durch einen Torbogen. Die Testroute 8b verläuft von

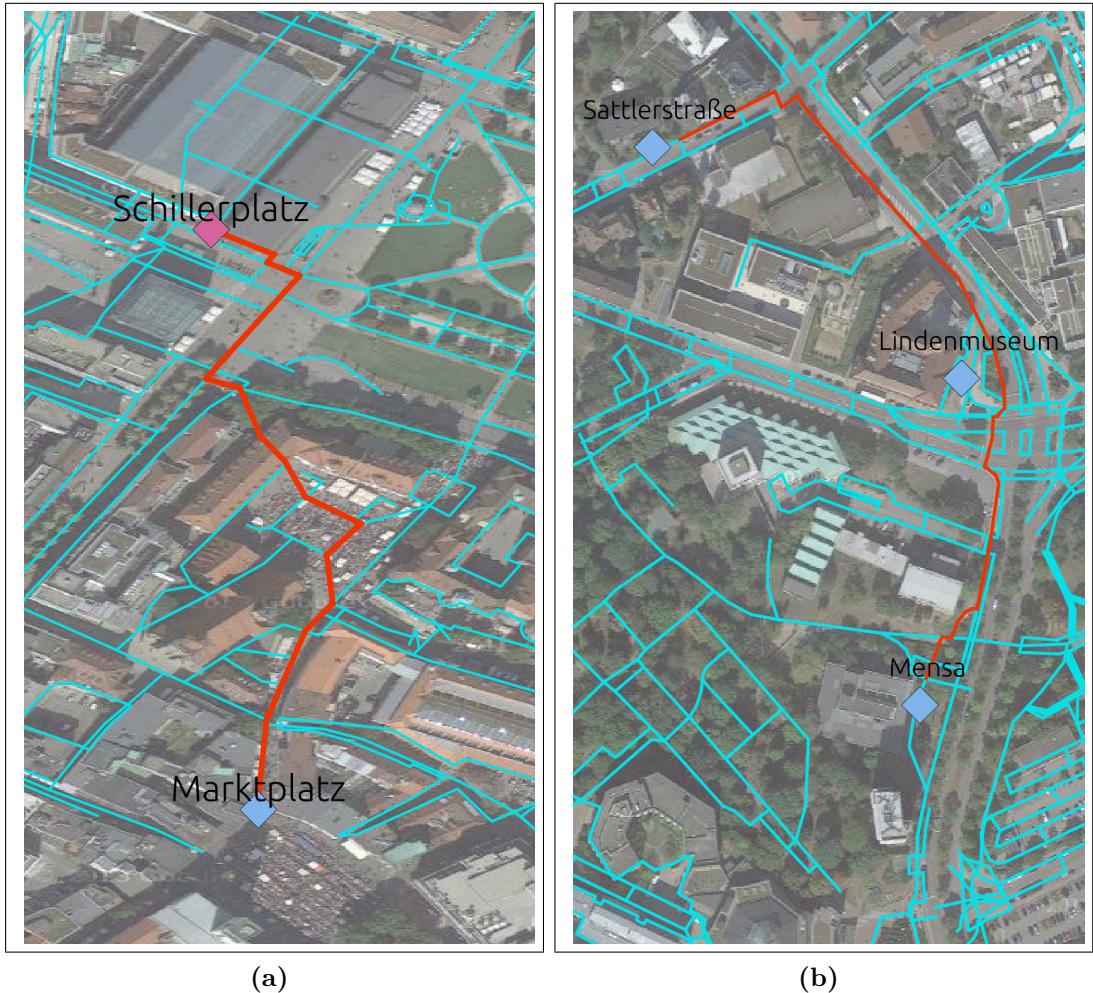


**Abbildung 7:** Testrouten: Route (rot), Routingtopographie (türkis). (a) Testroute vom Haupteingang der HfT durch den Stadtgarten zur Cafeteria der Uni Stuttgart. (b) Testroute über den Marktplatz Biberach. (Satellitenbilder: Google Maps)

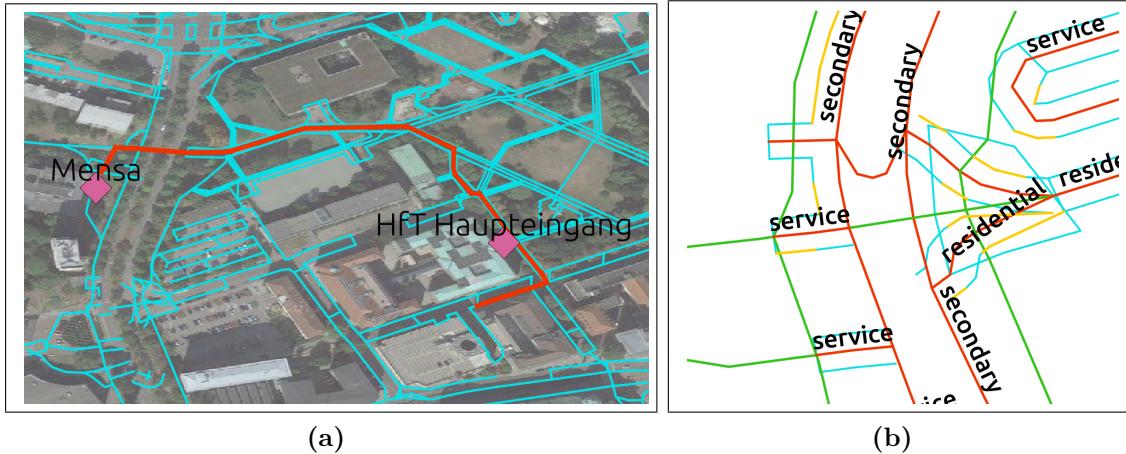
der Mensa Stadtmitte in Stuttgart am Lindenmuseum vorbei zur Sattlerstraße. An dieser Testroute wird deutlich, dass der Übergang von den OSM-eigenen Fußwegen zu den konstruierten Bürgersteigen funktioniert. Im Vergleich zur Route, die OpenRouteService anbietet, ist der Weg kürzer. Es werden nur zwei Straßen überquert, im Gegensatz zu drei Überquerungen. Die letzte Straßenüberquerung verläuft über einen ampelgesteuerten, offiziellen Überweg (vgl. Anhang B, Abbildung 1 (c)).

## 6.2 Spitze Winkel

Ab einem bestimmten Winkel kommt es bei der Bürgersteiggenerierung zu fehlerhaften Geometrien. Gleiches passiert bei sehr kurzen Straßensegmenten und verhältnismäßig großer Krümmung. Die Geometrien sind meist selbstschneidend. Dies führt zwar hinsichtlich des Routings nur zu unwesentlichen Fehlern, da dadurch lediglich die Länge des Weges vergrößert wird. Dennoch sind sie im Routinggraphen unerwünscht. Bei einer Darstellung der Topographie entstehen dadurch missverständliche Darstellungen. Eine Lösung dieses Problems wäre durch eine Analyse und Vereinfachung der Geometrien vor der Konstruktion möglich.



**Abbildung 8:** Testrouten: Route (rot), Routingtopographie (türkis). (a) Testroute vom Schillerplatz in Stuttgart zum Marktplatz. (b) Testroute von der Mensa zur Sattlerstraße. (Satellitenbilder: Google Maps)



**Abbildung 9:** Route vom Haupteingang HfT zur Mensa. (a) Darstellung der ermittelten Route (rot), Routinggraph (türkis). (b) Problembetrachtung an der Kreuzung: Routinggraph (türkis), gefilterte Wege durch das Kontrastverfahren (gelb), original Fußwege (grün), original KFZ-Wege (rot), Beschriftung: Wert des *highway*-Tags. (Satellitenbilder: Google Maps)

### 6.3 Fehlerhafte Überwege

Die Route vom Haupteingang der HfT Stuttgart zur Mensa (Abbildung 9) verdeutlicht, dass an einigen Kreuzungen die Synthese der Geometrien nicht funktioniert. Zu erwarten wäre eine Route über die Kreuzung links unten in Abbildung 9a. Da aber an der Kreuzung keine sinnvollen Geometrien entstehen, wird eine längere Route in die andere Richtung vorgeschlagen. Wird die Kreuzung näher betrachtet (Abbildung 9a), können Ursachen für dieses Fehlverhalten gefunden werden. Auf der Kreuzung befinden sich eine Fußgängerampel und ein Zebrastreifen. Zudem zeigt sich, dass das Kontrastverfahren einige Geometrien gelöscht hat (gelb). Das führt dazu, dass die generierten Überwege keinen Anschluss an das restliche Wegenetz besitzen. Außerdem wurden hier ein paar Wege mit *highway=service* kartiert. Dies sind Zufahrtswege, die im Algorithmus nicht berücksichtigt werden. Ein Fußweg geht über in einen solchen Zufahrtsweg, und daher verliert sich der Routinggraph an dieser Stelle. Dieser Fehler kann sowohl aufgrund Unstimmigkeiten in den OSM-Daten entstehen, als auch durch fehlerhaftes Verhalten in der Topographiesynthese. Die Analyse einer solchen Situation bedarf meist viel Zeit, und es entstehen dabei nicht immer eindeutige Ergebnisse, die auf andere Situationen übertragbar sind.

## 7 Diskussion

### 7.1 Beurteilung Kontrastverfahren

Das Kontrastverfahren ist in gewisser Weise eigensinnig. Dahinter steht ein relativ komplexer Algorithmus, der gewisse Problematiken mit sich führt. Der Algorithmus erzeugt eine gewisse Menge an falsch-positiven Wegstücken, die nicht gelöscht werden dürfen. Er kann nicht für jede Situation kalibriert werden, so dass die Ergebnisse mit der vor Ort existierenden Wirklichkeit übereinstimmen. Wird die Software, wie unten beschrieben, von OSM-Mappern eingesetzt, sollte das Verhalten der Software nachvollziehbar sein. Eine Möglichkeit, die Nachvollziehbarkeit zu erhöhen, ist es, die detektierten falschen Bürgersteige in eine Ausgabetafel zu schreiben. Dadurch kann das Verhalten des Algorithmus beobachtet werden. Vorstellbar ist auch, dem Nutzer direkt Zugang zu den Parametern zu gewähren.

### 7.2 Krümmungsanalyse

Bisher unterliegt dem Algorithmus die Annahme, dass jede Straße überall überquerbar ist. Eine Überquerung ist jedoch dann gefährlich, wenn keine Sicht auf die Straße in einer der beiden Richtungen besteht, und dadurch Gefahren nicht rechtzeitig erkannt werden können. An kurvenreichen Straßen ist die Sicht in der Regel eingeschränkt, daher wäre es sinnvoll, in der Nähe und in der Kurve selbst auf regelmäßige Überquerungen zu verzichten. Mit einer Krümmungsanalyse könnten diese Stellen ermittelt werden. Die Krümmung berechnet sich aus dem Winkel zweier Liniensegmente zur Abweichung des Winkels zu 180 Grad. Je kleiner dieses Verhältnis ist, desto größer ist die Krümmung im Übergang zum nächsten Straßenstück. Mithilfe eines Suchfensters könnte der Mittelpunkt einer Kurve ermittelt werden, so dass in der Umgebung der Kurve Straßen nicht überquert werden.

### 7.3 Fehler in den Ausgangsdaten

Die Natur der Datenerhebung in OSM birgt, neben dem großen Potential, weltweit umfangreiche Karteninformationen hervorzubringen, eine gewisse Fehleranfälligkeit. Jeder kann den Datensatz nahezu beliebig editieren, daher tauchen verschiedenartige

Fehler auf. Fehler in den Geometrien können innerhalb von PedRO zu Programmabsturz führen, wenn zum Beispiel zwei aufeinander folgende Nodes über die gleiche Koordinate verfügen, oder wenn ein Way nur aus einem Node besteht. Fehler dieser Art lassen sich relativ einfach erkennen. Die Software kann ein Ausnahmeeignis erzeugen und den fehlerhaften Way ausgeben. Anhand solcher Mechanismen können Fehler aufgedeckt und in einem weiteren Schritt korrigiert werden.

Mit größerem Aufwand verbunden ist die Detektion von Tagging-Fehlern. Damit sind Unstimmigkeiten hinsichtlich der getroffenen Vereinbarungen über die Attributierung gemeint. Ist eine Straße als *highway=dirtroad* beschrieben, einer Beschreibung, die nicht vereinbart ist, wird die Straße im Extraktionsalgorithmus nicht beachtet. Ein ähnliches Verhalten entsteht bei Abbildungsfehlern, also Fehlern, die die Wirklichkeit vor Ort falsch abbilden. Wird eine Straße mit *sidewalk=left* beschrieben, die Reihenfolge der Nodes aber nicht beachtet, entsteht im Routinggraphen ein Bürgersteig an der falschen Straßenseite. Die entwickelte Software kann, wird sie in den Kartierungsprozess eingebunden, den Mapper dabei unterstützen, Daten in den OSM-Datensatz einzupflegen. Eine Voraussetzung dafür ist, dass die Extraktion weitgehend fehlerfrei funktioniert. Des Weiteren muss die Entstehung der neuen Geometrien für den Anwender nachvollziehbar sein, so dass eine Änderung der Eingabedaten zu erwartenden Ergebnissen in den Ausgabedaten führt.

Denkbar wäre eine Implementierung in JOSM, einem weit verbreitenden Editor für OSM-Daten (JOSM Community, 2005). Der Sinn einer solchen Implementierung wäre es, eine Prüfinstanz zu haben, die das Fußgängertagging überprüft. Ebenso sinnvoll wäre in diesem Zusammenhang eine Lösung für mobile Endgeräte. So könnte einerseits das Routing, bzw. der Routinggraph und dessen Erzeugung, überprüft werden, andererseits fallen im Feld Fehler in den Kartendaten auf.

## 7.4 Fazit

Die Arbeit zeigt, dass es grundsätzlich möglich ist, Fußweggeometrien aus den Openstreetmap Daten zu konstruieren. Noch treten an vielen Stellen Fehler auf, die nicht ohne weiteres behoben werden können. Um die Software in der Praxis nutzen zu können, müssen noch einige Sonderfälle untersucht werden. Lösungen für die meisten Probleme sind umsetzbar, jedoch oft nicht trivial und zeitaufwändig in ihrer Implementierung.

An vielen Stellen muss weiter ins Detail gegangen werden. Nützlich ist eine dif-

ferenziertere Klassifizierung der einzelnen Geometrien. So können die erzeugten Überwege hinsichtlich ihrer Nähe zur nächsten Kreuzung klassifiziert werden. Hierfür ist in pgRouting die Verwendung einer Klassen-ID vorgesehen, die bisher noch nicht berücksichtigt wird.

Die Software muss zudem umfassender getestet werden. Möglich wäre eine einbeziehung der OSM-Community für das Auffinden von Fehlern. Erfahrene Mapper können dabei mitdiskutieren, wie das Mapping üblich ist und aufdecken, ob das Fehlverhalten im Routing an der Extraktionssoftware oder an der Karte liegt.

# Literatur

- Bauer, C., Almer, A., Ladstätter, S., and Luley, M. (2014). Optimierte Wegefindung für Fußgänger basierend auf vorhandenen Open Street Map-Daten. *Angewandte Geoinformatik*, pages 408–413.
- Bernhard, R., Krampe, S., and Kollarits, S. (2012). *Vielmobil-Mobilitätslotse für die Region Frankfurt RheinMain*. na.
- Bohner-Degrell, C. and Köhler, A. (2014). Die Senioren haben die Wahl: Berücksichtigung individueller Nutzerpräferenzen bei der Gestaltung nahtloser Mobilitätsketten im Vorhaben namo. *Wohnen-Pflege-Teilhabe-„Besser leben durch Technik“*.
- BRouter Community (2013). BRouter. <http://brouter.de/brouter-web/>; zuletzt besucht: 29. 10.2015.
- Channell, T. (2015). Improving sidewalks globally in OpenStreetMap. <https://www.mapbox.com/blog/mapping-sidewalks/>; zuletzt besucht: 14.12.2015.
- Erdmann, J. (2013). Multimodalität und Nachfragegenerierung mit SUMO. In *Workshop Mobilitätssimulationen im Ereigniskontext*.
- GDAL, P. (2015). OGR API. [http://www.gdal.org/ogr\\_\\_api\\_8h.html](http://www.gdal.org/ogr__api_8h.html); zuletzt besucht: 24. 11.2015.
- Geue, A., Löchte, N., Balzer, D., Bargen, T. v., Helms, D., Howe, J., Lambacher, O., Radike, N., Retzlaff, J., and Szarvas, I. (2014). Entwicklung und Erprobung eines Konzepts zur Unterstützung der selbstständigen Mobilität älterer Menschen. In *VDE-Kongress 2014*. VDE VERLAG GmbH.
- Goetz, M. (2012). Using crowdsourced indoor geodata for the creation of a three-dimensional indoor routing web application. *Future Internet*, 4(2):575–591.
- GraphHopper Community (2012). GraphHopper. <https://graphhopper.com/maps/>; zuletzt besucht: 29. 10.2015.
- JOSM Community (2005). JOSM. <https://josm.openstreetmap.de/>; zuletzt besucht: 28.02.2016.

Kuntzsch, C. (2015). Konzeption und Implementierung eines multimodalen Campusroutenplaners am Beispiel der Universität Potsdam—Möglichkeiten und Grenzen der Nutzung von Open Source Software und freien Daten.

Lang, H. W. (2012). *Algorithmen in Java*. Oldenbourh Verlag.

Luxen, D. and Vetter, C. (2011). Real-time routing with OpenStreetMap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 513–516. ACM.

Mc Donough el al. (2000). METHOD AND SYSTEM FOR MAP DISPLAY IN A NAVIGATION APPLICATION.

OGC (2015). Location service (openls). <http://www.opengeospatial.org/standards/ols>; zuletzt besucht: 28. 10.2015.

OLU Community (2016). Open level up. <http://github.pavie.info/openlevelup/>; zuletzt besucht: 17.02.2016.

ORS Community (2008). OpenRouteService. <http://openrouteservice.org/>; zuletzt besucht: 29. 10.2015.

OSGEO (2015). GEOS - Geometry Engine Open Source. <http://geos.osgeo.org/doxygen/index.html>; zuletzt besucht: 24. 11.2015.

OSM Community (2015). Openstreetmap wiki. [https://wiki.openstreetmap.org/wiki/Main\\_Page](https://wiki.openstreetmap.org/wiki/Main_Page); zuletzt besucht: 28.10.2015.

OSM Community (2016). OpenStreetMap Wiki - Indoor Mapping. [http://wiki.openstreetmap.org/wiki/Indoor\\_Mapping](http://wiki.openstreetmap.org/wiki/Indoor_Mapping); zuletzt besucht: 28. 02.2016.

pgRouting Community (2016a). pgRouting. <http://pgrouting.org>; zuletzt besucht: 18.02.2016.

pgRouting Community (2016b). pgRouting workshop manual. <http://workshop.pgrouting.org/>; zuletzt besucht: 18.02.2016.

PostgreSQL Community (2016). Postgresql about page. <http://www.postgresql.org/about/>; zuletzt besucht: 23.02.2016.

Project OSRM (2015). Osrn map. <http://map.project-osrm.org>; zuletzt besucht: 28.10.2015.

Schuster, W. (2012). Partizipative Karten-und Routendienste für Menschen mit Mobilitätsbehinderung–Herausforderungen für Datenmodellierung und Interface-design.

Stark, M. and Torlach, V. (2003). *Realisierungsaufwand zur Herstellung einer Geodatenbasis für die intermodale Routenberechnung zwischen Individual- und öffentlichem Personennahverkehr*. na.

TagInfo (2016). TagInfo. <http://taginfo.openstreetmap.org/>; zuletzt besucht: 24. 11.2015.

Topf, J. (2012). Das Osmium-Framework. YouTube. <https://www.youtube.com/watch?v=CXTrEbKkt-Y>; zuletzt besucht: 24.11.2015.

Vetter, C. and Luxen, D. (2011). MoNav & OSRM: 1 Jahr später. In e. V., F., editor, *FOSSGIS Tagungsband – Anwenderkonferenz für Freie und Open Source Software für Geoinformationssysteme*, pages 42–43.

Wikipedia (2016a). PostGIS. <https://de.wikipedia.org/wiki/PostGIS>; zuletzt besucht: 13.02.2016.

Wikipedia (2016b). Problem des Handlungsreisenden. [https://de.wikipedia.org/wiki/Problem\\_des\\_Handlungsreisenden](https://de.wikipedia.org/wiki/Problem_des_Handlungsreisenden); zuletzt besucht: 28.02.2016.

Wikipedia (2016c). R-Baum. <https://de.wikipedia.org/wiki/R-Baum>; zuletzt besucht: 28.02.2016.

## A Relevante Attribute der OSM-Geometrien

**Tabelle 1:** Straßenspezifisches Tagging in OpenStreetMap

| Schlüssel | Wert            | Eigenschaften  | Annahmen   |
|-----------|-----------------|--|--|
| highway   | motorway,       | meist mehrspurig,  | kein Bürgersteig,  |
|           | trunk           | autobahnartig  | nicht überquerbar  |
| highway   | primary         | meist mehrspurig   | Bürgersteig innerorts,<br>vorhanden, nicht<br>überquerbar    |
| highway   | secondary,      | kleinere Straßen   | Bürgersteig standardmäßig                                    |
|           | tertiary        | als 'primary'  | vorhanden,<br>riskantes Überqueren,<br>wenn mehrspurig       |
| highway   | unclassified    | sehr kleine Straßen,<br>verbinden Dörfer und Weiler                    | kein Bürgersteig   |
| highway   | road            | unbekannter Straßentyp   | ignorieren   |
| highway   | residential     | Straßen in Wohngebieten  | mit Bürgersteig,<br>Überquerung grundsätzlich<br>möglich     |
| highway   | motorway_link,  | Ab- und Auffahrten,  | nicht überquerbar  |
|           | trunk_link,     | gleiche Kategorisierung  |  |
|           | primary_link,   |  |  |
|           | secondary_link, |  |  |
|           | tertiary_link   |  |  |
| highway   | living_street   | Fußgänger haben Vorrang,<br>im Sinne der deutschen<br>Spielstraße      | Gleichbehandlung wie<br>Fußweg                               |
| highway   | pedestrian,     | FußwegFahrräder erlaubt,   | kein Bürgersteig,  |
| highway   | footway         | wenn <i>bicycle=yes</i> angegeben<br>ist                               | Geometrie kann als<br>Fußweg übernommen<br>werden            |
| highway   | cycleway        | Fahrradweg   | Nur begehbar, wenn <i>foot=yes</i><br>gesetzt ist.           |
| highway   | track           | Feldweg,<br>Qualität kann im<br>Tag <i>tracktype</i> definiert<br>sein | kein Bürgersteig,<br>Geometrie wird als<br>Fußweg übernommen |

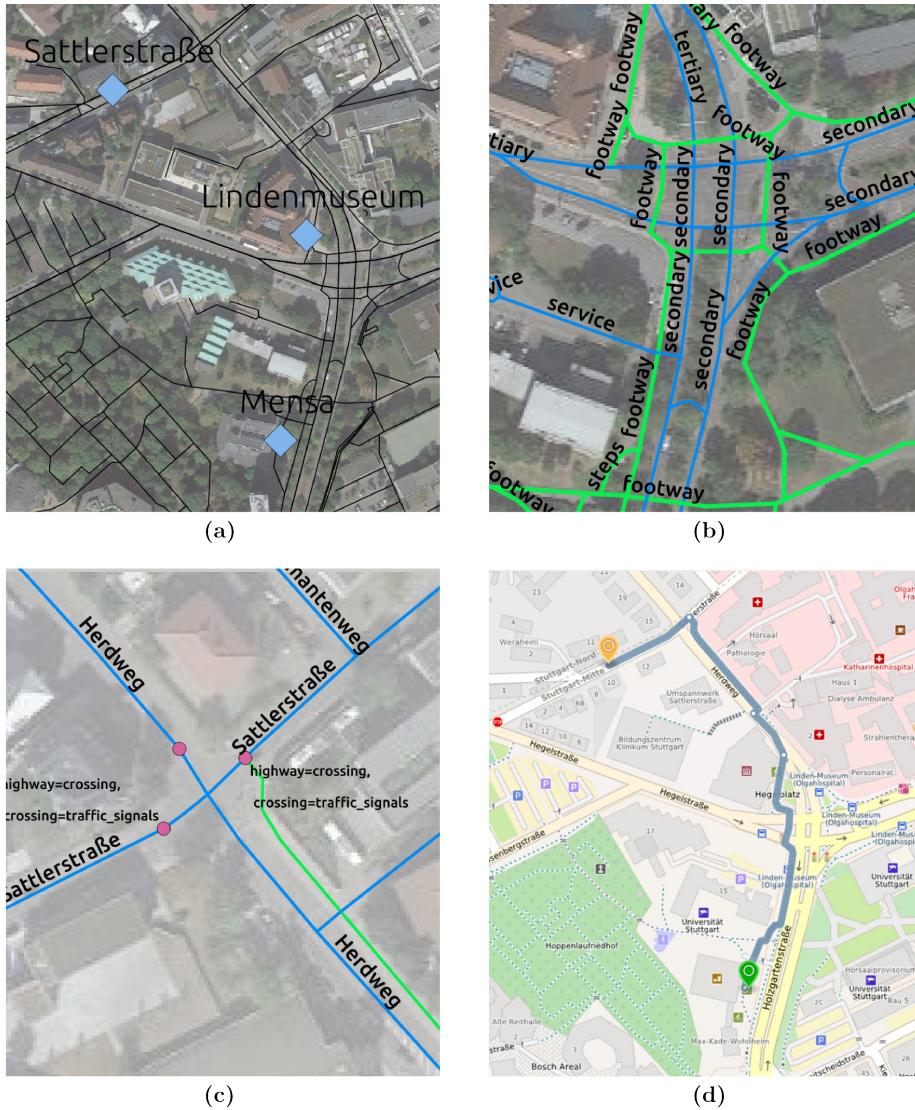
**Tabelle 1:** Straßenspezifisches Tagging in OpenStreetMap (Fortsetzung)

| Schlüssel | Wert          | Eigenschaften                                   | Annahmen   |
|-----------|---------------|---|--|
| highway   | path          | undefinierter Pfad,<br>z.B. Trampelpfad im Wald | Für Fußgänger begehbar,<br>jedoch eingeschränkt mit<br>Kinderwagen und Rollstuhl                                       |
| highway   | steps         | Treppen   | eingeschränkte Begehbarkeit  |
| incline   | [0..9][° / %] | Steigung in Grad<br>oder Prozent                | könnte berücksichtigt<br>werden, z.B. als Malus<br>in der Streckenmessung<br>oder als Hindernis für<br>Rollstuhlfahrer |
| junction  | roundabout    | Kreisverkehr                                    | innen kein Bürgersteig   |
| lanes     | [0..9]        | Anzahl der Spuren                               | Spurbreite: 3 Meter  |
| lit       | yes / no      | Straßenbeleuchtung                              | interessant für eine<br>Nachtnavigation  |
| motorroad | yes           | autobahnartig                                   | kein Bürgersteig,<br>nicht überquerbar   |

**Tabelle 2:** Tagging der Überwege in OpenStreetMap

| Schlüssel    | Wert            | Eigenschaften  |
|--------------|-----------------|--|
| highway      | crossing        | Fußgänger- und / oder Fahrradüberweg auf einem KFZ-Way                 |
| footway      | crossing        | Fußgängerüberweg auf einem Fußweg                                      |
| cycleway     | crossing        | Fahrradüberweg auf einem Fußweg  |
| crossing     | traffic_signals | Verkehrsfluss ist durch eine Ampelanlage geregelt                      |
| crossing     | uncontrolled    | Keine Ampel vorhanden, jedoch Straßenmarkierungen (z.B. Zebrastreifen) |
| crossing     | no              | Keine Überquerung möglich / erlaubt                                    |
| crossing     | unmarked        | Überweg ohne Ampel oder Straßenmarkierung                              |
| crossing     | unknown         | Unbekannte Überquerungsart   |
| crossing_ref | zebra           | Zebrastreifen, nur Fußgänger erlaubt                                   |
| crossing_ref | tiger           | Zebrastreifen, Fußgänger und Fahrradfahrer erlaubt                     |
| crossing_ref | pelican         | Überweg mit Ampelanlage, nur Fußgänger erlaubt                         |
| crossing_ref | toucan          | Überweg mit Ampelanlage, Fußgänger und Fahrradfahrer erlaubt           |
| crossing_ref | pegasus         | Überweg mit Ampelanlage, Fußgänger, Fahrradfahrer und Reiter erlaubt   |

## B Testgebiete



**Abbildung 1:** Testgebiet Stuttgart. (a) Übersicht des Testgebiets, OSM-Ways (schwarz), Routingpunkte (hellblau). (b) Karteninformationen der Kreuzung Lindenmuseum, Fußwege (grün), KFZ-Wege (blau), Beschriftung: highway-Tags. (c) Kreuzung Sattlerstraße-Herdweg, Fußwege (grün), KFZ-Wege (blau), Beschriftung: Tags der Nodes und Straßennamen. (d) Vergleichsroute OpenRouteService (OpenStreetMap). (Satellitenbilder: Google Maps)



**Abbildung 2:** Testgebiet Biberach an der Riß. (a) Übersicht des Testgebiets, OSM-Ways (schwarz), Routingpunkte (hellblau). (b) Karteninformationen des Testgebiets, Fußwege (grün), KFZ-Wege (blau), Beschriftung: `highway`-Tags. (c) Flächen mit `highway=pedestrian` (rot). (d) Vergleichsroute OpenRouteService (OpenStreetMap). (Satellitenbilder: Google Maps)

## C Speicherbedarf der Objekte in PedRO

**Tabelle 3:** Speicherbedarf PedRO

| Objekt              | Attribute (Typ)                                    | Quantität  |
|---------------------|--|--|
| location_handler    | lonlat (2 double)                                  | Anzahl aller Nodes des   |
|                     | ID (long)  | Datensatzes  |
| pedestrian_road_set | pedestrian (Pedestrian-Road)                       | Anzahl Fußwegstücke  |
| sidewalk_map        | connection_string (string) sidewalk (Sidewalk)     | Länge der VehicleRoads / Frequentierung der Überwege                         |
| crossing_set        | crossing (Crossing)                                | Anzahl OSM-Kreuzungen + Länge der VehicleRoads / Frequentierung der Überwege |
| pedestrian_road_map | ID (long)  | Anzahl OSM-Nodes der Fußwege   |
|                     | pedestrians (vector(long))                         | (2x)   |
| vehicle_node_map    | ID (long) vehicles (VehicleMapView)                | Anzahl KFZ-Wegstücke (2x)  |
| crossing_node_map   | ID (long) crossing (void)                          | Anzahl OSM-Kreuzungen  |
| finished_segments   | connecton_string (string)                          | Anzahl KFZ-Wegstücke (2x)  |
|                     | sidewalks (pair(void, void))                       |  |
| ortho_tree          | boundingbox (Envelope) geometry (void) index (int) | Länge PedestrianRoads / Frequentierung                                       |
| sidewalk_tree       | boundingbox (Envelope) geometry (void) index (int) | Anzahl Bürgersteiggeometrien / Frequentierung                                |
| temp_pedestrian_map | pedestrians (void)                                 | Anzahl PedestrianRoads (2x)  |
|                     | pedestrians (vector(void))                         |  |
| temp_sidewalk_map   | sidewalks (void)                                   | Anzahl Bürgersteiggeometrien   |
|                     | sidewalks (vector(void))                           | (2x)   |
| new_sidewalk_set    | sidewalk (void)                                    | Anzahl Bürgersteiggeometrien   |

## D UML-Diagramm

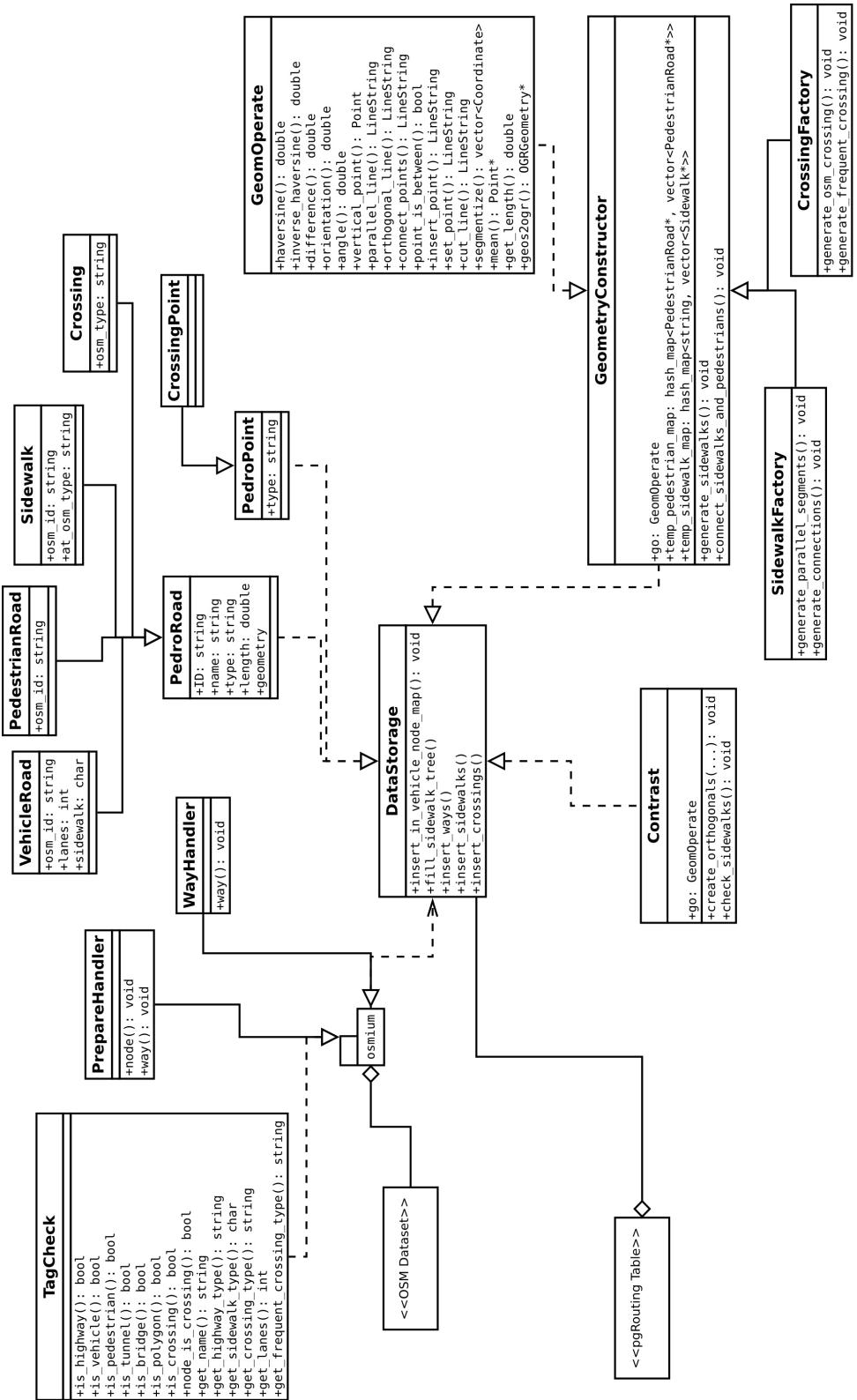


Abbildung 3: Übersicht UML PedRO. Laufzeit etwa in Uhrzeigersim, begonnen mit dem Einlesen von OSM.