Robert Hofstadler, BSc

Routingdienst für Einsatzkräfte auf Basis von Open Source und freien Geodaten

MASTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

Masterstudium Geomatics Science

eingereicht an der

Technischen Universität Graz

Betreuer

Dipl.-Ing. Dr.techn. Clemens Strauß

Institut für Geodäsie

Graz, Oktober 2015

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum

Unterschrift

Danksagung

Rückblickend auf diese Masterarbeit gibt es viele Personen, ohne deren Hilfe es mir nicht möglich gewesen wäre, die Arbeit in dieser Form umzusetzen. Einigen davon würde ich gerne anschließend persönlich meinen Dank aussprechen.

Als Erstes möchte ich mich bei meinen Eltern für die moralische und finanzielle Unterstützung im Zuge meines gesamten Studiums bedanken. Des Weiteren danke ich meiner Freundin, welche mir äußerst hilfreich mit Rat und Tat im Laufe dieser Masterarbeit beigestanden hat.

Besonders bedanken möchte ich mich auch bei Herrn Dipl.-Ing. Dr.techn. Clemens Strauß für die umfangreiche Betreuung dieser Arbeit. Die zahlreichen Gespräche haben mir geholfen die angestrebten Ziele in dieser Form zu erreichen.

Dank gebührt auch meinen Studienkollegen Roman Wilfinger und Julian Filwarny für ihre hilfreichen Anregungen und Diskussionen, sowie Daniela Triebelnig für das Korrekturlesen dieser Arbeit.

Abschließend möchte ich auch einen Dank an den Staat Österreich aussprechen, durch welchen der freie Universitätszugang für jedermann gewährt wird.

Danke!

Kurzfassung

Ziel dieser Masterarbeit ist es, auf Basis freier Geodaten und Open Source Software einen Routingdienst für Einsatzkräfte zu erstellen. Dieser Routingdienst soll für Aufgaben im In- sowie im Ausland eingesetzt werden können, sofern entsprechende Geodatensätze vorhanden sind. Mögliche Anwendung findet der Dienst beispielsweise bei Feuerwehr, Bundesheer, Rettung und Exekutive. Da es sich um ein Open Source GIS handelt, wird als Grundgerüst QGIS verwendet. In einer eigens auf QGIS-Basis erstellten grafischen Benutzeroberfläche (Standalone-Applikation) werden routingrelevante Operationen durchgeführt. Die für das Routing benötigten geocodierten Straßendaten entstammen der OpenStreetMap und werden für die Erstellung einer routingfähigen Knoten-Kanten-Struktur herangezogen. Um auch Höheninformationen in das Routing mit einfließen zu lassen, wurden unterschiedliche Höhendatensätze näher untersucht. Mit dieser Anwendung können neben dem Auffinden des schnellsten Weges zwischen zwei Orten A und B auch Bedingungen miteinbezogen werden. So ist es möglich, Tunnel und Brücken, aufgrund von Umwelteinflüssen oder nicht passenden Fahrzeugeigenschaften (Breite oder Gesamtgewicht) teilweise oder zur Gänze von der Routenfindung auszuschließen. Weiters können Sperrzonen in den Straßendaten definiert werden, die vom Routing ausgeschlossen werden sollen. Diese Sperrbereiche können nicht nur selbst erstellt, sondern auch von Dritten bezogen und in Form von Shape-Dateien oder aus Tabellen einer Datenbank in die Anwendung geladen werden.

Abstract

The goal of this thesis is the generation of a routing service for emergency forces based on free geodata and open source software. The routing service shall be used for tasks in Austria as well as abroad, as far as appropriate geodata sets are available. A possibly usage of the service is in the domain of fire brigade, military, ambulance and executive authorities. To match with the focus on open source, QGIS was used as a basic framework. A graphical user interface (standalone-application) has been developed on basis of QGIS to perform routing-relevant operations. The geocoded road data needed for the preparation of routable nodes-edges-structures are taken from the OpenStreetMap. To include even height information into the routing process, different elevation models have been analyzed in more detail. Beside the detection of the quickest path between two locations A and B, conditions can be included in this application as well. This is how tunnels and bridges can be partially or completely excluded from the route determination due to environmental factors or unsuitable vehicle properties (width or overall weight). Furthermore restricted zones can be defined to eliminate specific parts of a road. Those restricted areas can not only be created by oneself, but can also be purchased from third parties and loaded into the application in form of shape-files or tables from databases.

Inhaltsverzeichnis

1 Einleitung

Routing und die damit einhergehenden Anwendungen sind aus dem heutigen Alltag kaum mehr wegzudenken, weder im Berufsleben, noch in der Freizeit. Will man als Einzelperson nur möglichst schnell von einem Ort zum nächsten kommen, so spart eine optimal geplante Route in der Wirtschaft nicht nur Ressourcen, sondern bringt auch einen klaren Wettbewerbsvorteil gegenüber anderen. Seit GPS für den zivilen Sektor freigegeben wurde, ist es ein Leichtes seinen eigenen Standort zu bestimmen und unter Bekanntgabe eines Zielpunktes sich mittels Routingalgorithmen dorthin führen zu lassen. Viele Handelsunternehmen verfügen auf ihren Webseiten über implementierte Routingalgorithmen die, sofern gewünscht, den Kunden auf kürzestem Wege zur nächstgelegenen Filiale führen. Die Grundlage für ein Routing bildet ein georeferenziertes Wegenetz auf dem aufbauend Routingalgorithmen arbeiten. Diese Geodaten werden von großen digitalen Kartenherstellern erhoben und kostenpflichtig zum Kauf angeboten. Durch das OpenStreetMap-Projekt bekamen diese Anbieter einen umfangreichen und sich ständig verbessernden Konkurrenten, der seine Daten als freie Geodaten kostenlos und für jeden zu Verfügung stellt. Dieses Projekt beruht auf ständigem Geben und Nehmen und nur so ist es möglich einen weltumspannenden und detailreichen Datensatz zu generieren. Mittlerweile gibt es viele verschiedene Dienste, zum Teil Open Source, welche aufbauend auf der OpenStreetMap Lösungen für unterschiedliche Routingproblematiken anbieten.
Diese Masterarbeit zielt darauf ab, auf Basis freier Geodaten und Open Source Software einen Routingdienst für Einsatzkräfte zu schaffen, welcher sowohl für Aufgaben im In- sowie im Ausland eingesetzt werden soll.
Die Arbeit ist wie nachfolgend beschrieben gegliedert. Das Kapitel 2 behandelt die Grundlagen und gibt Aufschluss über den theoretischen Hintergrund. Im Kapitel 3 wird auf die Implementierung, im Speziellen auf den Aufbau der Standalone-Applikation und das Zusammenwirken von Software und verwendeter Bibliotheken eingegangen. Das anschließenden Kapitel 4 veranschaulicht die Lösungen zu unterschiedlichen Fragestellungen unter Bezug von räumlich variierenden Straßendatensätzen. Im letzten Kapitel wird ein Resümee gezogen und mögliche Verbesserungen diskutiert.

2 Grundlagen

2.1 Geoinformationssysteme

Unter einem Geoinformationssystem (GIS) versteht man ein System, mit dem Geodaten (siehe Abschnitt 2.1.1) erfasst, weiterverarbeitet und visualisiert werden können. Ein GIS besteht aus Hardware, Software, (Geo-)Daten und Datenbanken. GIS kommen überall dort zum Einsatz, wo geographische Daten im Spiel sind. Anwendungsgebiete sind z.B. Raum- und Stadtplanung, Kartographie und Katastrophenschutz. GIS lassen sich in eine Vielzahl an Unterarten einteilen. Im Nachfolgendem sind drei davon angeführt:

- Landinformationssystem (LIS)
- Netzinformationssystem (NIS)
- Umweltinformationssystem (UIS)

LIS bieten verschiedene digitale thematische Karten an. Diese reichen von großmaßstäblichen Straßenkarten bis hin zu detaillierten Karten für diverse Freizeitaktivitäten. Eine mögliche Umsetzung eines NIS ist der Leitungskataster von Elektrizitäsversorgungsunternehmen (EVU). Darin werden alle Leitungen, egal ob es sich um eine Freileitung oder um eine Leitung im Erdreich handelt, über die ein EVU verfügt, eingezeichnet. Im Schadensfall liegt der Vorteil des LIS darin, schnell abfragen zu können, wer davon betroffen ist und wie man eine Instandsetzung am effizientesten durchführt. Mittels UIS lassen sich beispielsweise Überflutungsgebiete leicht erfassen. Zwar lassen sich zukünftige Katastrophen dadurch nicht verhindern, jedoch können damit Schäden soweit vermieden werden, indem man Bereiche ausweist, in denen zukünftig nicht mehr gebaut werden darf.
GIS können als Desktop- oder als online-Anwendung genutzt werden. Neben den kommerziellen GIS Anbietern (z.B. ArcGIS[1]) drängen immer mehr Open Source[2] Projekte auf den Markt (z.B. QGIS[3]).

2.1.1 Geodaten

Geodaten sind digitale Daten denen ein Raum- und ein Sachbezug zu Grunde liegt. Der Raumbezug kann 2 (Lage)- oder 3- dimensional (Lage und Höhe) sein. Der Sachbezug oder die Semantik wird durch nicht geometrische Elemente, den Attributen, wie zum Beispiel Texte, Zahlen, Messwerte oder Eigenschaften wiedergegeben. Eine große Bedeutung kommt bei Geodaten den Metadaten zu. Metadaten sind Daten über Daten welche essentielle Informationen über die eigentlichen Daten beinhalten. Diese verfügen über wichtige

[1] `https://www.arcgis.com`
[2] `http://opensource.org`
[3] `https://www.qgis.org`

Informationen wie das verwendete Koordinatensystem und weitere geodätische Datensätze wie Projektionen oder Referenzellipsoide (siehe Abschnitt 2.8). Geodaten unterscheiden sich in Raster- und Vektordaten. Die Geometrie von Vektordaten wird durch Punkte, Linien und Polygone repräsentiert. Vektordaten sind skalierbar, dass heißt sie können ohne Informationsverlust verkleinert oder vergrößert werden und benötigen aufgrund dieser Eigenschaft viel weniger Speicherplatz als Rasterdaten. Im Falle von Rasterdaten wird die Geometrie durch eine Matrix aus Pixeln dargestellt. Die jeweilige Größe eines Pixels hängt von der Auflösung der Erfassungsmethode ab. Die Geodatenobjekte (Punkte, Linien, Polygone) werden auch als Entitäten bezeichnet. Um Geodaten leichter interoperabel zu machen, wurde das Open Geospatial Consortium[4] (OGC) gegründet, welches sich aus Universitäten, Regierungsorganisationen und der Industrie zusammensetzt. Der Schwerpunkt des OGC liegt in der Entwicklung offener Standards. Hierbei geht es nicht primär um Software, sondern um die Definition von Schnittstellen, die einen reibungslosen Austausch der Geodaten garantieren. Ein bekanntes Beispiel für eine OGC-Spezifikation ist Simple Feature Access [5] (aktuelle Version 1.2.1). Diese Spezifikation definiert Zugriff, Speicherung und räumliche Operationen der Geometrieobjekte. Mittlerweile ist Simple Feature Access auch ein ISO-Standard (19125) geworden.
Die Geometrie wird in zwei Formaten gehalten. Einerseits in dem Well-known Text (WKT) Format und auf der anderen Seite im Well-known Binary (WKB) Format. Bei WKT handelt es sich um ein menschenlesbares Format im Gegensatz zu WKB. In Tabelle 2.1 ist ein Punkt mit den Koordinaten (2,2) in beiden Formaten dargestellt.

Tabelle 2.1: WKT - WKB

WKT	**WKB**
Point (2 2)	01010000000000000000000040000000000000040

Um Daten in Datenbanken zu transferieren und zu speichern, wird das WKB-Format eingesetzt. WKT und WKB werden von einer Vielzahl an räumlichen Datenbanken unterstützt.

Da alle Koordinaten immer einem Referenzsystem zu Grunde liegen, muss dieses für eine eindeutige Identifikation der Koordinaten bekannt sein. Der in Tabelle 2.1 angegebene Punkt, mit den Koordinaten (2,2), könnte sich demnach überall auf der Erde befinden. Damit das GIS den Punkt eindeutig und richtig wiedergibt, benötigt das System Informationen über das verwendete Koordinatensystem und über eine mögliche Projektion. Realisiert wurde das mittels dem Spatial Referenz Identifier (SRID). So hat jedes Koordinatensystem bzw. jede verwendete Projektion eine eindeutige Nummer. Die Maße der SRID-Code stammen von der European Petroleum Survey Group Geodesy (EPSG), die Mitte der 1980er Jahre, eindeutige Nummern für diverse Koordinatenreferenzsysteme zu vergeben begann. In Tabelle 2.2 sind beispielhaft drei unterschiedliche SRID-Nummern und das zu Grunde liegende Koordinatensystem mit möglicher Projektion angegeben.

[4] `http://http://www.opengeospatial.org`
[5] `http://www.opengeospatial.org/standards/sfa`

Tabelle 2.2: SRID

SRID	System bzw. Projektion
4326	WGS84 (φ, λ)
32633	UTM 33 N Projektion WGS84 (x, y)
3857	Pseudo-Mercator Projektion WGS84 (x, y)

Einen Sonderfall der SRID-Codes stellt die letzte Zeile in Tabelle 2.2 mit SRID 3857 dar. Bei diesem Code handelt es sich um eine Pseudo-Mercator Projektion. Hierbei werden die ellipsoidischen WGS84 Koordinaten als Kugelkoordinaten betrachtet und auf einen Zylinder (siehe Abschnitt 2.8.1 projiziert. Zur Anwendung kommt dieser SRID-Code bzw. diese Projektion u.a. bei OSM und bei Google Maps.

2.1.2 Geodatenformate

Mittlerweile gibt es eine Vielzahl an unterschiedlichen Datenformaten für Vektor- sowie Rasterdaten. Für Vektordaten ist wahrscheinlich das meist verbreitetste Format das Shapefile-Format. Bei Rasterdaten wird häufig das Geo Tagged Image File Format (GeoTIFF) eingesetzt. Hierbei handelt es sich um eine Erweiterung des TIFF-Formats. Durch Georeferenzierung bekommt jedes Pixel Koordinaten zugewiesen. In Grime[6] befindet sich eine ausführliche Auflistung der verschiedene Geodatenformate.

2.2 OpenStreetMap

Das OpenStreetMap (OSM)-Projekt wurde 2004 von Steve Coast mit dem Ziel initiiert, eine freie digitale Weltkarte nach dem Wiki-Prinzip[7] zu schaffen. Jeder darf diese Daten nutzen und sich an dem Projekt beteiligen. Die Grunddaten, im Speziellen Straßennetze der OSM wurden teilweise von Firmen und Regierungsorganisationen zur Verfügung gestellt. Weiters bilden Luftbilder von Yahoo und Bing eine wichtige Grundlage zur Vervollständigung der Karte. Diese Luftbilder werden von Hobbykartographen, den sogenannten Mappern, abgezeichnet. Es können auch selbst aufgezeichnete „Tracks“ in die OSM hochgeladen und integriert werden. Im Vordergrund steht die Nutzung freier Geodaten. Auf Grundlage der OSM lassen sich eigene Karten, für individuelle Anforderungen erstellen. Wie zum Beispiel Fahradkarten oder Rohlstuhlkarten. Die gesamte Datensammlung der OSM steht unter der Creative Commons Lizenz CC-BY-SA 2.0[8]. Diese Lizenz besagt, dass man die Daten verarbeiten, verbreiten, veröffentlichen und vervielfältigen darf unter der Bedingung, dass der Rechteinhaber/Urheber samt Lizenz und URL genannt wird (CC-BY-SA 2.0). Die OSM-Daten sind in einer PostgreSQL-Datenbank, welche sich auf dem Gelände des Imperial College[9] in Großbritannien befindet, gespeichert. Unter planet.osm stehen alle in der OSM-Datenbank enthaltenen Daten zum Download bereit. Diese Datei wird als das „planet-file“ bezeichnet und hat eine unkomprimierte Größe von 576,6 GB (komprimiert 28,8 GB als *.pbf-Datei, Stand 19.5.2015 17:17 Uhr). Diese Gesamtdatei wird wöchentlich auf den neuesten Stand gebracht. Um nicht jedes Mal die Gesamtdatei downloaden

[6] `http://www.grime.net/gistools/s.htm`
[7] `http://de.wikipedia.org/wiki/Wiki`
[8] `http://creativecommons.org`
[9] `https://www.imperial.ac.uk`

und prozessieren zu müssen, stehen eine Reihe an Webseiten zur Verfügung, die wiederum spezielle geografische Bereiche vorprozessiert haben und anbieten. Auf Geofabrik[10] stehen sämtliche Straßendaten des „planet-files“ kostenlos zur Verfügung. Der Detailgrad der Karte wird zunehmend genauer und als Bezugsellipsoid wird das World Geodetic System 1984 (WGS84) verwendet. Die Koordinaten liegen in Form von geografischer Breite φ und Länge λ vor.

2.2.1 OSM Datenmodell

OSM-Daten basieren auf dem Extensible Markup Language (XML) - Standard und es gibt drei Objekttypen: Punkt (Node), Weg (Way) und Relation. Attribute, die Objekttypen zugewiesen werden um zu beschreiben um was es sich handelt, werden als Tags bezeichnet. Alle Objekte sind mit einer eindeutigen Kennung versehen. Jeder Knoten und jeder Weg kann beliebig viele Tags beinhalten. Jedes einzelne Tag besteht aus einem Schlüssel (Key) und einem Wert (Value). Jeder Schlüssel darf pro Objekt nur einmal enthalten sein. In Summe hängen alle Punkte, Wege und Relationen zusammen und repräsentieren die editierbare OpenStreetMap-Weltkarte.

Datentyp Node

Ein Node beinhaltet folgende Informationen:

- geographische Breite und Länge (WGS 84, siehe Abschnitt 2.6.3)
- Änderungszeitpunkt
- beliebige Menge an Tags

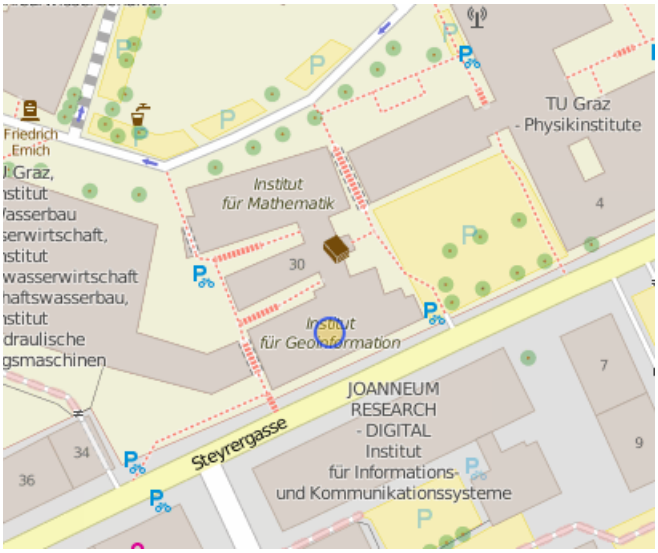


Abbildung 2.1: OSM - Node

[10] `http://www.geofabrik.de`

Nachfolgend ist ein Auszug der XML-Daten zu Abbildung 2.1 angeführt:

```
<node id="1484954795"lat="47.0641885"lon="15.4530975">
<tag k="addr:city"v="Graz"/>
<tag k="addr:country"v="AT"/>
<tag k="addr:housenumber"v="30"/>
<tag k="addr:postcode"v="8010"/>
<tag k="addr:street"v="Steyrergasse"/>
<tag k="amenity "v="university"/>
<tag k="name "v="Institut für Geoinformation"/>
<tag k="wheelchair" v="limited"/>
</node>
```

Datentyp Way

Ein Way besteht aus mindestens 2 Nodes und beinhaltet folgende Informationen:

- Liste der im Way enthalteten Nodes
- Änderungszeitpunkt
- beliebige Menge an Tags

Abbildung 2.2: OSM - Way

In Abbildung 2.2 ist der gegenüberliegende Straßenzug vom Geodäsiegebäude in der Steyrergasse 30 als Way abgebildet. Flächen lassen sich aus mehreren Ways modellieren, sofern die Tags untereinander übereinstimmen (siehe Abbildung 2.3).

Datentyp Relation

Mit Relationen lassen sich Beziehungen zwischen den anderen Datentypen (Node und Way) modellieren. Eine Relation enthält nachfolgende Informationen:

Abbildung 2.3: OSM - Area

- Liste aus Punkten und Wegen aus denen sich Relation zusammensetzt
- Änderungszeitpunkt
- Beliebige Menge an Tags

Sucht man in der OSM nach einer Relation mit dem Schlüssel-Wert-Paar „name“=“Graz“, so bekommt man das Stadtgebiet von Graz (siehe Abbildung 2.4).

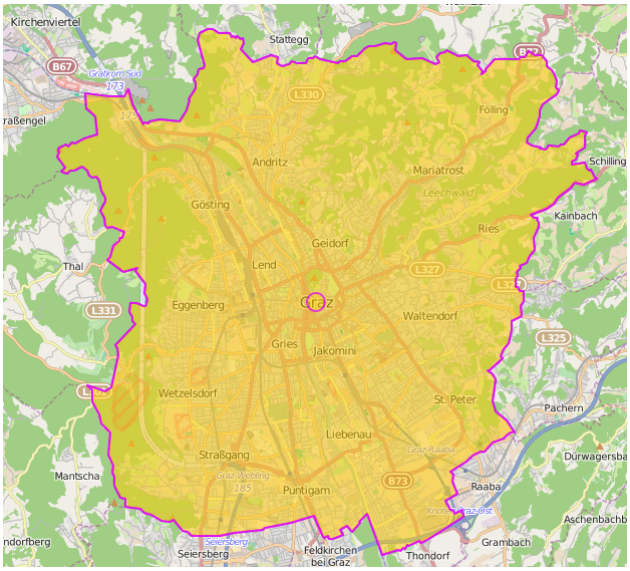


Abbildung 2.4: OSM - Graz

Straßen

Eines der wichtigsten Objekte der OpenStreetMap sind die Straßen. Sie werden mit dem Schlüssel „highway“ gekennzeichnet. Als Werte können alle möglichen Varianten einer Straße angeführt werden. Hier reicht die Pallette von A wie Autobahn, über F wie Fußweg bis hin zu Z wie Zebrastreifen. Wie oben erwähnt, dürfen Schüssel-Tags nur einmal pro Knoten oder Weg enthalten sein. Für den Fall, dass entlang einer Straße ein Radweg verläuft und baulich keine Abgrenzung besteht, so müssen Straße und Radweg separat modelliert werden. Eine genaue Aufstellung der unterschiedlichen Werte für den Schlüssel „highway“ ist unter[11] aufgelistet.

2.2.2 OSM Karte

Die Bedienung der OSM-Karte ist gleich intuitiv angelegt wie beispielsweise eine Karte von Google Maps[12]. Die Standard-Kartenansicht[13] der OSM wird „Slippy Map“ genannt und stellt die Erde von 85° südlicher bis 85° nördlicher Breite dar. Da durch Zoomen und Verschieben ein beliebiger Ausschnitt der Weltkarte gewählt werden kann, müsste jedes mal der Server die Karte mitsamt allen Objekten neu rendern. Da dies sehr rechen- und zeitintensiv ist, wurde dieses Problem mit einer Anordnung von Kacheln (engl. tiles) gelöst, die bereits vorberechnet sind. Aufgrund der Pseudo-Mercator Projektion (siehe Tabelle 2.2 und Abschnitt 2.8.1) ist eine quadratische Darstellung der Kacheln möglich. Es gibt 19 verschiedene Zoomstufen ($z : 0 - 18$), wobei 0 keinen Zoom bedeutet und 18 die maximal mögliche Zoomstufe wiedergibt. Je tiefer gezoomt wird, desto mehr Kacheln werden vom Tile-Server angefordert. Mittels Gleichung (2.1) lässt sich die Anzahl der Kacheln k abhängig von der Zoomstufe z berechnen.

$$k = 4^z \tag{2.1}$$

In Tabelle 2.3 ist ein Auszug der möglichen Zoomstufe, sowie die dafür benötigte Anzahl an Kacheln und der ungefähre Maßstab der Karte angeführt. Für detaillierte Informationen bezüglich der Kachelberechnung wird auf Ramm und Topf, 2010, Kapitel 14 verwiesen.

Tabelle 2.3: Kacheln

Zoomstufe z	Kacheln	Maßstab
0	1	$1 : 5 * 10^8$
1	4	$1 : 2.5 * 10^8$
10	$1 * 10^6$	1 : 50000

2.3 Datenvorhaltung

Datenakquise und Datenarchivierung sind stark miteinander verwurzelt. Die Erhebung und Anschaffung von Daten, insbesondere von Geodaten, ist mit einem hohen Zeitaufwand verbunden, was sich wiederum in einem ebenfalls hohen finanziellen Aufwand wiederspiegelt.

[11] `http://wiki.openstreetmap.org/wiki/Key:highway`
[12] `https://maps.google.at`
[13] `http://osm.org`

Aus diesem Grund sollten diese Daten möglichst lange mit einem hohen Maß an Sicherheit zur Verfügung stehen, damit eine neuerliche Erfassung ausgeschlossen werden kann. Da solche Daten, beispielhaft genannt sei hier ein Bankkonto oder eine Grundstücksvermessung, auf längere Sicht benötigt und genutzt werden, ist auch eine langfristige Datenarchivierung von Nöten, die eine sichere Speicherung und Bereitstellung gewährleistet.
Die einfachste Art Daten elektronisch zu speichern ist, wenn man sie in sogenannte *flat files* (flache Dateien) organisiert. Mit dem Adjektiv flach wird die Struktur dieser Datei gemeint. Für viele Anwendungen ist diese Struktur bei weitem ausreichend. Denkt man hier zum Beispiel an ein Messgerät, das den Wasserfüllstand in einem Gefäß misst und die Messung in eine gewöhnliche Textdatei schreibt, so könnte die Datenstruktur wie in Tabelle 2.4 aussehen.

Tabelle 2.4: flat file

Datum	**Uhrzeit**	**Füllstand**
01.05.2015	12:00	17 mm
02.05.2015	12:00	23 mm

Für eine Anwendung dieser Art ist diese Form der Datenspeicherung und Datenstruktur leicht ausreichend. Hat man mit großen Datenmengen zu tun und geht die Datenverarbeitung in Richtung Mehrbenutzersystem, wird das Arbeiten mit *flat files* zunehmend komplizierter bis unmöglich und ein Datenverlust durch unbeabsichtigte Manipulation ist viel leichter gegeben. Um diese Nachteile der *flat files* auszuräumen und die Datenverarbeitung zu erleichtern bzw. das Abfragen zu ermöglichen, empfiehlt es sich auf eine Datenbank überzugehen.

2.4 Datenbanken

Eine Datenbank (auch Datenbanksystem, kurz DBS) ist eine Sammlung an Daten (elektronisches Archiv) in strukturierter Form, die nach vordefinierten Regeln auf elektronischen Datenträgern dauerhaft (persistent) gespeichert wird und sich durch raschen Zugriff darauf auszeichnet. Eine Datenbank gliedert sich in zwei Teile. Den Daten an sich und der Verwaltungssoftware. Die Verwaltungssoftware wird auch als Datenbankmanagementsystem (DBMS) bezeichnet. Um Daten abzufragen oder verwalten zu können, ist eine Datenbanksprache im DBS integriert. Schematisch ist der Zusammenhang in Abbildung 2.5 dargestellt.

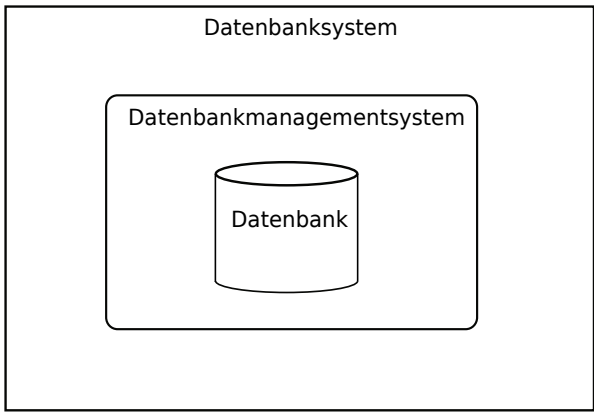


Abbildung 2.5: DBMS

2.4.1 Normalisierung

Auch wenn sich die Speicherplatzproblematik in Bezug auf Preis und Größe in den letzten Jahren zum positiven gewandt hat, sollte man auf eine strukturierte Datenhaltung mit möglichst wenig, bis keiner Redundanz hinarbeiten. Befinden sich die Tabellen einer Datenbank in diesem ökonomischen Zustand, so vereinfachen sich Wartung, Konsistenz und Abfragezeit der Daten. Durch Auferlegung gewisser Beschränkungen wird auch die Datenstruktur stabiler. Werden Beziehungen einer Datenbank so in Form gebracht, dass sie ein Höchstmaß an Stabilität der Datenstruktur haben, nennt man das auch Normalisierung, die wiederum unterschiedlich stark ausgeprägt sein kann. Unter der Normalisierung versteht man eine schrittweise Aufsplittung von einer in mehrere Relationen. Über funktionale Abhängigkeiten werden diese neu erstellten Relationen untereinander verknüpft.

1. Normalform

Die 1. Normalform liegt vor, wenn die Relation über einen Primärschlüssel (PS) verfügt und Attribute nur mehr einfache Attributwerte aufweisen.
Tabelle 2.5 liegt nicht in der 1. Normalform vor, da weder ein Primärschlüssel vorhanden ist, noch die Attribute in atomare Bestandteile zerlegt wurden.

Tabelle 2.5: 1.NF nicht erfüllt

Kunden-ID	Name	Produkt	Preis	Geschäft
999	Max Mustermann	Apfel, Birne	1,2	A
765	John Doe	Apfel	1	B

Erweitert man Tabelle 2.5 um zwei Spalten, eine für den Primärschlüssel, eine um Vor- und Nachname getrennt zu führen und um eine Zeile, damit pro Zeile nur ein Produkt vorhanden ist, so befindet sich die daraus resultierende Tabelle 2.6 in der 1. Normalform.

Tabelle 2.6: 1.NF erfüllt

ID (PS)	Kunden-ID	Nachname	Vorname	Produkt	Preis	Geschäft
1	999	Mustermann	Max	Apfel	1	A
2	999	Mustermann	Max	Birne	2	A
3	765	Doe	John	Apfel	1	B

2. Normalform

Die 2. Normalform liegt vor, wenn sich Relation in 1. Normalform befindet und jedes Nicht-Schlüssel-Attribut voll funktional vom Primärschlüssel abhängig ist. Betrachtet man Tabelle 2.6, so fällt auf, dass das Attribut Produkt nicht vom Attribut Geschäft abhängig ist. Um hier die 2. Normalform zu erfüllen, wird das Attribut Geschäft in eine eigene Relation ausgegliedert (siehe Tabelle 2.8). Damit Tabelle 2.6 die 2. Normalform erfüllt, wird diese Realtion in Tabelle 2.7 und Tabelle 2.8 aufgespalten.

Tabelle 2.7: 2. NF erfüllt I

ID (PS)	Kunden-ID	Nachname	Vorname	Produkt
1	999	Mustermann	Max	Apfel
2	999	Mustermann	Max	Birne
3	765	Doe	John	Apfel

Tabelle 2.8: 2. NF erfüllt II

ID (PS)	Preis	Geschäft
1	1	A
2	2	A
3	2	B

3. Normalform

Die 3. Normalform liegt vor, wenn sich Relation in der 2. Normalform befindet und kein Nicht-Schlüssel-Attribut vom Schlüssel-Attribut transitiv, über Umwege, abhängig ist. Betrachtet man Tabelle 2.9 so fällt auf, dass eine transitive Abhängigkeit des Herkunftslandes mit der ID über das Attribut Produkt gegeben ist.

Tabelle 2.9: Transitive Abhängigkeit

ID (PS)	Kunden-ID	Nachname	Vorname	Produkt	Herkunft
1	999	Mustermann	Max	Apfel	Stmk
2	999	Mustermann	Max	Birne	Bgld
3	765	Doe	John	Apfel	Stmk

Um die 3. Normalform zu erfüllen, werden transitive Abhängigkeiten in separate Relationen ausgegliedert. Als Schlüssel der neu erstellten Relation dient das Attribut Produkt (siehe Tabelle 2.12):

Tabelle 2.10: Transitive Abhängigkeit ausgegliedert

ID (PS)	Kunden-ID	Nachname	Vorname	Produkt
1	999	Mustermann	Max	Apfel
2	999	Mustermann	Max	Birne
3	765	Doe	John	Apfel

Tabelle 2.11: Transitiv I

ID (PS)	Preis	Geschäft
1	1	A
2	2	A
3	2	B

Tabelle 2.12: Transitiv II

Produkt (FS)	Herkunft
Apfel	Stmk
Birne	Bgld

Der Vollständigkeit halber soll erwähnt werden, dass es auch noch eine 4. und 5. Normalform gibt (siehe Schubert, 2007).

2.4.2 Datenbankmodelle

Es gibt unterschiedliche Arten von Datenbanken und die Art und Weise wie Daten verwaltet und gespeichert werden. Dies wird mit dem Datenbankmodell festgelegt. Datensicherheit, Mehrfachnutzung und Updatefreundlichkeit sind nur ein paar der Vorteile die Datenbanken gegenüber flat files mit sich bringen. Nachfolgend wird auf die unterschiedlichen Datenbankmodelle eingegangen. Für detailiertere Ausführung wird auf Bartelme, 2013 verwiesen.

Relationale Datenbank (RDB)

Das Datenbankmodell der relationalen Datenbank beruht auf Tabellen und kommt dem Alltag recht nahe. Jegliche Daten werden in Form von Tabellen abgespeichert. Jede einzelne Zeile einer Tabelle repräsentiert einen eigenen Datensatz, der sich wiederum aus einer Anzahl an Attributen (Spalten) zusammensetzt. Das Datenbankmanagementsystem wird auch als relationales Datenbankmanagementsystem (RDBMS) bezeichnet. Um Daten abzufragen oder zu verändern, kommt die Datenbanksprache „structured query language“ (SQL) zum Einsatz.

Objektorientierte Datenbank (OODB)

Bei dieser Variante einer Datenbank geht man von Relationen auf Objekte über. Dieses Datenbankmodell basiert auf dem Paradigma der objektorientierten Programmierung. Objekte sind Grundbausteine, die möglichst nahe der realen Welt nachempfunden sind. Jedes einzelne Objekt enthält Daten und Methoden. Aufgrund der Kapselung ist ein Zugreifen von außen auf die Daten nicht möglich. Die einzige Möglichkeit auf die Daten zuzugreifen, ist das Verwenden der objekt- bzw. klassenspezifischen Methoden. Ein Vorteil der OODB gegenüber der RDB besteht darin, dass alle Daten die ein Objekt betreffen, gesammelt ablegt und wenn erforderlich auch geschachtelt (für komplexe Objekte) werden. Da dieses

Datenbankmodell noch nicht allgemein anerkannt wurde, beschränkt sich die Anzahl der Schnittstellen auch noch auf ein Minimum.

Objektrelationale Datenbank (ORDB)

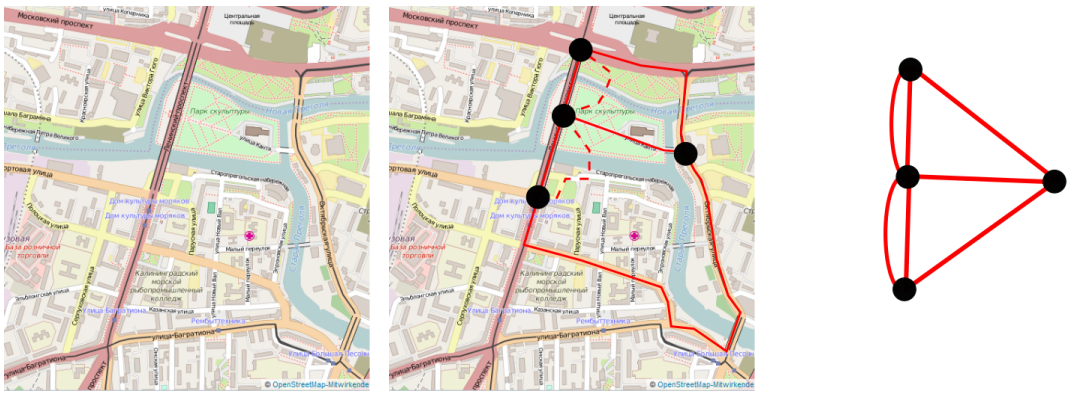
Bei einer objektrelationalen Datenbank handelt es sich um eine Synthese von relationalen und objektorientierten Datenbanktechnologien. Ausgehend von einem relationalen Konzept, wo die Daten in Tabellenform vorliegen, werden diese noch um spezielle Objekte, deren Datentyp beispielsweise Punkt, Linie oder Polygon ist, erweitert. Begründet wird das deshalb, da auszugsweise ein zweidimensionaler Punkt aus einem Koordinatenpaar besteht. Es würde hier wenig Sinn machen die x- und y-Komponente getrennt zu verwalten, da ja der Punkt an sich, und nicht seine atomaren Bestandteile, im Vordergrund steht. Häufige Anwendung findet das Modell der objektrelationalen Datenbank bei geografischen Informationssystemen (GIS), da spezielle räumliche Abfragen durchgeführt werden können.

2.5 Graphentheorie

Die Graphentheorie ist ein Teilgebiet der Mathematik, das sich mit dem Graphen und seine Relationen untereinander beschäftigt. Häufige Anwendung findet die Graphentheorie bei Optimierungsproblemen.

2.5.1 Geschichte

Die Anfänge der Graphentheorie reichen bis in die erste Hälfte des 18. Jahrhunderts zurück. Leonhard Euler veröffentlichte eine Lösung zum Königsberger Brückenproblem. Die Fragestellung lautete, ob die Möglichkeit bestand, einen Stadtrundgang durch Königsberg (heute Kaliningrad, Abbildung 2.6(a)) zu machen, ohne eine der sieben Brücken über die Pregel mehr als einmal zu betreten. Knoten repräsentieren dabei die Stadtteile und Kanten stellen die Brücken dar. Die in der Abbildung 2.6(b) rot strichlierten Linien stellen die heute nicht mehr existierenden Brücken über die Pregel dar. In Abbildung 2.6(c) ist der vereinfachte Graph von Abbildung 2.6(b) dargestellt.



(a) OSM-Karte Königsberg (b) Königsberger Brücken (c) Graph

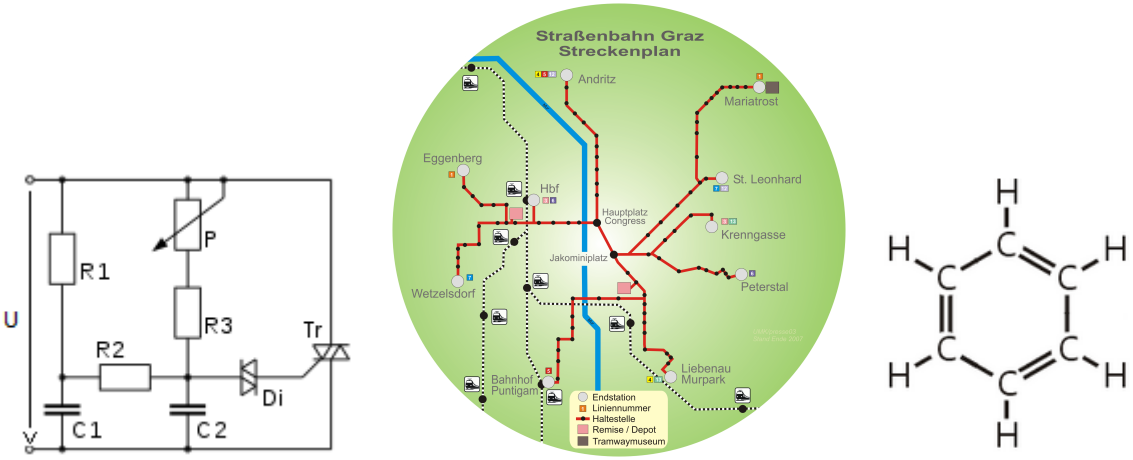
Abbildung 2.6: OSM-Karte Königsberg

2.5.2 Grundbegriffe

Ein Graph $G = G(V, E)$ ist als eine Menge von Knoten und Kanten definiert.

- V repräsentiert eine Menge an Knoten (engl. vertices, nodes) und
- E stellt eine Menge an Kanten (engl. edges) dar

Hierbei modellieren Knoten Positionen, Situationen oder Zustände und Kanten modellieren Relationen zwischen diesen Knoten. Die Graphentheorie ist heutzutage allgegenwärtig. Beispiele hierfür sind elektronische Schaltpläne (Abbildung 2.7(a)), Streckennetze von (öffentlichen) Verkehrsbetrieben (Abbildung 2.7(b)) oder chemische Verbindungen (Abbildung 2.7(c)).



(a) Schaltplan (b) Straßenbahnnetz Graz (c) Benzol

Abbildung 2.7: Anwendung von Graphen

Graphen lassen sich u.a. in nachfolgende Unterarten gliedern:

- Endlicher Graph: Wenn Menge an Knoten und Kanten endlich ist.
- Unendlicher Graph: Wenn Menge an Knoten und Kanten unendlich ist.
- Gerichteter Graph: Wenn Kanten eine Richtung aufweisen.
- Zusammenhängender Graph: Alle Knoten sind direkt oder indirekt über Kanten miteinander verbunden.
- Vollständiger Graph: Wenn Graph maximal mögliche Anzahl an Kanten hat (siehe Gleichung (2.2)).

$$E_{max} = \binom{V}{2} = \frac{V * (V-1)}{2} \tag{2.2}$$

- Planarer Graph: Wenn sich Graph in Ebene abbilden lässt, ohne dass sich zwei Kanten überschneiden.

- Bewerteter Graph: Wenn Kanten des Graphen eine Bewertung aufweisen. Mögliche Bewertungen sind die Länge der Kante (im Sinne der L2-Norm) oder die benötigte Zeit um Kante mit bestimmter Geschwindigkeit zu passieren. Man kann auch die Wahrscheinlichkeit einer Kantenbenutzung als Kostenfaktor ansetzen (sieheGleichung (2.3)).

$$c_{ij} = -log(p_{ij}) \tag{2.3}$$

c_{ij} ... positiver Kostenwert: $(0 \leq c_{ij} \leq \infty)$
p_{ij} ... Wahrscheinlichkeit: $(1 \geq c_{ij} \geq 0)$

Im Nachfolgenden sind ausgewählte Grundbegriffe der Graphentheorie aufgelistet:

- adjazent: Sind zwei Knoten durch eine Kante verbunden, so sind diese Knoten adjazent (benachbart).
- inzident: Beginnen oder enden zwei Kanten in einem Knoten, so spricht man von zwei inzidenten Kanten.
- Weg: Unter einem Weg versteht man eine geordnete Folge von Kanten, die von einem Knoten v_1 zu einem Knoten v_2 führt.
- Zyklus: Ein Zyklus ist ein Weg, in dem die Knoten in Kreisstruktur verbunden sind.
- Baum: Ein zusammenhängender kreisfreier Graph wird als Baum bezeichnet.
- Knotengrad: Anzahl der inzidierten Kanten im Knoten.
- Kosten: Attribut der Kante, z.B. Distanz oder Zeit.
- Bogen: Eine Kante mit aufgeprägter Richtung.

2.5.3 Speicherung von Graphen

Zwei wichtige und weitverbreitete Datenstrukturen zur Speicherung und Weiterverarbeitung von Graphen im Computer sind Adjazenzliste und Adjazenzmatrix.

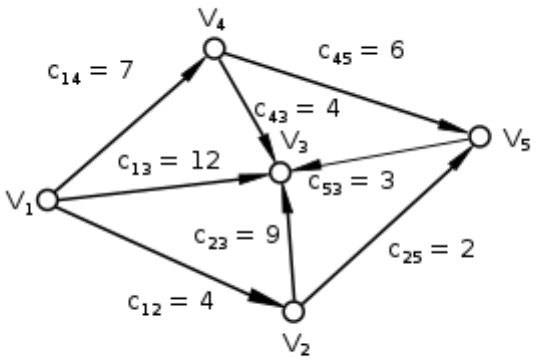


Abbildung 2.8: Graph mit bewerteten Bögen

Adjazenzlisten

Adjazenzlisten gehören zur Gruppe der verketteten Listen. Nachfolgend wird auf die einfach verkettete und auf die mehrfach verkettete Liste eingegangen.

Einfach verkettete Liste

Wird ein Graph in einer einfach verketteten Liste gespeichert, sind gesamt drei Listen (Vorgängerliste, Nachfolgerliste und Kostenliste) zur Verarbeitung nötig. Tabelle 2.13, Tabelle 2.14 und Tabelle 2.15 beziehen sich auf den Graphen in Abbildung 2.8 .

- Vorgängerknotenliste: beinhaltet die Startknoten der vorhandenen Kanten

Tabelle 2.13: Vorgängerknotenliste

Bogen	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Anfangsknoten	**1**	**1**	**1**	**2**	**2**	**4**	**4**	**5**

- Nachfolgerknotenliste: beinhaltet die Endknoten der vorhandenen Kanten

Tabelle 2.14: Nachfolgerknotenliste

Bogen	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Endknoten	**2**	**3**	**4**	**3**	**5**	**3**	**5**	**3**

- Kostenliste: enthält die Kantenbewertung der verwendeten Kanten

Tabelle 2.15: Bogenliste

Bogen	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Endknoten	**4**	**12**	**7**	**9**	**2**	**4**	**6**	**3**

Jede Liste hat so viele Einträge, wie der Graph Kanten hat.

Der Vorteil solcher Listen liegt am geringen Speicherbedarf und an der Einfachheit, weitere Daten anzufügen. Nachteilig ist die Suche nach einem bestimmten Attribut, da über jedes einzelne Element der Liste iteriert werden muss. Der Speicheraufwand beläuft sich auf das dreifache der Kantenanzahl ($3 * E$).

Mehrfach verkettete Liste

Dieser Listentyp enthält im Vergleich zum zuvor erläuterten Typen einen Zeiger (engl. Pointer) auf das vorangegangene Element und einen Zeiger auf das nachfolgende Element. Dadurch ist eine Suche vom Anfang der Liste und auch vom Ende der Liste nach einem Eintrag möglich. Für das erste Listenelement zeigt der Vorgängerzeiger auf den Wert „NULL“. Gleiches gilt für den Nachfolgerzeiger des letzten Elements. Im Vergleich zur einfach verketteten Liste ist hier aufgrund der Zeiger mehr Speicherplatz erforderlich. Durch das Wissen des Vorgänger- und Nachfolgerelementes ist ein schnelleres Löschen und Hinzufügen von Objekten möglich. In der Praxis findet dieser Listentyp unter dem Namen indizierte Adjazenzliste Anwendung. Im Vergleich zur einfach verketteten Liste werden für das Speichern eines Graphen nur zwei Listen, eine Knotenliste und eine Bogenliste verwendet. Tabelle 2.16 und Tabelle 2.17 beziehen sich auf den Graphen in Abbildung 2.8.

- Knotenliste: Die Knotenliste hat n+1 Einträge, wobei n die Anzahl der im Graph vorhandenen Knoten repräsentiert

Tabelle 2.16: Knotenliste

Knoten	**1**	**2**	**3**	**4**	**5**	**6**
Index	**1**	**4**	**6**	**6**	**8**	**9**

Der Index für den 1. Knoten ist immer 1. Um nachfolgende Indizes zu berechnen, muss zum Vorgängerindex die Ausgangsvalenz des aktuellen Knotens addiert werden.

- Bogenliste: Die Bogenliste hat so viele Einträge, wie der Graph Bögen hat. Sie beinhalten für jeden Bogen den Zielknoten und die Kostenbewertung.

Tabelle 2.17: Bogenliste

Bogen	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Zielknoten	**2**	**3**	**4**	**3**	**5**	**3**	**5**	**3**
Bewertung	**4**	**12**	**7**	**9**	**2**	**4**	**6**	**3**

Will man nun beispielsweise vom Knoten v_2 die Zielknoten und die Kosten der Bögen wissen, so sucht man in der Knotenliste nach der entsprechenden Knotennummer. Bildet man

die Differenz des nachfolgenden Index mit dem Aktuellen, so erhält man die Ausgangsvalenz des gesuchten Knotens. Um nun die dazugehörigen Zielknoten und Bewertungen zu bekommen, sucht man in der Bogenliste nach der Bogennummer, die dem Index des zu untersuchenden Knotens der Knotenliste entspricht. Ist die Ausgangsvalenz des Knotens 1, so werden die gesuchten Informationen (Zielknoten und Bewertung) den darunterliegenden Zeilen entnommen. Um auch die weiteren Zielknoten und Bewertungen bei einer Ausgangsvalenz > 1 zu bekommen, werden diese rechts der Ausgangsspalte entnommen (siehe Tabelle 2.17). Im Vergleich zur einfach verketteten Liste, reduziert sich der Speicheraufwand von $3 * E$ auf $2 * E + V + 1$.

Adjazenzmatrix

Die Adjazenzmatrix eines Graphen $G(V, E)$ ist als $A = (a_{ij})$ mit $i, j \; \varepsilon \; V$ definiert und es lässt sich die lineare Algebra auf diese Matrix anwenden. $Dim(A) = (n \times m)$, wobei n die Anzahl der Zeilen und m die Anzahl der Spalten darstellt. Das Element a_{ij} ist 1, wenn es im Graphen eine Kanten zwischen den Konten v_i und dem Knoten v_j gibt. Bei einem ungerichteten Graphen ist die Adjazenzmatrix symmetrisch.

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{array} \begin{array}{c} \begin{array}{ccccc} V_1 & V_2 & V_3 & V_4 & V_5 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \tag{2.4}$$

Die unter 2.4 dargestellte Adjazenzmatrix setzt sich aus dem in Abbildung 2.8 dargestellten Graphen zusammen. Da es sich um einen gerichteten Graphen handelt und folglich nicht jede Kante unbedingt in beide Richtungen befahrbar ist, ist diese Matrix nicht symmetrisch. Um mit dieser asymmetrischen Adjazenzmatrix zu arbeiten oder den Graphen zu rekonstruieren, muss zuvor die Ablesereihenfolge der Start- und Zielknoten der jeweiligen Bögen festgelegt werden. Definiert man die Startknoten als zeilenweise und die Zielknoten als spaltenweise angeordnet, so kommt man zu dem unter Abbildung 2.8 dargestellten Graphen. Transponiert man die Adjazenzmatrix und lässt die Ablesereihenfolge unverändert, so würden sich alle Bögen umkehren. Der Speicherbedarf einer Adjazenzmatrix ist gleich der Anzahl der Matrixeinträge und errechnet sich aus V^2.

Im Falle eines Graphen mit geringer Anzahl an Kanten, im Verhältnis zu den Knoten, benötigt die Adjazenzliste deutlich weniger Speicherplatz, als die Adjazenzmatrix. Eine Endscheidung darüber, ob eine Adjazenzliste einer Adjazenzmatrix vorzuziehen ist, lässt sich laut Hofmann-Wellenhof et al., 2011, Seite 90 treffen, wenn die Ungleichung in 2.5 erfüllt ist.

$$E < \frac{V^2}{2} \tag{2.5}$$

Möchte man prüfen, ob eine Verbindung zwischen dem Knoten v_i und v_j besteht, so muss lediglich in der Adjazenzmatrix der Wert an der Stelle $A = (a_{ij})$ auf 1 geprüft werden. Dies lässt sich in der konstanten Zeit $O\{1\}$ durchführen. Um die Laufzeit einer Adjazenzliste mit

einer Adjazenzmatrix gegenüberzustellen, ist der Aufwand zu berechnen, der benötigt wird, alle benachbarten Knoten eines beliebigen Knoten v_j zu durchlaufen. Realisiert wird dies mit Hilfe der Landau-Symbole (siehe Steger, 2013, Seite 10 ff). Im Falle der Adjazenzliste muss nur die Subliste des betroffenen Knotens durchlaufen werden. Der Zeitbedarf ist $O\{Ausgangsvalenz(v_i)\}$ (siehe Steger, 2013, Seite 65). Liegt der Graph in Form einer Adjazenzmatrix vor, so muss Zeile$_j$ oder Spalte$_j$ komplett durchlaufen und auf 1 geprüft werden. Der zeitliche Aufwand errechnet sich aus $O\{|V|\}$ (siehe Steger, 2013, Seite 65).

2.6 Referenzsysteme

Referenzsysteme bilden die physikalische und mathematische Grundlage, um die Position eines Objekts im Raum eindeutig zu beschreiben. Beispielsweise beschreibt ein Tripel von Zahlen (Koordinaten) einen 3-dimensionalen Punkt im Raum. Referenzsysteme gewährleisten eine konsistente Repräsentation dieser Koordinaten, die im Allgemeinen zeitabhängig sind. Dafür benötigt man einen Ursprung und eine Orientierung der drei orthogonalen Achsen. Man unterscheidet zwischen Referenzsystem und Referenzrahmen. Unter Referenzsystem versteht man die theoretische Definition. Der dazugehörige Referenzrahmen ist die praktische Realisierung dessen. Für die Definition eines Bezugssystems benötigt man einen Koordinatenursprung und die Orientierung der Achsen. Referenzsysteme werden in

- Inertialsystem
- Quasi-Inertialsystem
- Nichtlineares System

unterteilt.

2.6.1 Inertialsystem

Ein Inertialsystem ist ein System, das sich in Ruhe befindet oder einer gleichförmigen Bewegung unterliegt. In diesem System gelten die Gesetzte der Newtonschen Mechanik.

2.6.2 Quasi-Inertialsystem

Unter einem Quasi-Inertialsystem versteht man ein System, das bezüglich der Rotation inertial, jedoch nicht-inertial, bezüglich der Translation ist. Das geozentrische raumfeste Äquatorsystem ist ein Quasi-Inertialsystem. Um Satellitenbahnen beschreiben zu können, benötigt man ein raumfestes Koordinatensystem, da die Satellitenumlaufbahnen unabhängig von der Erdrotation im Raum liegen. Dieses Koordinatensystem ist mit dem Fixsternhimmel verbunden. Der Ursprung dieses Systems ist das Geozentrum. Die X_1-Achse zeigt in Richtung des Frühlingspunktes (Schnittpunkt der Äquatorebene mit der Ekliptikebene) und die X_3-Achse ist die mittlere Erdrotationsachse (siehe Abbildung 2.9). Es handelt sich hierbei um ein Rechtssystem, die Achsen stellen vektorielle Größen dar und stehen orthogonal aufeinander. Deshalb errechnet sich die X_2-Achse aus dem Kreuzprodukt $X_3 \times X_1$.

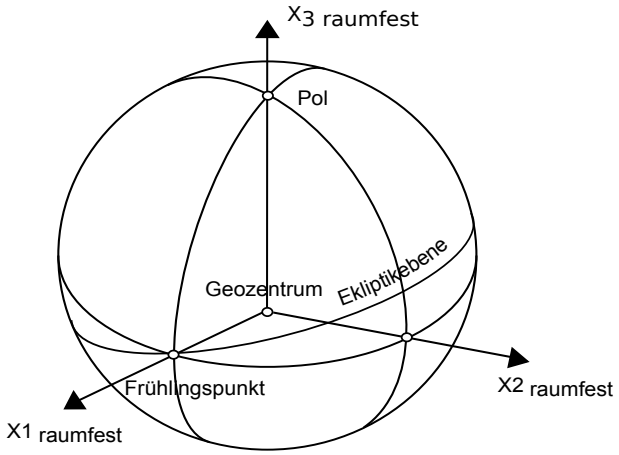


Abbildung 2.9: Quasi-Inertialsystem

2.6.3 Nichtlineares System

Ein nichtlineares System rotiert um mindestens eine seiner Achsen. Streng genommen gelten in diesen Systemen die Gesetze der Newtonschen Mechanik nicht. Jedoch stellen sie eine durchaus gute Näherung dar. Eine Realisierung dieses Systems ist das erdfestes Koordinatensystem (engl. earth centered earth fixed). Um einen Punkt auf der Erdoberfläche oder im erdnahen Raum zu beschreiben, bedient man sich dieser erdfesten Koordinatensysteme. Eine mögliche Realisierung eines erdfesten Koordinatensystems besteht darin, die X_1-Achse durch den Schnittpunkt von Äquator mit dem Greenwich Meridian verlaufen zu lassen. Die X_3-Achse ist parallel zur mittleren Erdachse. Die X_2-Achse berechnet sich gleich, wie beim zuvor erwähnten Quasi-Inertialsystem, aus $X_3 \times X_1$ (aufgrund der Übersichtlichkeit in Abbildung 2.10 wurde die X_2-Achse aber nicht eingezeichnet). Befindet sich der Ursprung des Koordinatensystems im Erdmittelpunkt, so handelt es sich um ein globalem Koordinatensystem (siehe Abbildung 2.10).

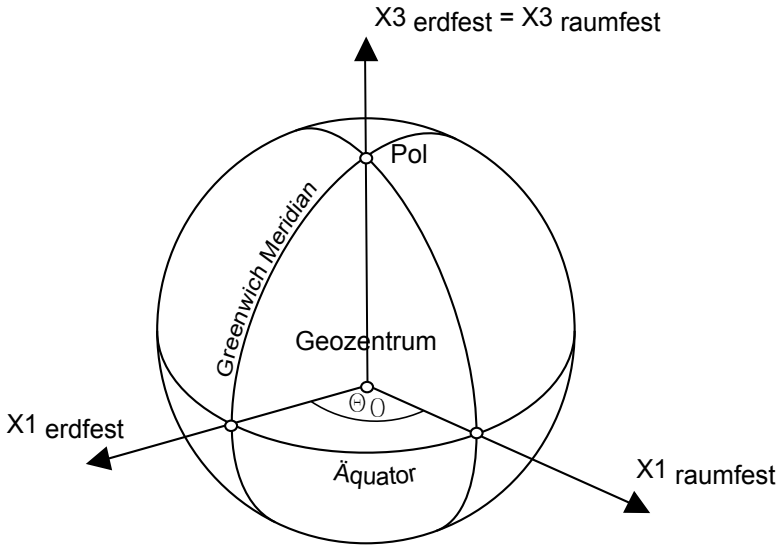


Abbildung 2.10: Nichtlineares System

Das World Geodetic System 84 (WGS84) ist ein globales Referenzsystem und dient als Grundlage zur weltweiten Beschreibung von Punkten auf der Erde und im erdnahen Raum. Der Winkel zwischen der X_1-Achse des raumfesten Systems und der X_1-Achse des erdfesten Systems, wird als Sternzeit Θ_0 bezeichnet (siehe Abbildung 2.10).

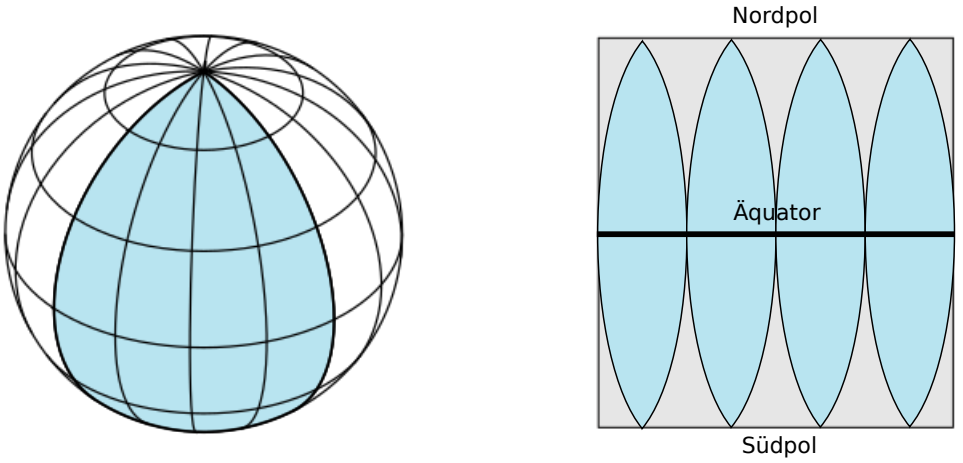
2.7 Bezugsfläche und Erdfigur

Ein Geoid ist ein physikalisches Modell der Erdfigur. Es stellt die Niveaufläche der Erde ohne den Einfluss der Gezeiten oder von Luftdruckschwankungen dar. Als ausgezeichnete Niveaufläche wählt man die mittlere Meeresoberfläche. Durch die unterschiedliche Massenverteilung, z.B. Erzlager- oder Salzlagerstätten, auf der Erde bekommt das Geoid im Erdmantel Beulen und Dellen. Aufgrund dieser Beulen und Dellen eignet sich das Geoid nicht als Bezugsfläche. Stattdessen wird in erster Näherung eine Kugel und in zweiter Näherung ein Rotationsellipsoid verwendet. Für Navigationsanwendungen ist eine Kugel als Annäherung an die Erdfigur ausreichend. Ist die Genauigkeitsanforderung jedoch hoch, wie beispielsweise in der Ingenieurvermessung, so wird auf ein Rotationsellipsoid zurückgegriffen. Rotationsellipsoide werden in lokale und globale Ellipsoide unterschieden. Für die in Österreich gebräuchlichen Gauß-Krüger-Koordinaten wird das lokale Bessel-Ellipsoid verwendet. Der Unterschied von lokalem zu globalem Ellipsoid besteht an der abzubildenden Erdoberfläche. Will man nur ein Teilgebiet der Erde abbbilden, so wählt man ein lokales, an das Teilgebiet bestangepasste Ellipsoid.

2.8 Kartenabbildungen

Das Abbilden einer Kugel in die Ebene lässt sich recht gut mit dem Schälen einer Orange vergleichen. Stellt man sich für die Illustration nur eine Hälfte der Orange ohne Frucht-

fleisch, die senkrecht von oben nach unten durchgeschnitten ist, vor und versucht diese an einer Tischoberfläche flach zudrücken, so wird die gekrümmte Schale unweigerlich einreißen. Um dem Einreißen zuvorzukommen, könnte man die Schale in mehrere Zonen einteilen und sie von oben sowie von unten bis zur Mitte einschneiden. Abbildung 2.11 gibt diese Erklärung wieder. Um wieder auf die Erde zurückzukommen entspricht Abbildung 2.11(b) einer verebneten Darstellung unseres Planeten.



(a) Kugel (b) Ebene

Abbildung 2.11: Verzerrung

Da die Verarbeitung von geraden Formen für das menschliche Gehirn wesentlich einfacher ist, als die von gekrümmten Formen, bedient man sich geometrischer und mathematischer Abbildungen, um die ansonsten grauen Flächen (siehe Abbildung 2.11(b)) zu füllen und somit ein einheitliches und leicht interpretierbares Abbild zu schaffen. Die dadurch entstehenden Verzerrungen der ebenen Darstellung werden hierfür in Kauf genommen.

Abbildungen werden nach Abbildungsflächen und Verzerrungseigenschaften klassifiziert.

2.8.1 Abbildungsflächen

Um die gekrümmmte Oberfläche zu verebnen, bedient man sich unterschiedlicher Hilfslächen. Diese sind Ebene, Kegel und Zylinder.

Azimutalabbildung

Mittels einer Ebene lässt sich eine azimutale Abbildung herbei führen (siehe Abbildung 2.12). Dabei berührt die Ebene die Kugel oder das Ellipsoid an einem Punkt P. Zum Einsatz kommt diese Abbildung bei kreisförmigen oder kreisähnlichen Gebieten. Beispielsweise werden die Polregionen mit dieser Abbildung dargestellt. Ein Vertreter dieser Abbildungsart ist die stereographische Abbildung.

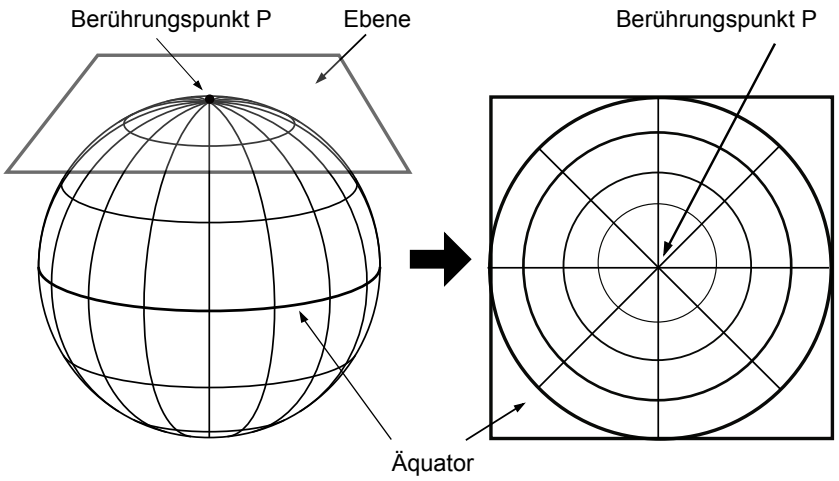


Abbildung 2.12: Azimutalprojektion

Die Verzerrungen im Berührungspunkt bzw. in unmittelbarer Nähe sind sehr gering. In der Verebnung bilden die Meridiane ein vom Pol ausgehendes Strahlenbündel. Dabei sind die Breitenkreise (Parallelkreise) konzentrisch um den Pol angeordnet. Breitenkreise und Meridiane schneiden sich im rechten Winkel.

Kegelabbildung

Bei einer Kegelabbildung wird die Kugel oder das Ellipsoid zuerst auf einen schneidenden oder berührenden Kegel abgebildet und im Anschluss in die Ebene abgewickelt (siehe Abbildung 2.13). Der berührende bzw. die schneidenden Parallelkreise werden dabei längentreu (siehe Abschnitt 2.8.2) in die Ebene abgebildet. Zum Einsatz kommt diese Abbildung u.a. bei Ländern mit Ost-West-Ausdehnung.

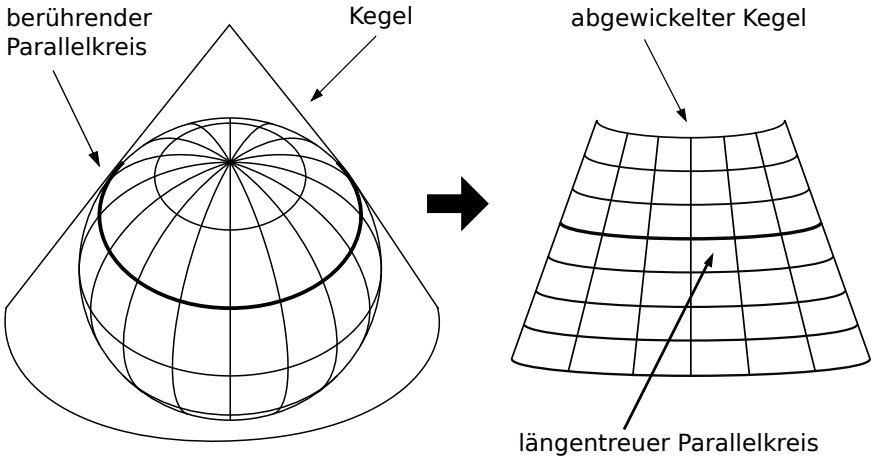


Abbildung 2.13: Kegelprojektion

Die Meridiane bilden ein Strahlenbündel und die Breitenkreise werden als konzentrische Teilkreissegmente dargestellt.

Zylinderabbildung

Bei der Zylinderabbildung wird die Kugel oder das Ellipsoid zuerst auf einen schneidenden oder berührenden Zylinder abgebildet und im Anschluss abgewickelt. Die bekannteste Zylinderabbildung ist die Mercatorprojektion. Hierbei berührt der Zylinder den Äquator und dieser wird längentreu in die Ebene abgebildet. Bei der in Österreich verwendeteten Gauß-Krüger-Abbildung berührt der Zylinder die Kugel entlang eines Grund- oder Hauptmeridians. Die Universal Transvere Mercator (UTM) Abbildung ist eine Sonderform der Gauss-Krüger-Abbildung. Der Grund- oder Hauptmeridians wird mit dem Faktor 0.99996 verkürzt in die Ebene abgebildet. Dies wird deshalb gemacht, da die Meridianstreifen mit 6° doppelt so breit sind, wie bei der Gauß-Krüger-Abbildung. Deshalb wird die Verzerrungen zum Rand etwas verringert.

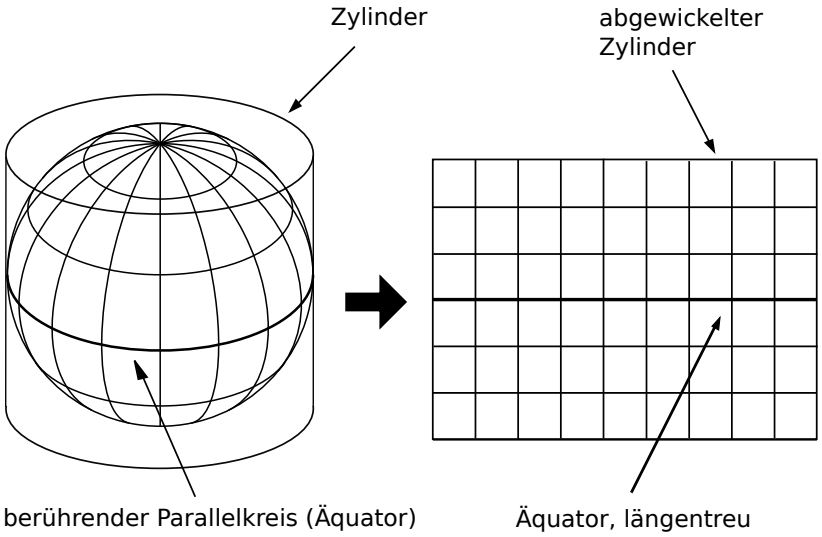


Abbildung 2.14: Zylinderprojektion

Bei der Zylinderabbildung werden Meridiane und Parallelkreise als Geraden abgebildet. Dadurch scheiden sie sich immer in einem rechten Winkel.

Weiterführende Literatur, betreffend der zuvor erwähnte Abbildungen, ist unter Iliffe J., 2008 zu finden.

2.8.2 Verzerrungseigenschaften

Wie bereits oben erwähnt, treten bei der Abbildung der Erde in die Ebene unvermeidbare Verzerrungen auf. Mittels der Tissot'schen Indikatrix können die Verzerrungseigenschaften von Kartenentwürfen überprüft werden. Man unterscheidet zwischen winkeltreuen, flächentreuen und längentreuen Verzerrungen.

Längentreu

Unter Längentreue verstehen man, dass eine Strecke auf der Erde unter Berücksichtigung des Maßstabs exakt einer Strecke in der Karte entspricht. Dies ist jedoch nur für gewissen Linien und Richtungen machbar. Beispielsweise wird bei der Zylinderabbildung, sofern der Äquator als berührender Parallelkreis gewählt wird, dieser längentreu in die Ebene abgebildet.

Flächentreu

In flächentreuen Abbildungen wird die Erde unter Berücksichtigung des Maßstabs in korrekten Größendarstellungen wiedergegeben. Die Flächen sind zwar exakt, jedoch entspricht die Form der Flächen nicht der Realität.

Winkeltreu

Winkeltreu, auch konform genannt, bedeutet dass ein Winkel in der Realität auch dem Winkel in der Karte entspricht. Besondere Bedeutung kommen winkeltreue Abbildungen in der Luft- und Seefahrt zu. So lässt sich mit diesen Karten einfach ein Kurs mit konstantem Winkel (auch Loxodrome genannt) steuern.

Der Vollständigkeit halber soll erwähnt werden, dass man Abbildungen auch nach der Lage der Hilfsfläche in normal, transversal und schiefachsig unterteilen kann.

2.9 Routenplanung

Unter Routenplanung (engl. Routing) versteht man die günstigste Route zwischen einem Start- und einem Zielpunkt zu bekommen. Mit „günstig“ kann die schnellste, kürzeste oder auch beispielsweise die schönste Route gemeint sein. Realisiert wird dies über eine entsprechende Kostenfunktion und Kantengewichte des Graphen (siehe Abschnitt 2.5.2). Eine Steigerung der Routenplanung ist die Tourenplanung. Hierbei soll die Tour (Rundreise) über fix vorgegebene Zwischenpunkte verlaufen. Als Forderung stehen die minimalen Gesamtkosten sowie auch die bestmögliche Reihenfolge der Zwischenpunkte im Raum. In der Literatur wird diese Aufgabenstellung auch das Problem des Handelsreisenden (engl. Traveling Salesman Problem, kurz TSP) genannt. Die Routenplanung beantwortet die Frage „Wo muss ich hin?“ und passiert zumeist vor Fahrtantritt.

2.10 Zielführung

Die Zielführung (engl. Guidance) beschreibt den Weg während der Fahrt zum Ziel. Es wird eine sich aus der Routenplanung und aus der aktuellen Position abgeleitete Manöverliste, Schritt für Schritt abgearbeitet. Die Bekanntgabe des Manövers erfolgt akustisch und/oder visuell und ist abhängig von der Geschwindigkeit. „In 100 m rechts abbiegen“ könnte ein Manöver innerhalb eines Ortsgebietes lauten. Auf einer Autobahn muss das Manöver der Geschwindigkeit entsprechend früher bekanntgegeben werden, um noch rechtzeitig Reagieren zu können. Die Zielführung liefert die Antwort auf die Frage „Was muss ich als nächstes tun, um an mein Ziel zu kommen?“.

2.11 Routing-Algorithmen

Um kürzeste Wege in einem Graphen zu finden, bedient man sich der Hilfe von Suchalgorithmen, die eine Lösung für verschiedene Varianten des „Problemes des kürzesten Weges“ (engl. Shortest Path Problem) bieten. Ausgehend von einem positiv bewerteten Netzwerk (siehe Gleichung (2.3)) gelten die Bedingungen nach Gleichung (2.6).

$$\sum_{Pfad} c_{ij} \stackrel{!}{=} \min \quad \rightleftarrows \quad \prod_{Pfad} p_{ij} \stackrel{!}{=} \max \tag{2.6}$$

Im Nachfolgenden sind unterschiedliche Varianten für das „Problemes des kürzesten Weges“ aufgelistet:

- Single Pair Shortest Path: Hierbei wird die kürzeste Verbindung von einem Ort A zu einem Ort B ermittelt.
- Sequential Shortest Path: Es werden Knoten vorgegeben, die in einer bestimmten Reihenfolge abzuarbeiten sind.
- Single Source Shortest Path: Hierbei werden die kürzesten Wege von einem Startknoten aus zu allen übrigen Knoten des Graphen ermittelt.
- Single Destination Shortest Path: Hierbei wird der kürzeste Weg zwischen einem Endknoten und allen übrigen Knoten des Graphen ermittelt.

Routing Algorithmen arbeiten u.a. nach Breiten- und Tiefensuche sowie nach Label-setting und Label-correcting Verfahren .

Breitensuche

Die Breitensuche (engl. breadth-first search) ist ein Verfahren zum Durchlaufen aller Knoten eines Graphen. Es werden zu Beginn alle Knoten besucht, die vom Startknoten direkt erreichbar sind, danach werden die Knoten durchlaufen, welche 2 Kanten vom Startpunkt entfernt sind usw. Eine mögliche Reihenfolge der besuchten Knoten ist in Abbildung 2.15 aufgezeigt.

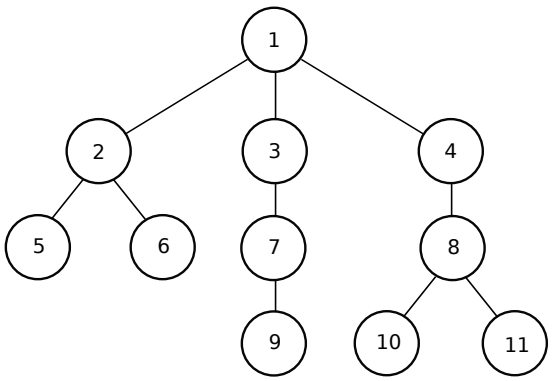


Abbildung 2.15: Breitensuche

In Abbildung 2.15 ist das „in die Breite gehen“ des Suchbaumes ersichtlich.

Tiefensuche

Bei der Tiefensuche (engl. depth-first search) handelt es sich ebenfalls um ein Verfahren zum Durchlaufen aller Knoten eines Netzwerks. Wie bei der Breitesuche beginnt die Tiefensuche mit einem adjazenten Knoten des Startpunktes. Ab nun, wie das Wort Tiefsuche besagt, wird als nächster Knoten aber nicht der nächstbenachbarte Knoten vom Ausgangspunkt gewählt, sondern der nächstgelegene des zuvor besuchten Knotens. Dadurch breitet sich der Suchbaum zuerst in die Tiefe aus. Existiert kein unbesuchter benachbarter Knoten mehr, so wird solange im vorhandenen Suchbaum zurückgegangen, bis es wieder einen unbesuchten Nachbarsknoten gibt.

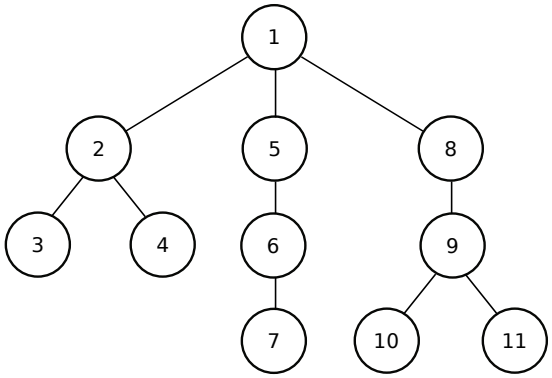


Abbildung 2.16: Tiefensuche

Abbildung 2.16 zeigt eine mögliche Reihenfolge der besuchten Knoten bei der Tiefensuche.

Label-setting und Label-correcting

Unter dem Label eines Knotens v_j versteht man die Erreichungskosten, ausgehend von einem Startknoten v_s, im Sinne der kürzesten Verbindung. Erreicht der Suchbaum bei seiner Ausbreitung einen Knoten v_j, so sind seine Erreichungskosten vom Startknoten aus bekannt und werden nicht mehr verändert. In diesem Fall handelt es sich um das Label-setting Verfahren, welches beim Dijkstra-Algorithmus (siehe Abschnitt 2.11.1) zum Einsatz kommt. Beim Label-correcting Verfahren kann sich das Label eines Knotens v_j auch noch dann ändern, wenn er bereits bei der Ausbreitung des Suchbaums erreicht wurde. Der optimale Weg von einem Start- zu einem Endknoten ist erst nach Vollständigem Durchlaufen des Netzwerks bekannt. Zum Einsatz kommt dieses Verfahren unter anderem beim A*-Algorithmus (siehe Abschnitt 2.11.2).

2.11.1 Dijkstra-Algorithmus

Der Dijkstra-Algorithmus wurde 1959 von Edsger Dijkstra veröffentlicht Dijkstra, 1959. Dieser Algorithmus dient der Berechnung kürzester Wege in einem Knoten-Kanten-Netzwerk. Es lässt sich damit die kürzeste Verbindung zwischen einem Start- und einem Endknoten berechnen (single pair shortest path) und andererseits auch die kürzesten Verbindungen zwischen einem Startknoten und allen übrigen Knoten eines Graphen (all pair shortest path). Der Algorithmus arbeitet mit dem Prinzip der Breitensuche (Abschnitt 2.11) und nach dem Label-setting-Verfahren. Der Suchbaum breitet sich kreisförmig rund um den Startpunkt aus.

Algorithmus 1 – Dijkstra

```
1:  l_s = 0
2:  T ← {v_s}
3:  P ← { }
4:  for v_j ∈ V\{v_s} do
5:      l_j = ∞
6:  end for
7:  while T ≠ { } do
8:      v_i = v_j ∈ T mit min(l_j)
9:      P ← P + {v_i}
10:     T ← T\{v_i}
11:     for v_j ∈ N(v_i) do
12:         if (v_j ∉ T) ∧ (v_j ∉ P) then
13:             l_j = l_i + c_ij
14:             p_j = v_i
15:             T ← T + {v_j}
16:         end if
17:         if (v_j ∈ T) ∧ (l_i + c_ij < l_j) then
18:             l_j = l_i + c_ij
19:             p_j = v_i
20:         end if
21:     end for
22: end while
```

In Algorithmus 1 ist der schematische Ablauf des Dijkstra-Algorithmus mittels Pseudocode dargestellt und dieser wurde dem Pseudocode aus Hofmann-Wellenhof et al., 2011, Algorithm 14.1 nachempfunden. Im Nachfolgenden wird seine Funktion erläutert:

In den Zeilen 1 bis 3 wird zuerst ein Startpunkt v_s gewählt und dessen Label l_s auf null gesetzt. Anschließend wird der Startknoten in die Liste T, der temporär vergebenen Label, aufgenommen. Zu diesem Zeitpunkt darf die Liste P, der permanent gesetzten Label, noch keinen Knoten beinhalten. In den Zeilen 4 bis 6 werden mittels einer Schleife alle übrigen Knoten des zu behandelten Netzwerks, durchlaufen und deren Labels auf unendlich gesetzt. Die Zeilen 8 bis 21 werden in einer Schleife solange wiederholt, bis alle Knoten ein permanentes Label zugewiesen bekommen haben. Der als erstes zu betrachtende Knoten v_i wird aus der Menge T, mit minimalen Kosten l_i, ausgewählt (Zeile 8). Dieser Knoten bekommt ein permanentes Label, befindet sich ab nun in der Menge P und wird zugleich aus der Menge T entfernt (Zeile 9 bis 10). Die Zeilen 11 bis 20 werden für alle adjazenten Knoten von v_i durchgeführt. Sollte der zu untersuchende Knoten noch nie besucht worden sein, folglich also über kein temporäres oder permanentes Label verfügen (Zeile 12), so setzt sich sein Label l_j aus der Summe des Labels l_i und der Kantenbewertung c_{ij}, also aus der Verbindung der beiden Knoten, zusammen. Weiters wird der Knoten, der in Zeile 9 ein permanentes Label bekommen hat, als Vorgängerknoten, für den aktuell zu untersuchten Knoten gespeichert und der untersuchte Knoten v_j wird in die Menge T, die Knoten mit temporärem Label, aufgenommen (Zeile 12 bis 15). Für den Fall, dass der aktuell zu untersuchende Knoten bereits in der Menge T enthalten ist und das neu berechnete Label,

bestehend aus der Summe des Labels l_i, des besetzten Knotens v_i plus Kantenbewertung c_{ij}, kleiner ist als das bereits berechnete Label l_j (Zeile 17), so wird das Label l_j durch das zuvor ermittelte kleinere Label ersetzt und der besetzte Knoten v_i als Vorgängerknoten für den aktuell untersuchten Knoten gespeichert (Zeile 18 und 19).

Gewinnung der Route

Um nach Erreichen des vorgegebenen Zielknotens v_z die optimale (kürzeste, schnellste, ...) Knotenfolge von v_s zu v_z zu bekommen, betrachtet man den zugehörigen Vorgängerknoten p_z. Dieser wiederum weißt auf dessen Vorgängerknoten hin. Wird dieser Vorgang solange wiederholt bis p_j auf den Startknoten v_s verweist, so liegt die Knotenreihenfolge von Ziel- zu Startknoten vor. Kehrt man die Reihenfolge um, so weist die Knotenreihenfolge den Weg vom Start zum Ziel.

2.11.2 A*-Algorithmus

Der A*-Algorithmus gehört zur Gruppe der heuristischen (informierten) Suchalgorithmen. Mit heuristischen Algorithmen wird versucht, menschliche Problemlösungskompetenzen zu modellieren. Zum Einsatz kommen diese Algorithmen meist dort, wo ein nichtinfomierter Algorithmus zu rechenintensiv ist. Hierbei wird auch eine suboptimale Lösung in Kauf genommen, wenn sich dadurch die Rechenzeit deutlich reduzieren lässt. Als Ausgangsbasis des A*-Algorithmus dient der zuvor behandelte Dijkstra-Algorithmus (siehe Abschnitt 2.11.1). Mit Hilfe einer heuristischen Funktion wird versucht, den Suchbaum in seiner Ausbreitung einzuschränken und den optimal kürzesten Weg zum Ziel zu finden. Eine mögliche Umsetzung dieses heuristischen Ansatzes ist eine gemischte Bewertungsfunktion $f(v_j)$. Diese besteht aus den tatsächlichen Kosten $l(v_j)$ vom Startpunkt v_s zu dem aktuellen Punkt v_j und den vermuteten Kosten $h(v_j)$ vom Knoten v_j zum Zielpunkt v_z (siehe Gleichung (2.7)).

$$f(v_j) = l(v_j) + h(v_j) \tag{2.7}$$

Für die vermuteten Kosten $h(v_j)$ kann man die Luftlinie, eine Gerade von v_s zum Zielpunkt v_z, ansetzen. Um dies zu realisieren braucht man neben einem bewerteten Netzwerk auch die geographischen Koordinaten aller Knoten. Aufgrund der Heuristik arbeitet der A*-Algorithmus nach dem Label-correcting Verfahren Abschnitt 2.11. Der Suchbaum breitet sich ellipsenförmig Richtung Zielknoten aus.

Algorithmus 2 – A*

```
1:  l_s = 0
2:  f_s = h_s                                                    ⋆
3:  T ← {v_s}
4:  P ← { }
5:  for v_j ∈ V\{v_s} do
6:      l_j = ∞
7:      f_j = ∞                                                  ⋆
8:  end for
9:  while T ≠ { } do
10:     v_i = v_j ∈ T mit min(f_j)                               ⋆
11:     P ← P + {v_i}
12:     T ← T\{v_i}
13:     for v_j ∈ N(v_i) do
14:         if (v_j ∉ T) ∧ (v_j ∉ P) then
15:             l_j = l_i + c_ij
16:             p_j = v_i
17:             f_j = l_j + h_j                                  ⋆
18:             T ← T + {v_j}
19:         end if
20:         if (v_j ∈ T) ∧ (l_i + c_ij < l_j) then
21:             l_j = l_i + c_ij
22:             p_j = v_i
23:             f_j = l_j + h_j                                  ⋆
24:         end if
25:         if (v_j ∈ P) ∧ (l_i + c_ij < l_j) then               ⋆
26:             l_j = l_i + c_ij                                 ⋆
27:             p_j = v_i                                        ⋆
28:             f_j = l_j + h_j                                  ⋆
29:             P ← P\{v_j}                                      ⋆
30:             T ← T + {v_j}                                    ⋆
31:         end if                                               ⋆
32:     end for
33: end while
```

In Algorithmus 2 ist der schematische Ablauf des A*-Algorithmus mittels Pseudocode dargestellt. Als Grundgerüst dient der zuvor behandelte Dijkstra-Algorithmus. Die mit ($\star$) gekennzeichneten Zeilen sind in Bezug auf Algorithmus 1 neu hinzugefügt und es wird nachfolgend näher darauf eingegangen.

Zu Beginn wird der Startknoten v_s festgelegt und die gemischte Bewertungsfunktion f_s des Startknotens der geometrischen Distanz von Start- zu Zielknoten v_z gleichgesetzt (Zeile 2). Weiters wird das Label l_s, des Startknotens auf null gesetzt und der Startknoten der Liste T, der Liste der Knoten mit temporärem Label, zugeordnet. Anschließend werden neben den Labels l_j, die Bewertungsfunktionen f_j aller Knoten des Netzwerks exklusive des Startknotens auf unendlich gesetzt (Zeile 5 bis 8). Die Zeilen 14 bis 32 werden in einer Schleife solange wiederholt, bis allen Knoten ein permanentes Label zugewiesen wurde. Im

Unterschied zur Zeile 8 Algorithmus 1 , wo der Knoten v_i mit dem kleinsten Label aus der Liste T gewählt wird, wird in Zeile 10 derjenige Knoten aus T gewählt, der die kleinste gemischte Bewertungsfunktion f_j aufweist. In den Zeilen 11 und 12 wird der aktuell zu untersuchende Knoten der Liste P, welche die Liste der Knoten mit permanentem Label ist, zugeordnet und aus T entfernt. In den Zeilen 13 bis 32 werden alle direkt benachbarten Knoten v_j von v_i untersucht. Zeile 14 bis 19 überprüft, ob diese Knoten weder in T noch in P enthalten sind. Wenn dem so ist, so wird das Label l_j und die Bewertungsfunktion f_j berechnet, sowie der Vorgängerknoten p_j ermittelt und der entsprechende Knoten aus T entfernt. In den Zeilen 20 bis 24 wird überprüft, ob die benachbarten Knoten in T enthalten und ihre aus $l_i + c_{ij}$ berechneten Labels kleiner sind, als die aktuellen Labels l_j. Trifft dies zu, so werden l_j und f_j neu berechnet und die Vorgängerknoten neu gesetzt. In den Zeilen 25 bis 31 verbirgt sich das Label-correcting Verfahren. Sollte einer der möglichen Nachbarknoten von v_i sich bereits in P befinden und sein aktuell berechnetes Label kleiner sein als das zuvor ermittelte, so wird dieses dadurch ersetzt. Zusätzlich wird der Vorgängerknoten neu gesetzt, die Bewertungsfunktion neu errechnet, sowie der zu untersuchende Knoten aus P entfernt und wieder T zugeteilt.

Um die Knotenreihenfolge vom Start- zum Zielknoten zu bekommen, bedient man sich dem gleichen Schema wie beim Dijkstra-Algorithmus (siehe Abschnitt 2.11.1).

3 Implementierung

Die Wahl der verwendeten Computerprogramme, Geodaten und Softwarewerkzeuge wurde den gesetzten Themenschwerpunkten - Open-Source und freie Geodaten - entsprechend getroffen.

Die erstellte graphische Benutzeroberfläche (engl. graphical user interface, kurz GUI) basiert auf QGIS. QGIS eignet sich aufgrund seiner Programmierschnittstelle besonders gut, da es sich dadurch individuell an die Aufgabenstellung anpassen lässt. Des Weiteren stellt QGIS von Haus aus Werkzeuge zum Verarbeiten von Vektor- und Rasterdaten zur Verfügung. Über das QGIS-GUI wird eine Verbindung zur räumlichen Datenbank (PostgreSQL mit PostGIS-Erweiterung) hergestellt. Die darin in Tabellenform enthaltenen Straßendaten entstammen der OSM (siehe Abschnitt 3.3) und werden über die Datenbankanbindung in das GUI geladen. Aufbauend auf diesen Straßendaten und mit der PostgreSQL Erweiterung pgRouting, können anschließend routing-relevante Operationen durchgeführt werden. In Abbildung 3.1 ist das Zusammenwirken der einzelnen Komponenten grafisch dargestellt.

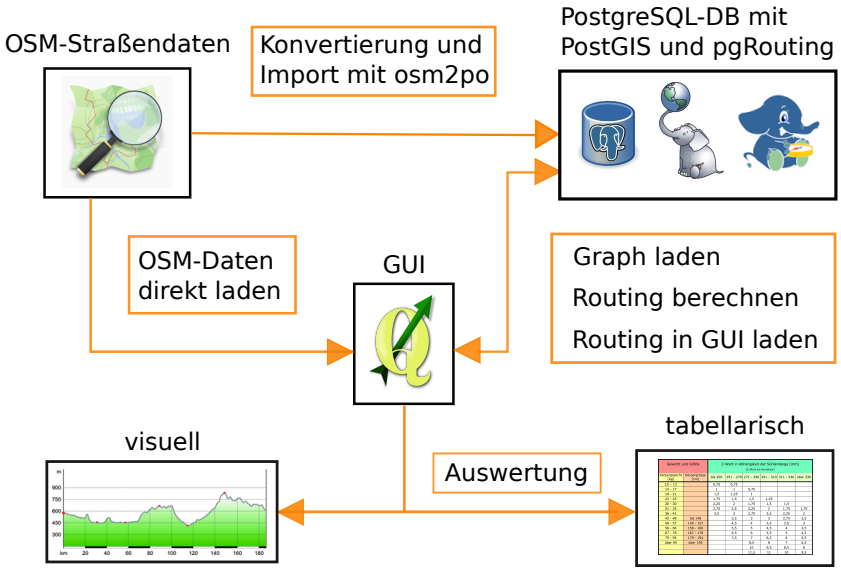


Abbildung 3.1: Zusammenwirken der Komponenten

3.1 QGIS

QGIS[1] ist ein freies GIS, welches in den Varianten Desktop-GIS, Server-GIS und mobile-GIS zur Verfügung steht und seinen Ursprung in Alaska hat. Finanziert wird das QGIS-Projekt durch Firmen, Behörden und privaten Spenden und steht unter der GNU General Public License[2]. Mit QGIS lassen sich eine Vielzahl an unterschiedlichen Raster- und Vektordatenformate laden und weiterverarbeiten. Interoperabilität besteht u.a. durch die Kompatibilität mit diversen (räumlichen) Datenbanke,n wie beispielsweise PostgreSQL und SQLite sowie durch die Unterstützung von Geowebdiensten (WMS, WFS, WPS, CSW). Datenbanktabellen, die eine Geometriespalten enthalten, lassen sich in QGIS sehr einfach grafisch darstellen und können auch im Shape-Format abgespeichert werden. Umgekehrt lassen sich auch Shape-Dateien mittels QGIS in eine räumliche Datenbank exportieren. QGIS ist plattformunabhängig programmiert und ist für Betriebssysteme wie Windows, Linux, Mac und Android verfügbar. QGIS ist in der objektorientierten Programmiersprache C++ und die Oberfläche in Qt[3], eine Programmbibliothek zur plattformübergreifenden Softwareentwicklung, geschrieben. Bekannte, in Qt geschriebene, Beispiele sind Google Earth, Virtual Box, Android und Skype. Die großen Vorteile liegen neben der freien Verfügbarkeit darin, dass QGIS über eine Programmierschnittstelle in den Programmiersprachen C++ oder Python 2.7[4] angesteuert werden kann und somit Zugriff auf sämtliche QGIS-Klassen besteht. Ein weiterer Pluspunkt besteht durch die Möglichkeit, Erweiterungen (Plugins) aus dem offiziellen QGIS-Erweiterungsrepositorium zu laden. Diese Plugins ermöglichen es, das auf dem PC vorhandene QGIS, um gewünschte Funktionen einfach zu erweitern, ohne eine Neuinstallation durchzuführen. Mittels der zuvor erwähnten Programmierschnittstellen lassen sich Skripte, eigene Plugins oder auch ein eigenes anwendungsspezifisches GIS (Standalone-Variante), auf Basis von QGIS erstellen. Diese Masterarbeit wurde mit QGIS 2.6 Brighton unter Windows 7 64 bit entwickelt.

QGIS setzt sich aus den Hauptmodulen[5]

- QGIS.analysis
- QGIS.core
- QGIS.gui
- QGIS.networkanalysis
- QGIS.server
- MapComposer

zusammen, die sich individuell über das API ansprechen lassen.

Mit über 400 core-Klassen ist das QGIS.core-Modul eines der am häfuigsten verwendeten

[1] http://www.qgis.org/de/site/
[2] http://www.gnu.org/licenses/old-licenses/gpl-2.0.html
[3] http://www.qt.io/developers
[4] https://www.python.org/download/releases/2.7
[5] http://www.qgis.org/api/modules.html

Module. Wichtige Programmbibliotheken, wie PROJ.4, für die Darstellung oder Konvertierung in eine beliebige Projektion sowie die Bibliotheken GDAL und OGR für die Analyse von Raster- und Vektordaten sind darin enthalten.

3.1.1 API

Eine Programmierschnittstelle (engl. Application Programming Interface, kurz API) ist ein Programmteil für die Ansteuerung eines Betriebssystems, eines Geräts oder einer Anwendung, die vom Hersteller zur Verfügung gestellt wird. Die objektorientierte Programmiersprache Python hat sich zu einem Quasi-Standard bei GIS etabliert und ist seit der Version 0.9 in QGIS integriert. Um mit Python rund um QGIS arbeiten zu können, wird ein Grundverständnis der objektorientierten Programmierung (OOP) vorausgesetzt. Python an sich, ist eine leicht zu erlernende und übersichtliche Programmiersprache, da sie ohne geschweifte Klammern auskommt. Einen schnellen und guten Einblick in Python gibt Theis, 2014. Der Zugriff mit Python auf die plattformunabhängige Bibliothek Qt erfolgt mittels Toolkit PyQt. Der Zugriff auf die QGIS-Module mit Python wird als PyQGIS bezeichnet. Für PyQGIS wird auf das "PyQGIS Developer Cookbook"[6] und auf die QGIS API Dokumentation[7] verwiesen.

3.1.2 Plugin

Wie bereits erwähnt, können Anwendungsgebiete und Funktionen von QGIS mithilfe Plugins problemlos erweitert werden. QGIS-Plugins werden in zwei Kategorien, Kern- und Erweiterungs-Plugins unterschieden. Kern-Plugins sind von Haus aus bereits bei der Installation vorhanden. Erweiterungs-Plugins können von dem offiziellen QGIS - Erweiterungsrepositorium[8] oder diversen anderen Verzeichnissen, die bei einer Internetsuche aufscheinen, geladen werden. Von Zeit zu Zeit kommt es auch vor, dass Erweiterungs-Plugins in den Kreis der Kern-Plugins aufgenommen werden.
Auch ein eigenes Plugin lässt sich mit etwas Geduld und Ausdauer, in Abhängigkeit der gewünschten Anforderung, relativ schnell erzeugen. Ein eigens dafür entwickeltes Plugin names *Plugin Builder*, aus dem offiziellen QGIS-Erweiterungsrepositorium, erstellt dabei ein Grundgerüst (siehe Abschnitt 3.1.2). Auf diesem aufbauend kann der Entwickler zur Tat schreiten und das Plugin individuell erweitern und gestalten. Eine detaillierte Anleitung für das Erstellen eines eigenen Plugins mittels *Plugin Builder*, ist im Internet[9,10] zu finden.

3.1.3 Standalone

Bei einer QGIS-Standalone-Applikation handelt es sich um ein eigenständiges, in den Funktionen eher reduziertes, GIS auf QGIS-Basis. Gründe, eine Standalone-Applikation einem Plugin vorzuziehen, können u.a. die Einschränkung von im QGIS vorhandener Werkzeuge sein. Dies dient dazu einer unbeabsichtigte Datenmanipulation vorzubeugen. Nur mit den

[6] `http://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook`
[7] `http://qgis.org/api`
[8] `https://plugins.qgis.org/plugins`
[9] `http://www.digital-geography.com/build-qgis-plugin`
[10] `http://www.qgisworkshop.org/html/workshop/plugins_tutorial.html`

Funktionen, die in QGIS als Kartenwerkzeuge (engl. map tools) bezeichnet werden, die auch den Anforderungen entsprechen, ist auch mehr Übersicht und ein leichteres Arbeiten gegeben.In Abbildung 3.2 ist die im Rahmen dieser Masterarbeit erstellte Standalone-Anwendung dargestellt.

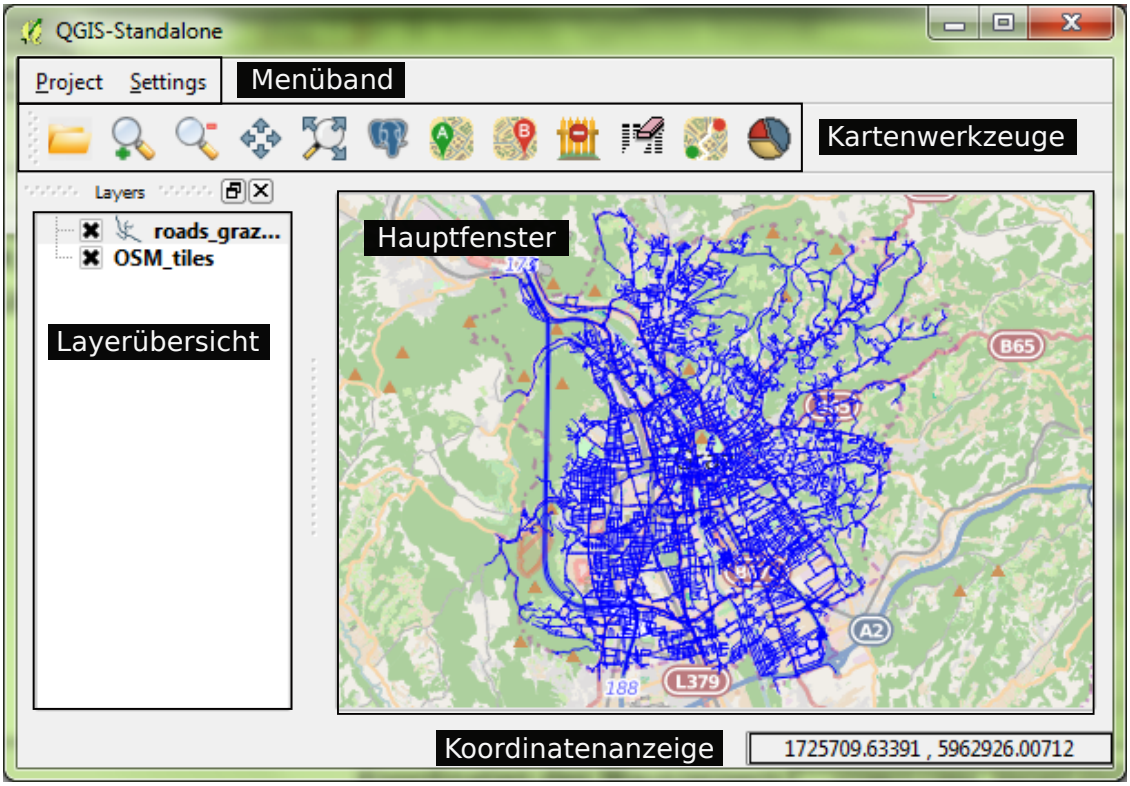


Abbildung 3.2: Standalone-Applikation

Die grafischen Elemente, wie Kartenfenster, Menüband, Ebenenübersicht oder Schaltflächen werden mittels PyQt erzeugt. Damit Raster- und Vektordaten verarbeitet und dargestellt werden können, werden die Hauptmodule QGIS.core und QGIS.gui importiert. Nachfolgend werden die einzelnen Elemente der Standalone-Anwendung näher erläutert.

Hauptfenster

Das Hauptfenster ist das zentrale Element der Standalone-Anwendung. Um dieses Hauptfenster herum werden alle relevanten Elemente, wie Kartenwerkzeuge, Ebenenübersicht oder aktuelle Mausposition, im Koordinatensystem der geladenen Ebene, angefügt.

Menüband

Das in Abbildung 3.2 und Abbildung 3.3 ersichtliche Menüband beinhaltet zwei Schaltflächen. Die erste Schaltfläche *Projcet* dient nur dem Schließen der Anwendung. Sie lässt sich (zukünftig) problemlos erweitern. Die zweite Schaltfläche *Settings* hat wiederum zwei Unterpunkte, die als *Routing* und *Database* bezeichnet werden.

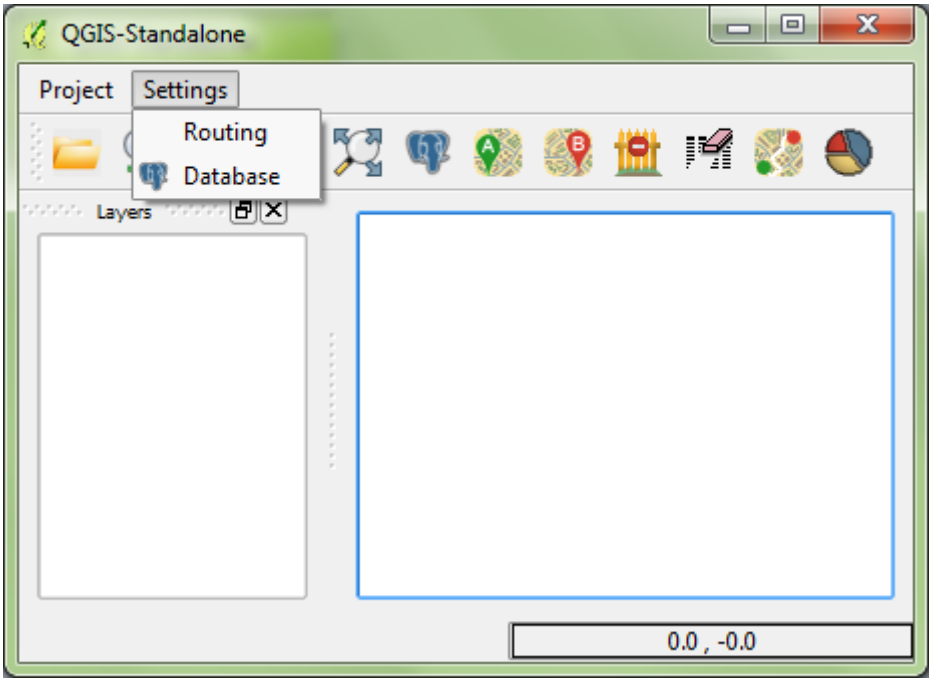


Abbildung 3.3: Menüband - Einstellungen

Routing. Mit dem Unterpunkt *Routing* können Einstellungen für eine nachfolgende Routenberechnung getroffen werden (siehe Abbildung 3.4).

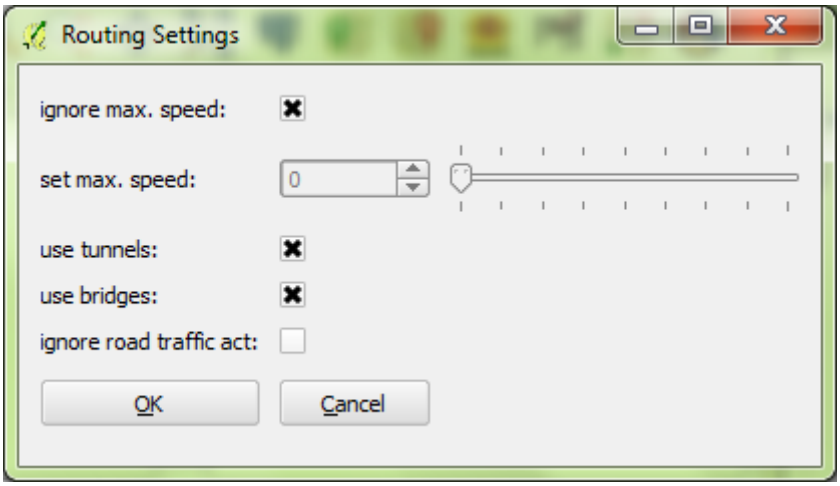


Abbildung 3.4: Routing - Einstellungen

Bei den in Abbildung 3.4 gesetzten Einstellungen handelt es sich um die Standardeinstellung der Routing-Optionen. Nachfolgend werden die einzelnen Einstellungen nochmals aufgelistet:

- ignore max. speed

- use tunnels
- use bridges
- ignore road traffic act

Mittels *ignore max. speed* kann das Tempolimit, der jeweiligen Straße außer Acht gelassen und auf das Limit des tatsächlichen Fahrzeugtyps, gesetzt werden. Durch diese Option können sich neue Routing-Varianten ergeben. Ist das Fahrzeug vom Typ LKW, so muss nicht immer die Route über eine Autobahn die schnellste Verbindung sein, da LKWs auf Grund eigener Tempolimits idR. nicht so schnell fahren dürfen, wie PKW. Somit eignet sich die Autobahn nicht unbedingt besser, als eine Schnell- oder Bundesstraße.

Mit *use tunnels* und *use bridges* können Tunnels und Brücken in die Routenfindung einbezogen bzw. ausgeschlossen werden. Anwendung finden diese beiden Optionen beispielsweise bei Sonder- oder Gefahrenguttransporten, wo Auflagen seitens des Gesetzgeber bestehen.

Die letzte Option *ignore road traffic act* lässt die Straßenverkehrsordnung komplett außer Acht. Zum Einsatz könnte diese Einstellung beispielsweise bei einem Unfall auf einer Autobahn kommen, wo in Fahrrichtung, aufgrund verstopfter Fahrspuren, weder mittels Rettungsgasse noch über den Pannenstreifen der Unfallort von den Einsatzkräften erreicht werden kann. Diese Option ermöglicht es, eine Route zu finden, die weder Rücksicht auf Fahrtrichtungen noch auf Einbahnen nimmt. Dadurch wird die Route für die Einsatzkräfte über die Autobahnabfahrt, in entgegengesetzter Fahrtrichtung zum Unglücksort ermittelt.

Database. Im Untermenü *Database* können Datenbankparameter eingetragen bzw. wenn schon eine Verbindung hergestellt wurde, eingesehen werden (siehe Abbildung 3.5).

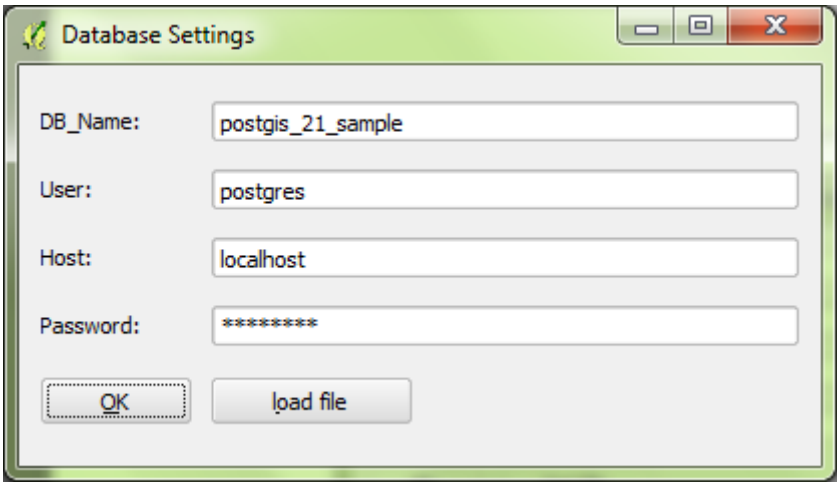


Abbildung 3.5: Datenbank - Einstellungen

Mit der Schaltfläche *load file* lassen sich einfach Datenbankparameter, aus zuvor getätigte Datenbankverbindungen, laden.

Kartenwerkzeuge

Bei einem Kartenwerkzeug handelt es sich um ein Objekt einer Klasse, welches aufgrund seiner Methoden ein spezielles Verhalten aufweist. Über diese Kartenwerkzeuge erfolgt jegliche Interaktion zwischen dem Benutzer und der Standalone-Anwendung. Nachfolgend sind alle verwendeten Kartenwerkzeuge der im Rahmen dieser Masterarbeit entwickelten Standalone-Anwendung aufgelistet:

1.  *Datei öffnen*
   Das Laden von Raster- und Vektordaten erfolgt in QGIS über zwei eigene Schaltflächen. Bei dem Kartenwerkzeug Nummer 1 *Datei öffnen*, handelt es sich um eine Weiterentwicklung der QGIS-Funktion, um über eine Schaltfläche unterschiedliche Datenformate laden zu können. Damit lassen sich problemlos Daten im Shape-, TIFF- oder XML-Format (siehe Abschnitt 2.1.2) öffnen.

2.  *Vergrößern*

3.  *Verkleinern*

4.  *maximale Ausdehnung*

5.  Verschieben
   Bei den Kartenwerkzeugen Nummer 2 bis 5, handelt es sich um QGISeigene Werkzeuge. Sie dienen rein der Betrachtung, der als Karte geladenen Daten.

6.  *Layer aus Datenbank laden*
   Über das Werkzeug Nummer 6, *Layer aus Datenbank laden*, können ein oder mehrere Layer aus einer Datenbank in die Anwendung geladen werden.

7.  *Startpunkt setzen*

8.  *Zielpunkt setzen*
   Die Werkzeuge Nummer 7 *Startpunkt setzen* und Nummer 8 *Zielpunkt setzen* sind beide Objekte der selben Klasse. Einmal wird damit der Startpunkt und einmal der Endpunkt der Route festgelegt. Die Klasse ist so konzipiert, dass auch bei einem nicht exakten Klicken auf den Anfangs- oder Endpunkt einer Kante, der Start- bzw. Endpunkt auf den nähest gelegenen Anfangs- oder Endpunkt der Kante gesetzt wird. Um diese beiden, in der Karte gesetzten Punkte, voneinander unterscheiden zu können, wird der Startpunkt, durch ein grünes Kreuz und der Endpunkt, durch ein rotes Kreuz dargestellt.

9.  *Routing*
   Das Kartenwerkzeug *Routing* dient, nach erfolgreichem Setzen des Start- und Zielknotens, dem Berechnen der optimalen Route. Die Klasse, die sich hinter diesem Kartenwerkzeug verbirgt, überprüft ob Start- und Zielknoten vorhanden sind und wenn dies der Fall ist, so übergibt die Klasse beide Knoten an die pgRouting-Funktion *pgr_astar* (siehe Abschnitt 3.2.4) in der PostgreSQL-Datenbank. Sobald pgRouting

ein Ergebnis berechnet hat, wird dieses grafisch in der Karte dargestellt (siehe Abschnitt 3.4.3).

10.  *Sperrfläche*
Kartenwerkzeug Nummer 10 *Geofence* dient dem Einzeichnen von Sperrbereichen, durch die keine Route verlaufen darf. Polygone repräsentieren dabei Sperrzonen. Sobald mit diesem Kartenwerkzeug ein (erster) Sperrbereich in der Karte eingezeichnet wurde, erscheint ein eigener Layer, in dem die Sperrflächen als Polygone erfasst sind. Es können auch Shape-Dateien, mit bereits enthaltenen Polygonen, geladen werden, um sich so das Einzeichnen zu ersparen. Klickt man nun auf *Routing*, werden die Sperrbereiche bei der Routenberechnung berücksichtigt.

11.  *Sperrfläche löschen*
Kartenwerkzeug Nummer 11 *delete Geofence* ermöglicht ein selektives Löschen der zuvor selbst gezeichneten Sperrzonen.

12.  *Auswertung*
Das letzte Kartenwerkzeug *Auswertung* gibt das Routingergebnis in tabellarischer und grafischer Form wieder. In einer Tabelle werden die Routingeinstellungen sowie die errechnete Zeit und Distanz aufgelistet. In einem Diagramm werden die An- und Abstiege der OSM-Teilsegmente dargestellt.

Layerübersicht

Für komfortables Arbeiten und schnelles Wechseln zwischen einzelnen Layer, ist eine Layerübersicht von Vorteil. Eine von der Organisation Ecotrust (2007) und den Personen Aaron Racicot (2007) und Germán Carrillo[11] (2009) entwickelte PyQt-Layerübersicht wurde für diese Masterarbeit übernommen, angepasst und weiterentwickelt. Die in Abbildung 3.6 dargestellte Layerübersicht beinhaltet zwei unterschiedliche Datensätze. Zum einen das Straßennetz von Graz, welches in blau dargestellt ist, und zum anderen die OSM-Karte als Hintergrund für die Straßendaten.

[11] `http://geotux.tuxfamily.org`

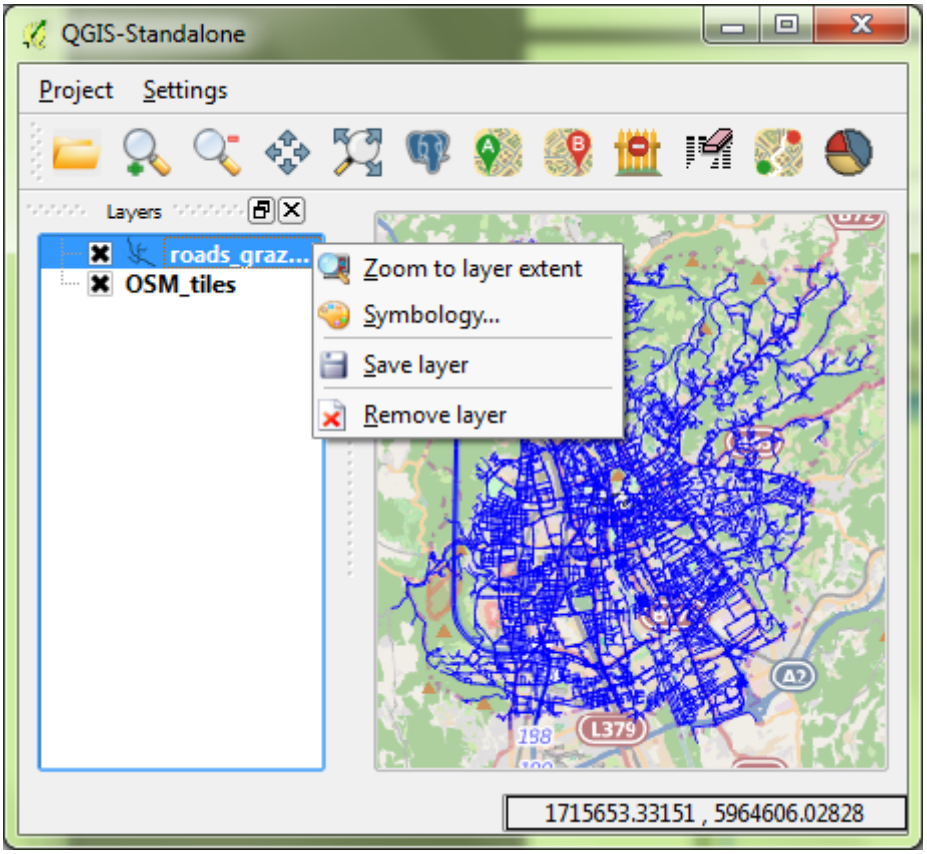


Abbildung 3.6: Layerübersicht

Mit der linken Maustaste können die Layer in einer beliebigen Reihenfolge angeordnet werden. Mit dem Setzten des Kästchens links neben dem Layernamen, kann die Sichtbarkeit des Layers aktiviert werden. Über die rechte Maustaste gelangt man zum Layer-Menü. Die vier Menüpunkte sind:

- Zoom to layer extent: Gleiche Funktion, wie beim Kartenwerkzeug Nummer 4.
- Symbology...: Dadurch lässt sich die Farbe des Datensatzes beliebig ändern.
- Save layer: Im Falle eines Vektordatensatzes kann dieser Layer als Shape-Datei gespeichert werden. Vorteilhaft ist dies, wenn ein Datensatz als Tabelle aus einer ORDB (siehe Abschnitt 2.4.2) geladen wird.
- Remove layer: Hiermit wird der Layer aus der Layerübersicht entfernt.

Koordinatenanzeige

Die Koordinatenanzeige dient der Wiedergabe der aktuellen Mausposition im Koordinatensystem, bzw. der Projektion mit der SRID 3857 (siehe Tabelle 2.2 und Abschnitt 2.1.1).

3.2 PostgreSQL

Bei PostgreSQL (auch Postgres genannt) handelt es sich um ein freies ORDBMS (siehe Abschnitt 2.4.2), dessen Quellcode zur Gänze öffentlich einsehbar ist. Begonnen hat das Projekt Mitte der 1980er Jahre an der Universität Berkeley in Kalifornien. PostgreSQL wird nicht von einem gewinnorientiertem Unternehmen, sondern von einem weltweiten Team, bestehend aus Softwareentwicklern auf freiwilliger Basis, betreut und weiterentwickelt. PostgreSQL unterliegt der Berkeley Software Distribution (BDS)-Lizenz, die eine kostenlose Veränderung, Verwendung und Verbreitung gestattet. Die aktuelle PostgreSQL Version ist 9.4 und unterstützt größtenteils den SQL2003-Standard[12]. PostgreSQL arbeitet, wie viele andere DBMS auch, als Server in einem Client-Server-System. Anfragen werden von einem Client an den Server gestellt. Die Abfragesprache ist SQL und als Antwort bekommt man eine Tabelle aus der Datenbank. PostgreSQL ist besonders auf das Verwalten und Verarbeiten von großen Datenmengen ausgelegt und zeichnet sich auch durch Erweiterungen, wie pgAdmin, PostGIS und pgRouting aus.

3.2.1 pgAdmin

Bei pgAdmin handelt es sich um eine grafische Benutzeroberfläche für die Administration von Datenbankservern. PgAdmin ist eine Open-Source Entwicklung und eignet sich besonders gut für PostgreSQL-Datenbanken. Diese Software ist für die Betriebssysteme BSD, Linux, Mac OSX, Solaris und Windows verfügbar. Sollte man trotz dieser Fülle mit einem anderen Betriebssystem arbeiten, so ist pgAdmin auch im Quellcode verfügbar und kann selbst kompiliert werden. Für Liebhaber der Kommandozeile kann die Administration des Datenbankservers auch über das pgAdmin-Terminal "psql" ausgeführt werden. Mittels pgAdmin und SQL können Tabellen sehr angenehm erstellt, manipuliert und gelöscht werden.

3.2.2 PostGIS

Bei PostGIS handelt es sich um eine Erweiterung von PostgreSQL zur Verwaltung räumlicher Geodaten. PostGIS unterstützt u.a. die Geometrietypen WKT und WKB (siehe Abschnitt 2.1.1). Mit den räumlichen PostGIS-Funktionen können Vektor- und Rasterdaten analysiert und Berechnungen von u.a. Distanzen, Flächen oder Pufferzonen durchgeführt werden. Die räumliche Information ist in einer zusätzlichen Spalte, auch Geometriespalte genannt, der Tabelle enthalten. Mit der PostGIS-Erweiterung ist PostgreSQL GIS-fähig und repräsentiert eine Geodatenbank.

3.2.3 PL/pgSQL

Bei Procedural Language/PostgreSQL (PL/pgSQL) handelt es sich ebenfalls um ein Erweiterungsmodul von PostgreSQL. Dieses Modul erlaubt es, zu den bereits vorhandenen Funktionen der PostGIS-Erweiterung, eigene Funktionen zu erstellen und der Datenbank hinzuzufügen. PL/pgSQL erweitert den SQL-Funktionsumfang und unterstützt Variablen,

[12] `http://sigmod.org/publications/sigmod-record/0403/E.JimAndrew-standard.pdf`

Schleifen und Bedingungen. Für Näheres zur Programmierung siehe PL/pgSQL[13] und Stones, 2005.

3.2.4 pgRouting

Bei pgRouting[14] handelt es sich um eine Erweiterung von PostgreSQL, mit dem Ziel Routing- und Netzwerkanalyse-Funktionen bereitzustellen, die aktuell von den Unternehmen Georepublic[15] und iMaptools[16] betreut wird.

Routing-Funktionen

Nachfolgend sind die enthaltenen Routingfunktionen aufgelistet:

- *pgr_ apspJohnson* (All Pairs Shortest Path, Johnson's Algorithm)
- *pgr_ apspWarshall* (All Pairs Shortest Path, Floyd-Warshall Algorithm)
- *pgr_ astar* (Shortest Path A*)
- *pgr_ bdDijkstra* (Bi-directional Dijkstra Shortest Path)
- *pgr_ bdAstar* (Bi-directional A* Shortest Path)
- *pgr_ dijkstra* (Shortest Path Dijkstra)
- *pgr_ drivingDistance* (Driving Distance)
- *pgr_ ksp* (K-Shortest Path, Multiple Alternative Paths)
- *pgr_ kDijkstra* (K-Dijkstra, One to Many Shortest Path)
- *pgr_ tsp* (Traveling Sales Person- TSP)
- *pgr_ trsp* (Turn Restriction Shortest Path - TRSP)

Die exakte Syntax für o.a. Routing-Algorithmen kann der pgRouting-Dokumentation[17] entnommen werden.

[13] `http://www.postgresql.org/docs/9.3/static/plpgsql.html`
[14] `http://pgrouting.org/index.html`
[15] `https://georepublic.info/de`
[16] `http://imaptools.com/contact.html`
[17] `http://docs.pgrouting.org/2.0/en/doc/index.html`

Netzwerkanalysefunktionen

Netzwerkanalysefunktionen dienen dem Erstellen einer Knoten-Kanten-Struktur (Routing-Topologie). PgRouting beinhaltet Funktionen, mit denen aus einer Kantentabelle eine Knotentabelle erzeugt werden kann. Zusätzlich sind Funktionen inkludiert, mit denen sich isolierten Kanten, Kanten die nicht mit dem Hauptgraphen zusammenhängen, aufspüren lassen. Nachfolgend sind die Netzwerkanalysefunktionen aufgelistet:

- *pgr_createTopology*
- *pgr_createVerticesTable*
- *pgr_analyzeGraph*
- *pgr_nodeNetwork*

Wie auch bei den Routing-Funktionen, kann auch hier die Syntax für das Verwenden o.a. Funktionen der pgRouting-Dokumentation entnommen werden.

3.3 Datenbeschaffung

3.3.1 Geofabrik

Geofabrik[18] wurde von Jochen Topf und Frederik Ramm gegründet und stellt u.a. auch kostenlos OSM-Daten zur Verfügung. Einmal täglich wird dazu das „planet-file“ der OSM dafür herangezogen und daraus kontinent- bzw. länderweise OSM-Daten extrahiert[19]. Die Daten stehen in den Datenformaten *osm.pbf*, *shp.zip* und *osm.bz2* zur Verfügung. Für diese Masterarbeit wurden Daten im *osm.pbf*-Format verwendet. Daten aus der OSM bzw. von Geofabrik verwenden das geodätisches Datum WGS84 mit EPSG-Code 4326.

3.3.2 osm2po

Um die OSM-Daten in eine ORDB zu bekommen, bedient man sich des osm2po[20] Konverters, der die Daten in eine Datei mit SQL-Anweisungen umwandelt. Da osm2po in Java programmiert ist, benötigt man zur Ausführung eine Java-Laufzeitumgebung[21]. In Tabelle 3.1 sind die Spaltennamen samt Datentyp[22], der aus osm2po resultierenden SQL-Datei aufgelistet.

[18] `http://www.geofabrik.de`
[19] `http://download.geofabrik.de`
[20] `http://osm2po.de`
[21] `https://www.java.com/de/download`
[22] `http://www.postgresql.org/docs/9.4/static/datatype.html`

Tabelle 3.1: SQL-Datei

Spaltenname		Datentyp
id		integer
osm_id		bigint
osm_name	⋆	character varying
osm_meta	⋆	character varying
osm_source_id	⋆	bigint
osm_target_id	⋆	bigint
clazz		integer
flags	⋆	integer
source		integer
target		integer
km		double precision
kmh		integer
cost		double precision
reverse_cost		double precision
x1		double precision
y1		double precision
x2		double precision
y2		double precision
geom_way		geometry(LineString,4326)

Durch Löschen der in Tabelle 3.1, mit ⋆ gekennzeichneten, nicht benötigten Spalten, reduziert sich der Speicherplatz in der Datenbank. Mittels psql-Konsole von pgAdmin (siehe Abschnitt 3.2.1) wird die zuvor erzeugte SQL-Datei in der Datenbank ausgeführt. Fortan befindet sich das jeweilige Straßennetz, des zuvor gewählten Landes, als Kantentabelle in der Datenbank. Die für Routingaufgaben relevante Spalte *cost* berechnet sich laut Gleichung (3.1) und die Einheit dieser Spalte ist Stunden.

$$cost = \frac{km}{kmh} \tag{3.1}$$

3.3.3 Overpass-API

Das Overpass-API[23] dient zum selektiven Download von Daten aus dem OSM-Datenbestand. Über die eigene Abfragesprache (engl. query language, QL) Overpass QL[24], hat man lesenden Zugriff auf sämtliche Daten der OSM. Die Anfrage kann geometrisch, geografisch oder auch sachlich sein. Abfragen werden nach dem Key-Value-Prinzip und dem OSM-Datentyp formuliert (siehe Abschnitt 2.2.1). Die abgefragten Daten liegen im XML-Format vor und sind aktueller, als der Datenbestand der Geofabrik.

3.3.4 Höhendaten

Um in die optimale Routenfindung auch die Topografie einfließen zu lassen, werden die verebneten Straßendaten um Höheninformationen erweitert. Für die Auswahl der passenden

[23] `http://overpass-turbo.eu`
[24] `http://wiki.openstreetmap.org/wiki/Overpass_API/Language_Guide#Non-exact_names`

Höhendaten, kommt es neben der freien Verfügbarkeit vor allem auf deren räumliche Abdeckung und räumliche Auflösung an. Nachfolgend sind drei in Abdeckung und Auflösung unterschiedliche, jedoch frei verfügbare Höhendatensätze angeführt.

- Shuttle Radar Topography Mission (SRTM[25]):
  SRTM-Daten wurden im Jahr 2000 mittels Space-Shuttle-Mission von der NASA erhoben. Die Auflösung beträgt 3 Bogensekunden. Dies entspricht ca. 90 m am Äquator. Zwar gibt es schon Daten mit einer Auflösung von 1 Bogensekunde, jedoch sind diese Daten bis heute nur für Nordamerika verfügbar.
- Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER[26]):
  ASTER-Daten entstammen einer Kooperation der NASA und dem japanischen Wirtschafts- und Handelsministerium. Die Daten haben eine Auflösung von 1 Bogensekunde, das entspricht 30 m am Äquator.
- Airborne Laserscanning-GIS Steiermark (ALS[27]):
  Die GIS Steiermark-ALS-Daten werden aus regelmäßig stattfindenden Scanflügen abgeleitet. Die Aktualität ist im Vergleich zu SRTM und ASTER sehr hoch. Die Auflösung beträgt 10 m.

Legt man um die Steiermark ein minimal umgebendes Rechteck (engl. minimum bounding rectangle), das die Landesgrenzen gerade noch einschließt (siehe Abbildung 3.7), so erhält man die in Tabelle 3.2 aufgelisteten, geografischen Koordinaten.

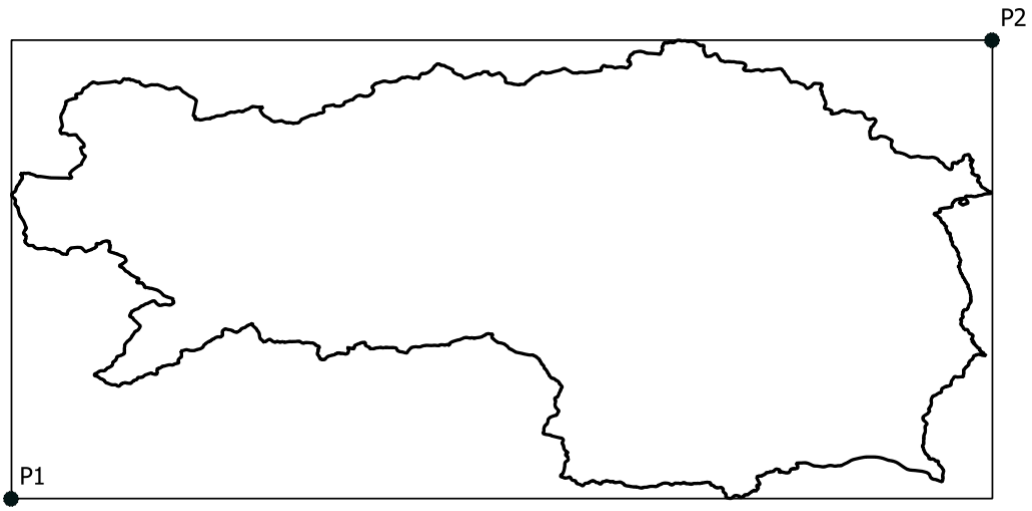


Abbildung 3.7: Minimales Rechteck um die Steiermark

[25] http://www2.jpl.nasa.gov/srtm
[26] https://asterweb.jpl.nasa.gov
[27] http://www.gis.steiermark.at/cms/beitrag/10481712/14292094

Tabelle 3.2: Ausdehnung Steiermark

Punkt	nördl. Breite [°]	östl. Länge [°]
P1	46,6	13,5
P2	47,8	16,2

In Tabelle 3.3 sind die technischen Merkmale der zuvor vorgestellten Höhendatensätze, nochmals aufgelistet und gegenübergestellt. Da es sich bei SRTM und ASTER um Satellitenmissionen handelt und die Abdeckung von den Orbits abhängig ist, wurden die Polregionen nicht mit erfasst. Um die verschiedenen Datensätze, bezogen auf die geografische Ausdehnung der Steiermark, besser untereinander vergleichen zu können, sind die Auflösungen, bezogen auf 47° nördlicher Breite, in Tabelle 3.3 ebenfalls aufgelistet.

Tabelle 3.3: Höhendaten

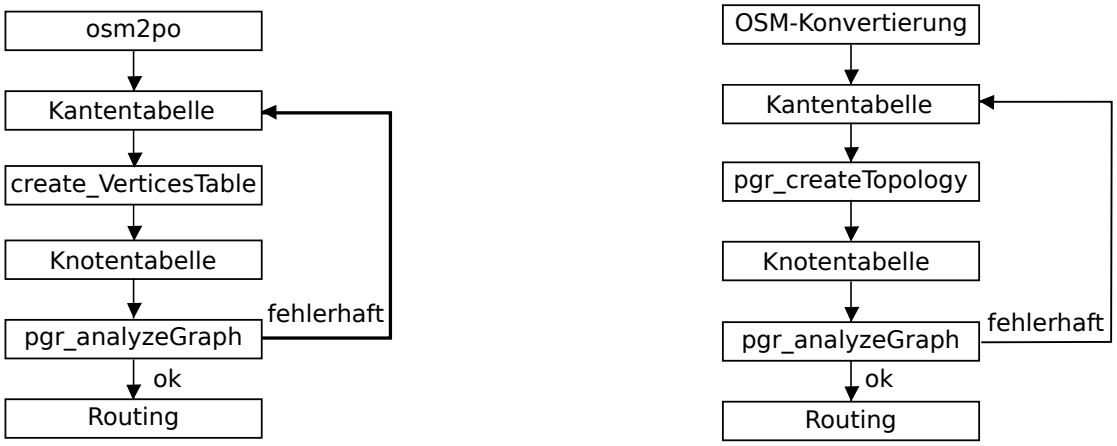
Höhendaten	Auflösung	Auflösung bei 47° nördl. Breite	Abdeckung	Bereich
SRTM	90 m	61 m	global	$56°\text{S} < \varphi < 60°\text{N}$
ASTER	30 m	20 m	global	$\varphi \pm 83°$
ALS-Stmk	7 m	10 m	regional	steiermarkweit

3.4 Vorprozessierung der Daten

Bevor man mit der QGIS-Standalone-Applikation die eigentliche Arbeit aufnehmen kann, bedarf es noch ein paar Vorarbeiten. So müssen, die mittels osm2po in die Datenbank importierten OSM-Straßendaten, zuerst einer Topologieprüfung unterzogen werden. Im Anschluss werden Informationen über das Vorhandensein von Brücken und Tunnel dem Datensatz hinzugefügt und abschließend wird die Tabelle um Höheninformationen erweitert.

3.4.1 Graph erstellen

Die mittels osm2po-Konverter (siehe Abschnitt 3.3.2) erstellte und in die Datenbank importierte Straßendatentabelle bekommt von osm2po automatisch den Tabellennamen "Präfix"_2po_4pgr zugewiesen. Der Vorteil an osm2po ist, dass die in Tabelle 3.1 angeführten Spalten bereits befüllt sind. Abbildung 3.8 gibt die Graphenerstellung in zwei unterschiedlichen Varianten wieder. In Abbildung 3.8(a) wird der Graph mit dem osm2po-Konverter erzeugt. Alle Spalten, (siehe Tabelle 3.1) der daraus resultierenden Kantentabelle, sind bereits befüllt. Werden die OSM-Daten mit anderen Konvertern als osm2po konvertiert, so muss nach dem Schema in Abbildung 3.8(b) vorgegangen werden.



(a) mit osm2po

(b) andere Konverter

Abbildung 3.8: Graphenerstellung

Da für die pgRouting-Funktion *pgr_ analyzeGraph* (siehe **??** 30) eine Kantentabelle und eine zugehörige Knotentabelle vorhanden sein muss, wird diese mithilfe der Funktion *create_ VerticesTable* erstellt. In Tabelle 3.4 ist die Knotentabelle mit den enthaltenen Spalten dargestellt.

Tabelle 3.4: Knotentabelle

Spaltenname	**Datentyp**
id	bigint
cnt	integer
chk	integer
ein	integer
out	integer
the_geom	geometry(Point,4326)

Um mit der Funktion *pgr_ analyzeGraph* arbeiten zu können, muss der Namen der zu untersuchenden Kantentabelle sowie die Snapping-Toleranz angegeben werden. Mündet in der Realität eine Straße in eine Kreuzung, so muss das in der OSM nicht unbedingt der Fall sein. Aufgrund Zeichenfehler oder Messungenauigkeiten der Koordinaten kann die zuvor erwähnte Straße kurz vor der Kreuzung enden und stellt somit eine Sackgasse dar. Um dem entgegenzuwirken, gibt es die Snapping-Toleranz, die im Umkreis um jeden Knoten, der zuvor erstellten Knotentabelle, nach Kanten sucht. Dabei ist die Snapping-Toleranz der Radius des Umkreises in der jeweiligen Einheit der Abbildung. Entweder ellipsoidische Koordinaten (φ, λ) mit der Einheit Altgrad [°] oder verebnete Koordinaten $(x,\ y)$ mit der Einheit Meter [m]. Der SQL-Befehl hierfür lautet:

```
SELECT pgr_analyzegraph(Kantentabelle, Snapping-Toleranz);
```

Mögliche Fehler die von *pgr_ analyzeGraph* aufgedeckt werden können sein:

- isolierte Segmente

- Überschneidungen
- Ringgeometrien

Sind isolierte Segmente vorhanden, so ist der Graph nicht zusammenhängend. Hier muss (visuell) geprüft werden, ob es sich um einen Fehler handelt oder nicht. In Grenzregionen kann es durchaus vorkommen, dass Straßen, beispielsweise eine Taleinfahrt, zwar auf dem zur Tabelle gehörenden Staatsgebiet liegen, jedoch nur über das benachbarte Land zu erreichen sind. Überschneiden sich zwei Kanten, ohne dass sich am Schnittpunkt ein Knoten befindet, so kann dies mit der Funktion *pgr_ NodeNetwork* behoben werden.

```
SELECT pgr_NodeNetwork(Kantentabelle, Snapping-Toleranz);
```

Die Funktion *pgr_ NodeNetwork* bricht die überschnittenen Kanten in Teilsegmente auf und erweitert die Knotentabelle um den Schnittpunkt und die Kantentabelle um das neuentstandene Kantensegment. Jedoch sollte man diese Überschneidungen zuvor auch visuell überprüfen, da es sich auch um ein Autobahnkreuz oder eine Brücke handeln kann. Bei Ringgeometrien handelt es sich meist um die graphentheoretische Umsetzung einer Umkehrmöglichkeit in einer Sackgasse. Sind keine Fehler vorhanden bzw. wurden die Fehler in der Kantentabelle behoben, so liegt eine routingfähige Knoten-Kanten-Struktur vor.
Das in Abbildung 3.8(b) gezeigte Schema kann auf Straßendaten in Tabellenform angewandt werden, die nur den Straßennamen und die Geometrie beinhalten. Um einen routingfähigen Graphen daraus zu erstellen, muss die vorhanden Kantentabelle um die fehlenden Spalten (siehe Tabelle 3.1) erweitert werden. Die Funktion *pgr_ createTopology* befüllt die zuvor hinzugefügten Spalten der Kantentabelle und erstellt eine Knotentabelle (siehe Tabelle 3.4). Nachfolgend ist der SQL-Befehl für *pgr_ createTopology* angeführt:

```
SELECT pgr_createTopology(Kantentabelle, Snapping-Toleranz,Geometriespalte) ;
```

Anschließend erfolgt eine Analyse der Graphen mittels der Funktion *pgr_ analyseGraph* und eine Behebung der sich möglicherweise daraus ergebenden Fehler. Die Vorgehensweise ist dabei identisch mit der Erklärung zu Abbildung 3.8(b). Als Ergebnis liegt nun ein routingfähiger Graph vor.

3.4.2 Graph erweitern

Im Rahmen dieser Arbeit können die zu untersuchenden Graphen um

- Brücken und Tunnel
- Höhendaten
- Verknüpfen von Länderdaten

erweitert werden.

Brücken und Tunnel

Die von dem osm2po-Konverter erstelle Kantentabelle entspricht geometrisch einer 1:1-Kopie der OSM. Bei den Attributen ist sie aber stark reduziert und generalisiert. Wichtige Informationen über die Straße, beispielsweise, ob es sich um eine Brücke oder um einen Tunnel handelt, gehen dabei verloren. Um diese Klassifizierung jedoch in die Routenfindung einfließen lassen zu können, muss die bestehende Kantentabelle um ein Attribut erweitert werden. Mittels Overpass-API (siehe Abschnitt 3.3.3) können die verloren gegangenen Informationen rückgewonnen werden. Handelt es sich um eine Brücke, so wird in der Kantentabelle in der jeweiligen Zeile das Element, der neu hinzugefügten Spalte, auf 1 gesetzt. Analog dazu bekommen Tunnel den Wert -1 zugewiesen. Den Routingeinstellungen entsprechend können nun Brücken und Tunnel in der Routenplanung erlaubt oder auch verboten werden. Im Falle eines Verbots von Tunnel und/oder Brücken werden die Kosten der betroffenen Kanten auf -1 gesetzt.
Da der A*-Algorithmus (siehe Abschnitt 2.11.2) lediglich positive Kosten der einzelnen Kanten, nicht aber zusätzliche Attribute wie z.B. Brücken oder Tunnel berücksichtigt, garantiert diese Vorgehensweise, dass eine Routenfindung trotz Tunnelverbot durch einen oder mehrere Tunnel ausgeschlossen werden kann. Um die Kosten wieder auf ihren ursprünglichen Wert zu setzten, bedient man sich Gleichung (3.1), integriert in einem SQL-*INSERT INTO*-Befehl.

Höhendaten

Die Straßensteigung ist ein wesentlicher Parameter für die Wahl des richtigen Fahrzeugs. Aus den in Abschnitt 3.3.4 vorgestellten und in Tabelle 3.3 aufgelisteten unterschiedlichen Höhendatensätzen, sollen für das Routing relevante Steigungen, errechnet werden. Mit der QGIS-Funktion *slopeaspectcurvature* werden aus den Höhendaten, in diesem Fall handelt es sich um digitale Geländemodelle (DGM), im Rasterformat, Hangneigungsmodelle errechnet. Jede einzelne Rasterzelle des Hangneigungsmodells, bekommt als Wert dabei die maximale Steigung zugewiesen. Abbildung 3.9 zeigt in einer Gegenüberstellung die einzelnen Auflösungen der Hangneinungsmodelle für eine geografische Breite von 47°. Die schwarze Kurve repräsentiert dabei den Median aus dem Österreichdatensatz von Geofabrik (Länge: 62 m, siehe Tabelle 3.5) und gibt einen Größenvergleich, Straßenelement im Verhältnis zu Rasterzelle, wieder.

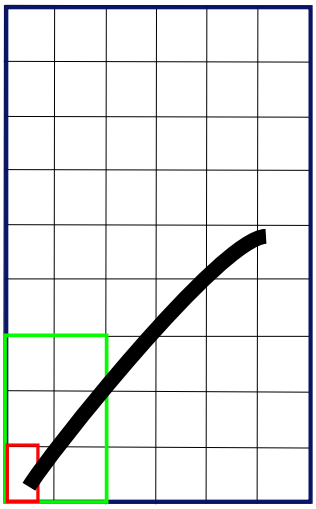


Abbildung 3.9: Auflösung-Rasterzelle

Um eine Aussage treffen zu können, welche Auflösung ein Höhenmodell haben muss, um daraus eine richtige Höheninformation bzw. die Steigung für die OSM-Straßenkanten abzuleiten, wird eine Mindeststraßenbreite von 3 m angenommen. Abbildung 3.10 stellt einen Geländeaufriss quer zur Fahrbahn dar. Der Höhenwert pro Rasterzelle, entspricht bei konstanter Steigung oder Gefälle dem Mittelwert (markiert durch schwarzem Kreis auf rot strichlierter Linie), errechnet sich aus der minimalen (roter Punkt) und maximalen Höhe (blauer Punkt). Sollte die Neigung des Geländes pro Rasterzelle nicht konstant sein, so ist der abgegriffene Höhenwert ein gewichtetes Mittel.

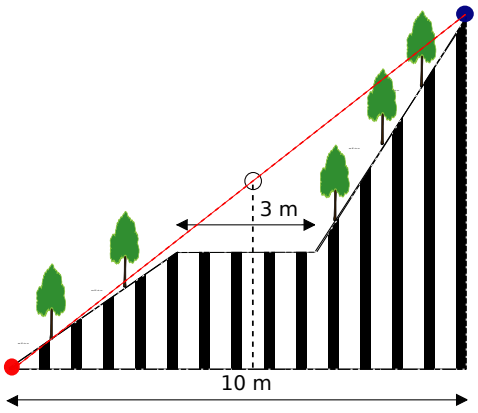


Abbildung 3.10: Geländeaufriss

Um eine nicht verfälschte Höhe entlang einer 3 m breiten Straße aus einem Höhenmodell ableiten zu können, darf die Rasterzellenauflösung des Höhenmodells 1,5 m nicht übersteigen.

Hierbei handelt es sich um einen theoretischen Wert und unter der Voraussetzung, dass die Straße in ihrer Ausrichtung mit der Lage der Rasterzellen übereinstimmt. Abbildung 3.11 zeigt unterschiedliche Lagemöglichkeiten von Rasterzellen mit einer theoretischen Auflösung von 1,5 m. Sollte die Auflösung größer als 1,5 m sein, fließen auch Höheninformationen neben der Straße in die der Rasterzelle mit ein.

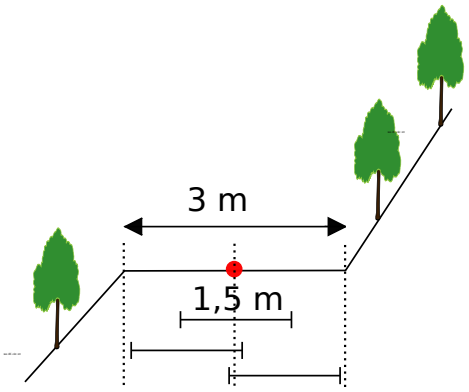


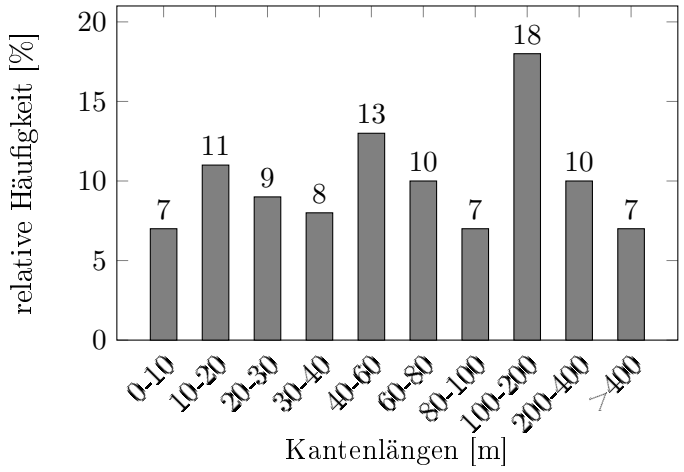
Abbildung 3.11: Geländeaufriss-Rasterzellenauflösung

Für die Bestimmung der theoretischen Auflösung des Höhenmodells wurde die Straßenbreite herangezogen. Um jedoch die maximale Steigung einer OSM-Kante zu ermitteln, muss man auch die Länge der einzelnen Straßenelemente näher untersuchen. In Tabelle 3.5 sind die beiden Extremwerte (Minimum und Maximum), sowie Mittelwert und Median aufgelistet.

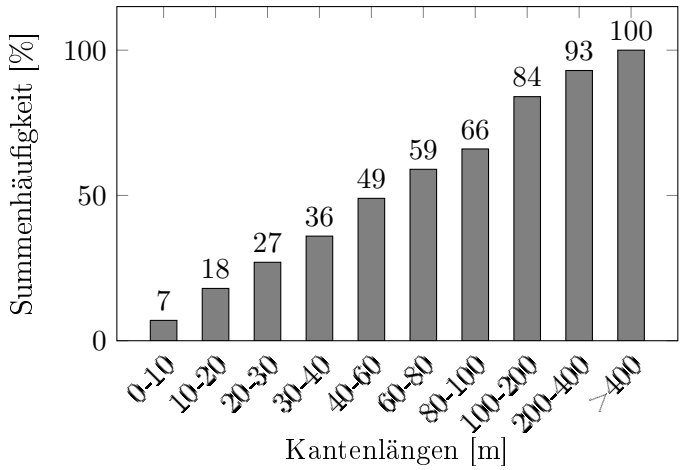
Tabelle 3.5: Statistik Österreichdatensatz Spalte *km*

Anzahl der Elemente	min.	max.	Mittelwert	Median
1.138.351	1 m	10,65 km	132 m	62 m

Abbildung 3.12(a) und Abbildung 3.12(b) geben einen genauen statistischen Aufschluss über die Straßenkantenlängen des OSM-Österreichdatensatz.



(a) Histogramm



(b) Summenhäufigkeit

Abbildung 3.12: Analyse Österreichdatensatz

Da nahezu alle Kanten länger sind als die theoretische Auflösung von 1,5 m des Höhenmodells, müssten für eine exakte Ermittlung der maximalen Steigung jedes einzelne Element des OSM-Datensatzes in Subelemente mit einer Kantenlänge kleiner gleich 1,5 m unterteilt werden. Verläuft eine Subkante wie in Abbildung 3.13, so wird die Rasterzelle Nr. 3 in der Höhenermittlung nicht berücksichtigt, da für das Abgreifen der Information aus dem Höhenmodell der Anfangs- und der Endpunkt herangezogen werden.

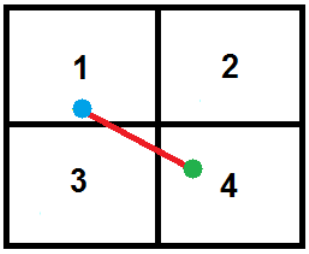


Abbildung 3.13: Subkante

Da Straßen in den seltensten Fällen wirklich Geraden sind und die Ausrichtung zum Höhenmodell beliebig sein kann, wird für ein praxisrelevantes Höhenmodell eine noch höhere Auflösung von Nöten sein. Weiters bedarf es spezieller Algorithmen und Filtermethoden die im Fall von sehr engen Kurven den zur Straße lagerichtigen Höhenwert aus dem Modell entnehmen und Probleme wie in Abbildung 3.13 vermeiden. Hinzu kommt noch die Mess- bzw. Erfassungsungenauigkeit der OSM-Daten. Beispielsweise fehlt es hier an der Metainformation über den Standort des Datenerfassers während dieser Daten für eine Straße generiert. Für die richtige Lage der Straße macht es einen großen Unterschied ob der Datenerfasser am rechten Gehsteig oder in der Fahrbahnmitte die Daten aufgezeichnet hat. Vor diesem Hintergrund dass keines der Höhenmodelle diese theoretische Auflösung besitzt, kann die Höhen- bzw. Steigungsinformation nicht zweckmäßig zur Routenberechnung herangezogen werden und wird in der weiteren Arbeit nicht weiter verfolgt.

Verknüpfen von Straßendaten

Kleine Datensätze bringen neben schnelleren Ladezeiten in der Datenbank auch noch weitere Vorteile, wie gezielteres Aktualisieren und einfacheres Auffinden von Fehlern in den einzelnen Tabellen, mit sich. Umgelegt auf Geodaten bedeutet es, einzelne Länderdatensätze dem Kontinent-Datensatz vorzuziehen (siehe Abschnitt 3.3.1). Liegt das Einsatzgebiet aber nicht innerhalb eines Staatsgebietes, sondern verteilt auf zwei oder mehrere Länder, so ist ein länderübergreifendes Routing mit den von pgRouting (siehe Abschnitt 3.2.4) zur Verfügung gestellten Funktionen nicht möglich. Um ein grenzüberschreitendes Routing trotz alledem zu ermöglichen, wird der größte Datensatz in der Datenbank kopiert und als Ausgangstabelle herangezogen. Im Anschluss daran, werden die restlichen vom Einsatzgebiet betroffenen Ländertabellen der Ausgangstabelle hinzugefügt. Durch das Zusammenfügen der Tabellen beinhalten manche Zeilen möglicherweise selbe Knoten-IDs an den Spaltenpositionen *source* und *target* (siehe Tabelle 3.4), obwohl diese Zeilen bzw. Kanten geografisch eine unterschiedliche Lage aufweisen. Werden diese Fehler nicht behoben, kommt es zu gravierenden Fehlern in der Routenberechnung. Um diesen Fehlern vorzubeugen, bedient man sich dem in Abschnitt 3.4.1 vorgestellten Schema, womit ein routingfähiges Netzwerk erstellt werden kann (siehe Abbildung 3.8(b)). Hierbei ist zu beachten, dass der erste Schritt *OSM-Konvertierung* außer Acht gelassen wird, da man bereits über konvertierte OSM-Daten verfügt. Die pgRouting-Funktion *pgr_createTopology* erstellt dabei aus der zuvor zusammengefügten Kantentabelle, eine Knotentabelle. Die Knoten werden dabei von 1 beginnend, aufsteigend nummeriert. Im Anschluss werden in der Kantentabelle

für jede Zeile die Spalteneinträge *source* und *target*, dies sind die Anfangs- und Endknoten der einzelnen Kantenelemente, der Knotentabelle entsprechend neu gesetzt. Mit der zuvor erstellten Kantentabelle und dem Anwenden des zuvor erwähnten Topologieschemas, liegt nun eine routingfähige multinationale Knoten-Kanten-Struktur vor.

3.4.3 Routing

Steht eine routingfähige Knoten-Kanten-Struktur zur Verfügung (siehe dazu Abschnitt 3.4.1 und Abschnitt 3.4.2), so kann mit dem Routing begonnen werden. Für Routingaufgaben im Rahmen dieser Masterarbeit wird die pgRouting-Funktion *pgr_ astar* verwendet. Die genaue Arbeitsweise des A*-Algorithmus ist unter Abschnitt 2.11.2 detailliert beschrieben. Der Funktionsaufruf für *pgr_ astar* sieht wie folgt aus:

```
pgr_astar(
sql text,
source integer,
target integer,
directed boolean,
has_rcost boolean);
```

In Tabelle 3.6 sind die für die Funktion *pgr_ astar* erforderlichen Parameter samt Datentyp, nochmals aufgelistet.

Tabelle 3.6: Parameter pgr_astar

Parameter	**Datentyp**	**Erklärung**
sql	text	Verweis auf Kantentabelle
source	integer	Startknoten-ID
target	integer	Zielknoten-ID
directed	boolean	Angabe ob es sich um gerichtete Kanten (Bögen) handelt
has_rcost	boolean	Angabe ob eine Rückwärtsbewertung der Kanten vorliegt

- sql: *select id, source, target, cost, x1, y1, x2, y2 ,[reverse_ cost] from Kantentabelle* Dadurch werden die für das Routing relevanten Spalten aus der Kantentabelle selektiert und die Spalte *reverse_ cost* ist dabei optional. Für die Heuristik werden die Spalten *x1, y1, x2 und y2* benötigt.
- source: Startpunkt
- target: Zielpunkt
- directed: Durch Angabe von *true/false* wird der Funktion mitgeteilt, ob es sich um einen gerichteten oder ungerichteten Graphen handelt.
- has_rcost: Wird dieser Parameter auf *true* gesetzt, so muss in der obigen SQL-Anweisung auch die Spalte der Kantenrückbewertung *reverse_ cost* angegeben werden.

Die Ausführung des nachfolgenden SQL-Befehls liefert das Ergebnis im Format *pgr_ costResult* zurück.

```
select pgr_astar('select id, source, target, cost, x1, y1, x2, y2
from Kantentabelle',Startknoten,Zielknoten,false,false);
```

pgr_ costResult ist der Rückgabewert von *pgr_ astar* und beinhaltet das aus der Kantentabelle ermittelte Routingergebnis. Dieser Rückgabewert ist ein Tupel mit vier Parametern (siehe Tabelle 3.7).

Tabelle 3.7: Parameter pgr_costResult

Parameter	**Datentyp**	**Erklärung**
seq	integer	Nummerierung beginnend mit 1
id1	integer	Knoten-ID
id2	integer	Kanten-ID
cost	float	Kosten der Kante

pgr_ costResult liefert als Ergebnis so viele Zeilen, wie Kanten im Routingergebnis enthalten sind. Das Ende der berechneten Route wird durch Hinzufügen einer weiteren Zeile am Ende des Resultats und durch Setzen von *cost* auf 0 und *edge* auf -1 signalisiert. Um das Routingergebnis visualisieren zu können, werden die Werte aus *pgr_ costResult* in der eigenen Tabelle *routing_ result* geschrieben. Anschließend werden die dazugehörigen Geometrieinformationen aus der zuvor verwendeten Kantentabelle herangezogen.

```
insert into routing_result(seq, node, edge, cost)
select seq, id1 as node, id2 as edge, cost from
pgr_astar('select id, source, target, cost, x1, y1, x2, y2
from Kantentabelle',Startknoten,Zielnoten,false,false);
```

Mit dem Kartenwerkzeug Nummer 9 (siehe Abschnitt 3.1.3), der QGIS-Standalone-Anwendung, wird das Routingresultat in der geladenen Karte dargestellt.

4 Ergebnisse

In diesem Kapitel werden unterschiedlich große Datensätze für unterschiedliche Routing - Simulationen herangezogen. Das Grazer Straßennetz wird dabei unter teilweisem Ausschluss von Tunnel und Brücken untersucht. Mit einem steiermarkweitem Straßendatensatz soll analysiert werden, ob die Benutzung von Autobahnen im Vergleich zu Freilandstraßen signifikante Unterschiede in Distanz und Fahrtzeit für LKW mit sich bringt. Weiters wird die Flüchtlingsproblematik vom Sommer 2015 in Südost- und Mitteleuropa beleuchtet. Dabei kann eine Analyse, der von Schleppern benutzen Routen von Ungarn nach Österreich, für die Exekutive durchaus hilfreich sein, um Ressourcen richtig einzusetzen und Schwerpunktkontrollen gezielt, durchführen zu können. Sollten einzelne Länder ihre Grenzen für Flüchtlinge schließen und sich dadurch der Flüchtlingsstrom auf andere Länder verlagern, so könnte dies u.a. für Hilfsorganisationen durchaus relevant sein, um humanitäre Unterstützung rechtzeitig, umfangreich und am richtigen Ort anzubieten zu können.

4.1 Routing mit und ohne Tunnel/Brücken

Den Anforderungen entsprechend, kann es von Nutzen sein, Brücken und/oder Tunnel teilweise oder zur Gänze aus der Routenberechnung auszuschließen. Gründe hierfür können Naturkatastrophen, wie beispielsweise Unwetter, Hochwasser etc. sein, die ein Befahren dieser Bauten unmöglich machen. Auch können Sondertransporte möglicherweise gewisse Brücken und Tunnel[1] aufgrund ihrer Abmessungen nicht Passieren und müssen demnach über eine Ausweichroute geführt werden.
In Abbildung 4.1 sind die Brücken (in rot mit schwarzer Umrandung) und die Tunnel (in grün) innerhalb des Stadtgebiets von Graz dargestellt. Die Masse der Grazer Brücken führt über die Mur, die Graz in einen linken und einen rechten Stadtteil halbiert.

[1] `https://www.bmvit.gv.at/verkehr/gesamtverkehr/gefahrgut/faq/tunnelregelungen.html`

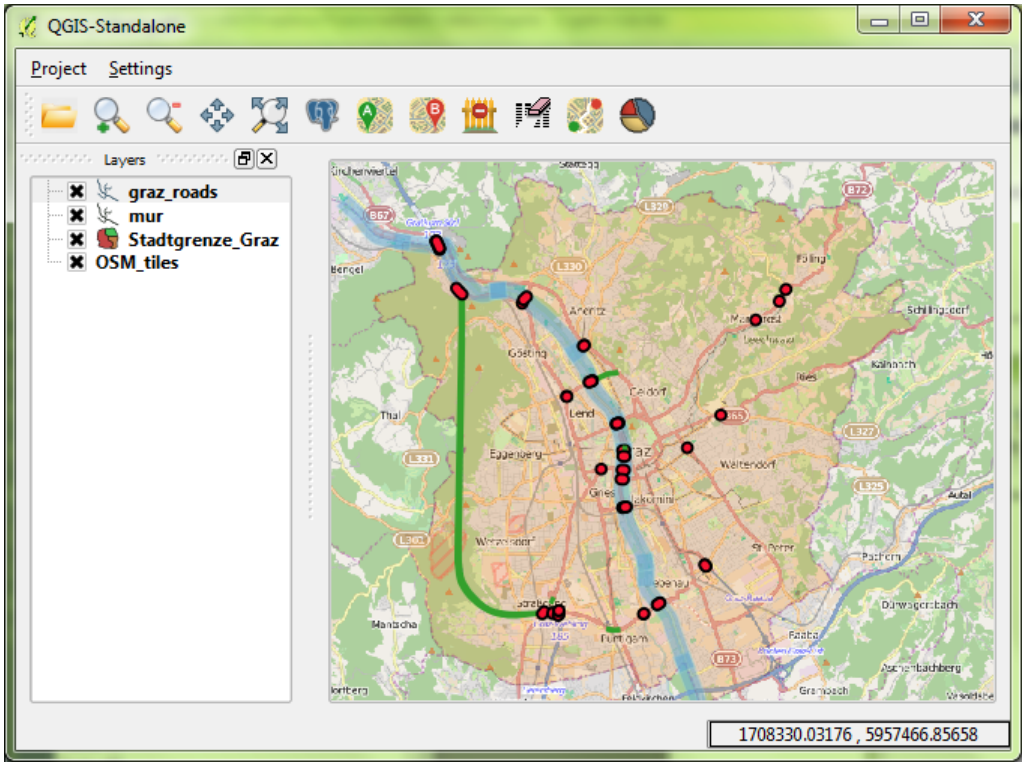
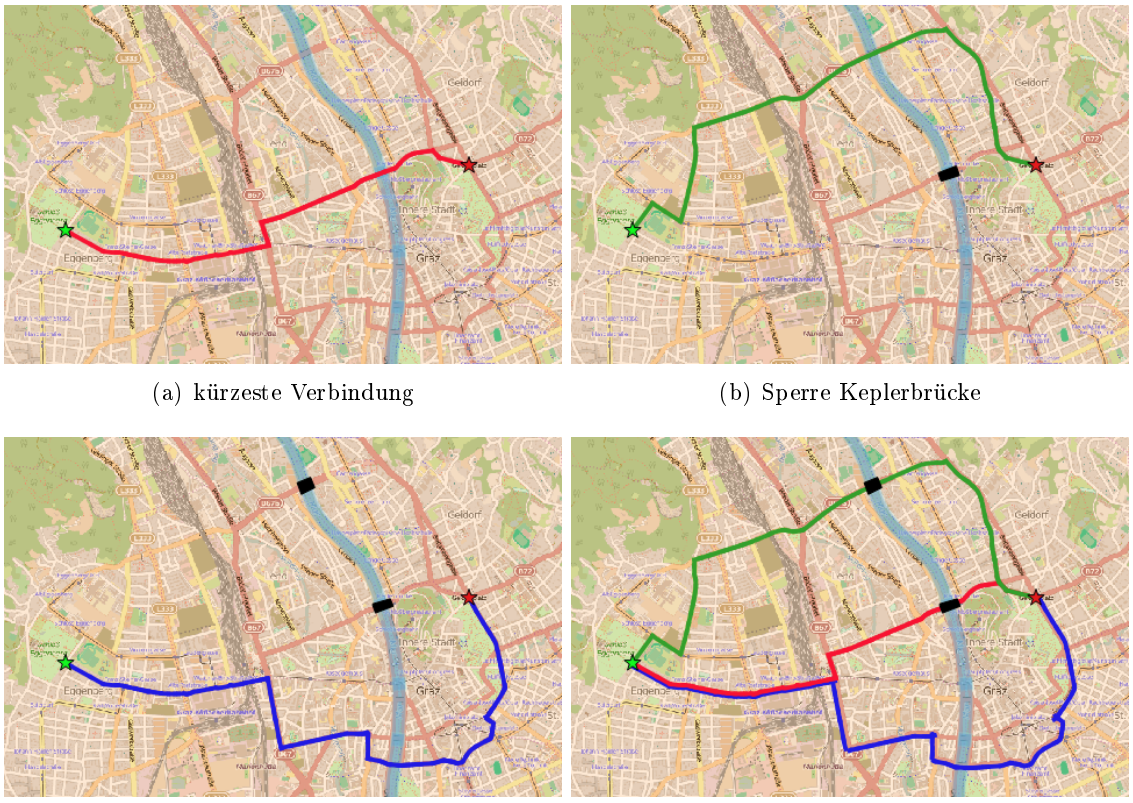


Abbildung 4.1: Tunnel und Brücken im Grazer Stadtgebiet

4.1.1 Blockieren ausgewählter Brücken

Um Brücken aus dem Routing ausschließen zu können, muss die Routing-Option *use bridges* deaktiviert sein (siehe Abschnitt 3.1.3). In den Abbildungen 4.2(a) bis 4.2(c) sind drei unterschiedliche, kürzeste Routen mit identen Start- und Zielpunkten dargestellt. Als Startpunkt dient die Kreuzung vor dem Schloss Eggenberg und als Endpunkt der Geidorfplatz (Kreuzung Glacis/Heinrichstraße/Bergmanngasse). In Abbildung 4.2(a) ist die kürzest mögliche Verbindung, unter Einhaltung der Straßenverkehrsordnung (StVO), von Start- zu Zielpunkt, in rot dargestellt. Ist das Befahren der Keplerbrücke, beispielsweise aufgrund einer Baustelle nicht möglich und wird deshalb von der Routenberechnung ausgeschlossen, so ergibt sich die kürzeste Verbindung von Start- zu Zielpunkt, wie in Abbildung 4.2(b) in grün dargestellt. Das schwarze Polygon stellt dabei den nicht zu befahrenden Bereich im Straßennetzwerk dar. Anstatt der zuvor verwendeten Keplerbrücke, führt die Route nun über die nördlich zur Keplerbrücke gelegene Kalvarienbergbrücke. Erweitert man die Liste der gesperrten Brücken, um die zuvor erwähnte Kalvarienbergbrücke, so sind nun zwei Brücken von der Routenfindung ausgeschlossen und als kürzeste Verbindung ergibt sich, die in Abbildung 4.2(c) blau dargestellte Route. In Abbildung 4.2(d) sind die drei zuvor erwähnten Routen nebeneinander dargestellt.



(a) kürzeste Verbindung

(b) Sperre Keplerbrücke

(c) Sperre Kepler- und Kalvarienbergbrücke

(d) alle Routen

Abbildung 4.2: Blockieren von ausgewählten Brücken

In Tabelle 4.1 sind die unterschiedlichen Distanzen und Fahrtzeiten, der in den Abbildungen 4.2(a) bis 4.2(d) dargestellten Routen, aufgelistet. Die Aussagekraft der Spalte *Zeit*, ist im Vergleich zur Spalte *Distanz*, wesentlich geringer, da es sich hier um die netto Fahrtzeit handelt. Mögliche Stehzeiten an den Kreuzungen werden in der berechneten Route nicht berücksichtigt.

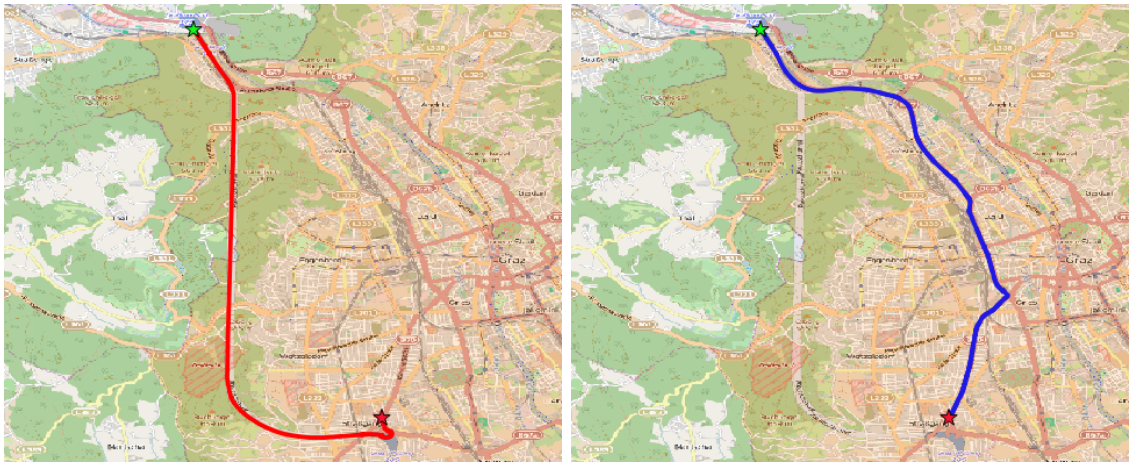
Tabelle 4.1: Kürzeste Verbindungen

Route	Distanz [km]	Zeit [min]
rot	4,2	5
grün	5,6	6
blau	6,2	7

4.1.2 Sperre des Plabutschtunnel

In diesem Beispiel soll eine Alternativroute für den Plabutschtunnel (siehe Abbildung 4.1 - grüne Linie in Nord-Süd-Richtung) gefunden werden. Als Startpunkt dient die Autobahnauffahrt Gratkorn Süd (grüner Stern) und als Zielpunkt die Kärntnerstraße im Bereich Verteilerkreis Webling (roter Stern). Um das in Abbildung 4.3(a) gezeigte Resultat zu erhalten, muss zuvor unter *Settings* und *Routing* die Option *use tunnels* aktiviert sein. Um zu dem in Abbildung 4.3(b) gezeigten Ergebnis zu gelangen, stehen zwei Möglichkeiten

zur Auswahl. Entweder man deaktiviert die Option *use tunnels* oder man zeichnet einen Sperrbereich über den aus dem Routing auszuschließenden Tunnel.



(a) Routing über Plabutschtunnel (b) Routing ohne Plabutschtunnel

Abbildung 4.3: Routing mit und ohne Tunnel

In Tabelle 4.2 sind die Details der in Abbildung 4.3 dargestellten Routen aufgelistet. Zwar ist die Strecke durch das Stadtgebiet kürzer (blaue Route), jedoch ist die Routenführung über die Autobahn und durch den Tunnel die zeitlich schneller Verbindung. Grund dafür ist das höhere Tempolimit.

Tabelle 4.2: Routing mit und ohne Tunnel

Route	Distanz [km]	Zeit [min]
rot	12,4	8
blau	11,4	11

4.2 Vergleich Autobahn zu Freilandstraße mit LKW

Da Tempolimits für LKW auf Autobahnen und Autostraßen nur unwesentlich höher sind (siehe Tabelle 4.3), als die Geschwindigkeitsbegrenzung auf Freilandstraßen, muss für diesen Fahrzeugtyp die schnellste Verbindung von einem Ort A zu einem Ort B nicht unbedingt über solche Straßen führen. Speziell, wenn der Start- und/oder der Zielpunkt der Route weiter entfernt von möglichen Autobahn- oder Autostraßenauffahrten liegt, so kann die schnellste Route durchaus ohne diese Straßentypen auskommen. Abbildung 4.4 gibt einen Überblick über das Autobahn- und Autostraßennetz der Steiermark.

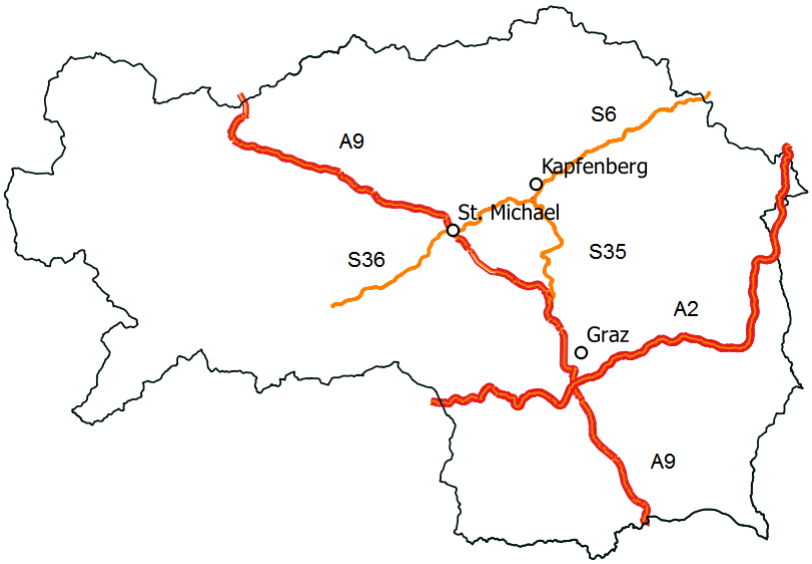


Abbildung 4.4: Autobahnen und Autostraßen in der Steiermark

In Tabelle 4.3 sind die LKW-Tempolimits[2] auf Autobahnen, Autostraßen und Freilandstraßen jenen von PKW gegenübergestellt.

Tabelle 4.3: max. Tempolimits

Straßentyp	**LKW**	**PKW**
Autobahn	80	130
Autostraße	80	100
Freilandstraße	70	100

In Abbildung 4.5 ist die schnellste Route für einen LKW (in rot) und die schnellste Route für einen PKW (in blau) dargestellt. Als Ausgangspunkt dient der Geidorfplatz in Graz (grüner Stern) und das Ziel ist die Stadt Feldbach im Südosten der Steiermark (roter Stern). Wie in der Abbildung zu erkennen ist, führt die PKW-Route ab Graz bis nach Gleisdorf über die Süd-Autobahn. Die LKW-Route hingegen kommt ohne Autobahnbenutzung aus, da trotz höherem Geschwindigkeitslimit auf der Autobahn sich die Gesamtfahrtzeit verlängern würde. Der Grund für die Fahrtzeitverlängerung liegt im Erreichen der Autobahnauffahrt Graz Ost bzw. dem Verlassen der Autobahn ab Gleisdorf bis zum Kreisverkehr Studenzen (ab dort haben beide Routen wieder den selben Streckenverlauf).

[2] `https://www.help.gv.at/Portal.Node/hlpd/public/content/6/Seite.063300.html`

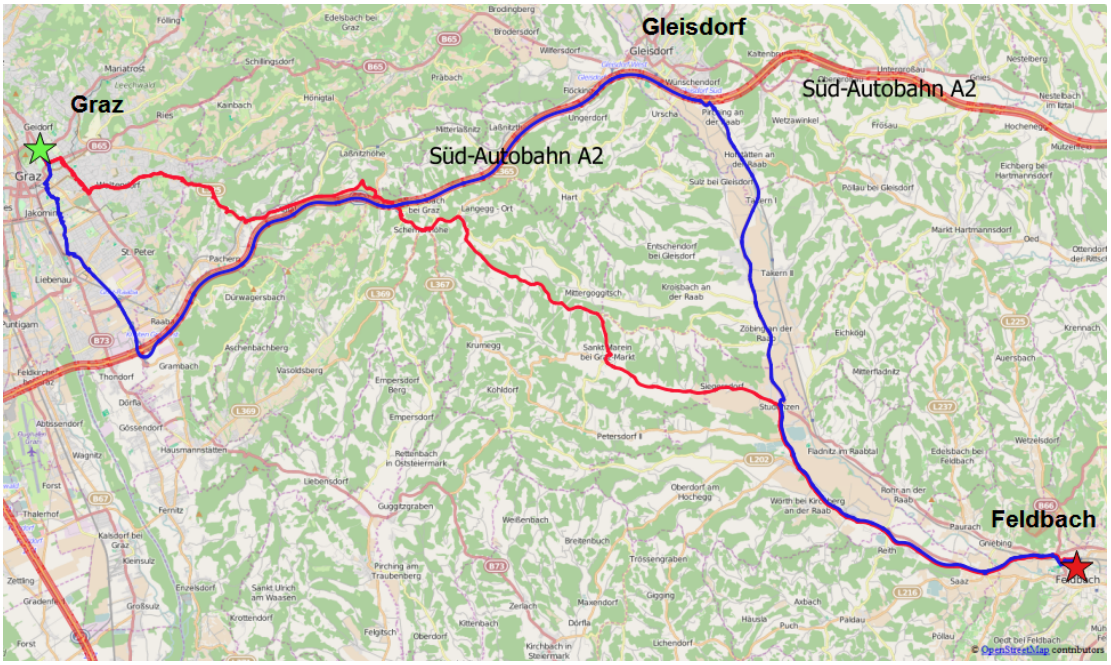


Abbildung 4.5: LKW im Vergleich zu PKW

In Tabelle 4.4 sind Distanzen und Fahrtzeiten der in Abbildung 4.5 dargestellten Routen aufgelistet. Aufgrund der für PKW erlaubten höheren Maximalgeschwindigkeit auf Autobahnen wird die um 10 km längere Strecke von Graz nach Feldbach trotzdem in kürzerer Zeit absolviert.

Tabelle 4.4: Routing Graz-Feldbach

Fahrzeugtyp	Distanz [km]	Zeit [min]
LKW (rot)	44,3	43
PKW (blau)	54,3	37

4.3 Schlepperrouten Ungarn - Österreich

Durch Konflikte im Nahen Osten hat sich die Flüchtlingsproblematik in Mittel- und Osteuropa zugespitzt. Die Hauptroute der Flüchtlinge, die nach Mittel- und Westeuropa strömen, verläuft über die Türkei, Griechenland, Mazedonien, Serbien, Ungarn, Österreich und Deutschland. Da Ungarn Ende August 2015 den Bahnverkehr von Budapest nach Wien eingestellt hat und somit eine Weiterreise der Flüchtlinge per Zug nicht mehr möglich war, begaben sich viele Flüchtlinge in die Hand von kriminellen Schlepperbanden. Einen traurigen Höhenpunkt fand die Schlepperei am 27. August 2015 mit 71 Toten Menschen in einem abgestellten LKW auf der A4 bei Parndorf, Burgenland. Nach diesem tragischen Unglück, wurden die seit 2007 ausgesetzten Grenzkontrollen entlang der österreichisch-ungarischen Grenze, mit dem Ziel die Schlepperei zu bekämpfen, temporär wieder eingeführt. Die Grenzübergänge zwischen Österreich und Ungarn, welche mit einem PKW befahren werden können, sind in Abbildung 4.6 ersichtlich.

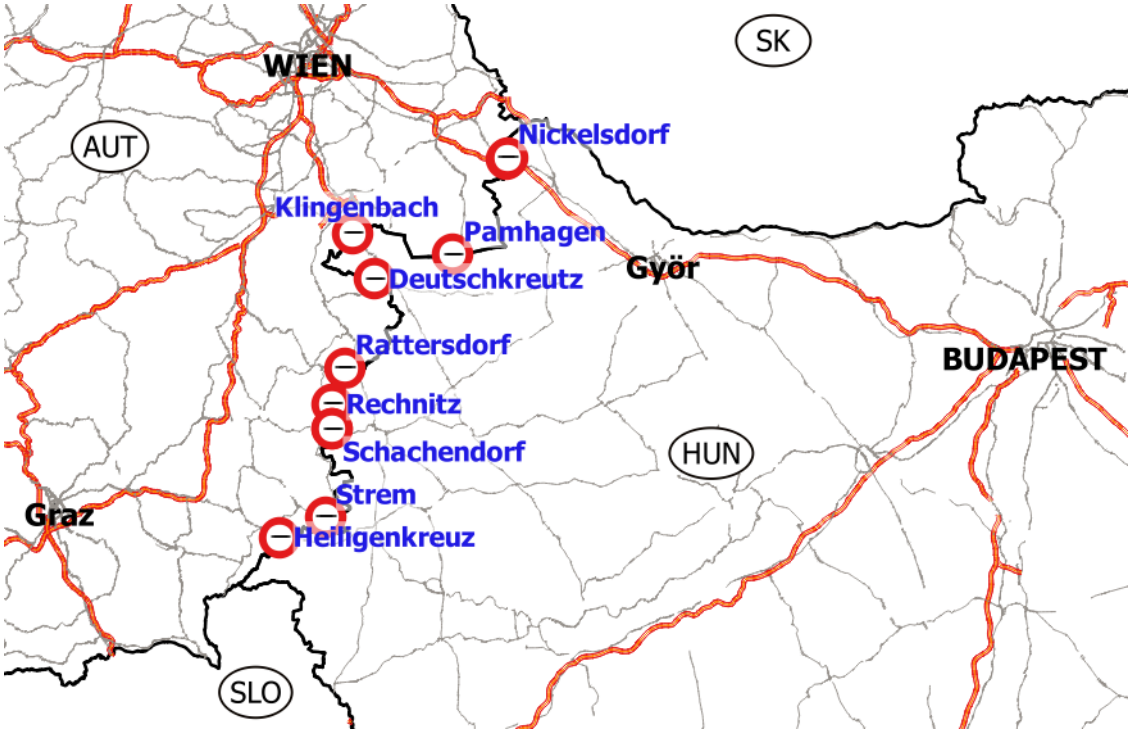


Abbildung 4.6: Grenzübergänge zu Ungarn

Mit dieser Anwendung lassen sich mögliche Ausweichrouten der Schlepperbanden, auf andere Grenzübergänge zwischen Österreich und Ungarn, simulieren. Für Schlepper ist nicht unbedingt der kürzester Weg der Optimale, sondern der, wo sie unerkannt ihr illegales Geschäft ausüben können. Die Wegstrecke und die sich daraus ergebende Fahrtzeit haben jedoch auch einen großen Anteil am Erfolg der Schlepper, da sich mit zunehmender Fahrtdauer auch die Wahrscheinlichkeit eines Aufgriffes erhöht.
In Tabelle 4.5 sind Distanz und Fahrtzeit von Budapest zu allen 9 in Abbildung 4.6 ersichtlichen Grenzübergänge aufgelistet. Die schnellste Route verläuft von Budapest über Győr zu dem Grenzübergang Nickelsdorf. In der letzten Spalte der Tabelle 4.5 ist der relative Fahrtzeitunterschied zur Wegstrecke Budapest - Nickelsdorf angeführt.

Tabelle 4.5: Budapest - Grenzübergang

Nr.	Grenzübergang	Distanz	Zeit	Δt zu Nickelsdorf
1	Nickelsdorf	170 km	1 h 44 min	—
2	Pamhagen	180 km	2 h 2 min	18 min
3	Deutschkreutz	192 km	2 h 10 min	26 min
4	Klingenbach	205 km	2 h 22 min	38 min
5	Rattersdorf	210 km	2 h 35 min	51 min
6	Rechnitz	218 km	2 h 43 min	59 min
7	Schachendorf	228 km	2 h 53 min	1 h 9 min
8	Strem	287 km	3 h 33 min	1 h 49 min
9	Heiligenkreuz	289 km	3 h 36 min	1 h 52 min

Die Reihenfolge der Grenzübergänge in Tabelle 4.5 wurde nach der Fahrtzeit festgelegt, da es sich hierbei um das routingrelevante Kriterium handelt. Teilt man die Spalte **Δt zu Nickelsdorf** von Tabelle 4.5 in drei annähernd gleich große Klassen mit einem Intervall von ca. 31 Minuten ein, so ergeben sich drei Gruppen von Grenzübergängen wie in Tabelle 4.6 aufgelistet.

Tabelle 4.6: Klassifizierung Grenzübergänge

Klasse	Zeit [min]	Grenzübergang
1	18 - 50	Pamhagen, Deutschkreutz, Klingenbach
2	51 - 80	Rattersdorf, Rechnitz, Schachendorf
3	81 - 112	Strem, Heiligenkreuz

Die in Tabelle 4.5 aufgelisteten und in Tabelle 4.6 klassifizierten Fahrtzeiten von Budapest zu den österreichisch-ungarischen Grenzübergängen sind in Abbildung 4.7 dargestellt. Durch die Verwendung von unterschiedlichen Schrift- und Symbolgrößen, ist ein leichtes Erkennen der einzelnen Klassen gegeben.

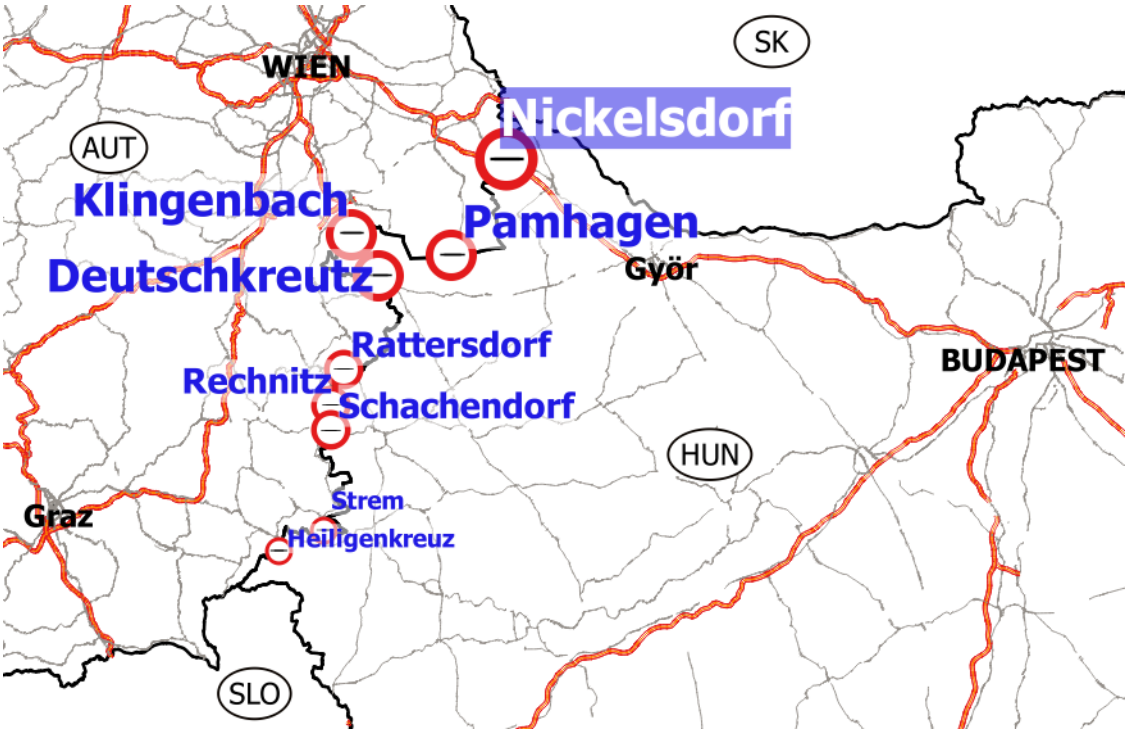


Abbildung 4.7: Grenzübergänge zu Ungarn

Kombiniert man die zuvor erläuterte Erreichbarkeitsanalyse mit der aktuellen Schlepperstatistik der einzelnen Grenzübergänge und der Expertise der Exekutive, so könnten neue Schwerpunktkontrollen an den sich ergebenden Hotspots ressourceneffizient durchgeführt werden.

4.4 Verlagerung der Balkanroute

Als Balkanroute, werden zusammenfassend Routen zwischen dem Nahen Osten und Europa, die über den Balkan führen, bezeichnet. Flüchtlinge, die diese Route nehmen kommen meist aus Syrien, dem Irak und aus Afghanistan. Über die Türkei gelangen die Flüchtlinge nach Griechenland, von wo sie über Serbien weiter nach Ungarn, Österreich und Deutschland reisen. Bei Ungarn und Serbien handelt es sich um Haupt-Transitländer, da die Masse der Flüchtlinge diese Länder durchquert, aber dort nicht bleiben möchte. In der nachfolgenden Simulation soll gezeigt werden, wie sich die Balkanroute auf andere Länder verlagert, falls Staatsgrenzen entlang der Route für Flüchtlinge geschlossen werden. Abbildung 4.8 zeigt einen Teilabschnitt der Balkanroute ab der Grenze Mazedonien/Serbien über Ungarn und Österreich bis nach München. Die rote Linie stellt eine von den Flüchtlingen gewählte Hauptroute, vor der Grenzschließung Ungarns zu Serbien, dar. Die blaue Linie in der Grafik repräsentiert die kürzeste Verbindung von Mazedonien/Serbien bis nach München.

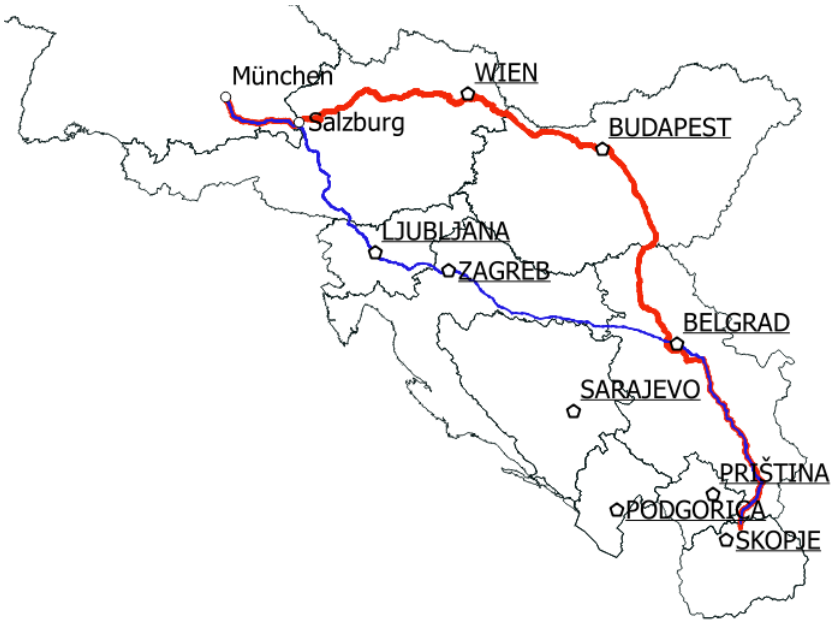


Abbildung 4.8: Balkanroute

Wie in Abbildung 4.8 ersichtlich, haben die rote und die blaue Linie bis Belgrad nahezu den selben Verlauf. Im Vergleich zur Balkanroute (rot) verläuft die kürzeste Verbindung (blau) ab Belgrad über Kroatien und Slowenien bis nach Salzburg. Von Salzburg bis nach München ist der Routenverlauf wieder ident. Der Längenunterschied zwischen beiden Routen beträgt rund 120 km (siehe Tabelle 4.7). Die Ursache für die Wahl der Balkanroute, anstatt der kürzesten Verbindung, kann nicht eindeutig belegt werden. Eine mögliche Begründung ist, dass diese Route schon seit Jahrzehnten von Emigranten und Schleppern in Anspruch genommen wird und diese sich wahrscheinlich ein gut funktionierendes Netzwerk entlang der Strecke aufgebaut haben. Auch wird die Korruption der Behörden in den Ländern entlang der Balkanroute ausgenutzt, um Menschen ungehindert hindurch zu schleusen. Am

Beispiel des hier gezeigten Teilabschnittes der Balkanroute nehmen Flüchtlinge einen Umweg von rund 120 km in Kauf, wenn dadurch die Wahrscheinlichkeit für das Erreichen ihres Zieles steigt. Relativ ausgedrückt bedeutet ein Mehrweg von 120 km, im Verhältnis zur kürzesten Verbindung, lediglich eine Erhöhung der Distanz um rund 8 % auf diesem Teilabschnitt.

Tabelle 4.7: Streckenlängen

Route	**Distanz [km]**
Balkanroute (rot)	1613
kürzeste Verbindung (blau)	1489

In Abbildung 4.9 ist die kürzeste Verbindung (in blau) und die schnellste Verbindung (in rot) von Mazedonien nach München dargestellt. Wie aus der Grafik hervor geht, unterscheiden sich beide Route nicht wesentlich. Leichte Unterschiede ergeben sich um Belgrad, Zagreb und Ljubljana. Haben Flüchtlinge die Wahl zwischen der kürzesten und der schnellste Verbindung von einem Ort A zu einem Ort B, so dürften sie sich wahrscheinlich für den kürzeren, als den schneller Weg entscheiden. Begründet kann dies dadurch werden, dass Flüchtlinge selten die gesamte Route von Schleppern transportiert werden und ihre Wege eher zu Fuß und mit öffentlichen Verkehrsmitteln zurücklegen. Aus diesem Grund macht der kürzere Weg mehr Sinn, da die zurückzulegende Distanz und nicht die maximal mögliche Geschwindigkeit entlang der Route, das entscheidende Kriterium ist.

Abbildung 4.9: kürzeste und schnellste Verbindung

In Tabelle 4.8 sind die Längen, der in Abbildung 4.9 dargestellten kürzesten und schnellste Verbindung, aufgelistet. Aufgrund von Mess- und Erfassungsungenauigkeiten des Wegenetzes der OSM, ist der Längenunterschied der beiden Routen nicht signifikant und es

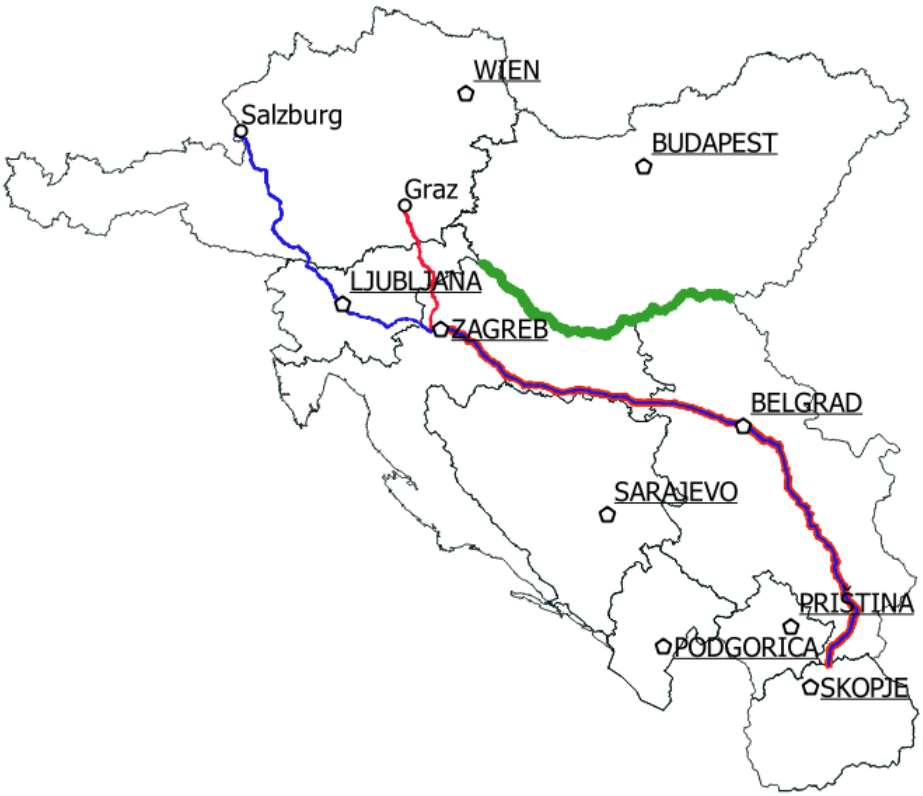
kann aus diesem Grund die kürzeste Verbindung zugleich auch als schnellste Verbindung angesehen werden.

Tabelle 4.8: Schnellste und kürzeste Verbindung

Route	Distanz [km]
schnellste (rot)	1469
kürzeste (blau)	1489

4.4.1 Grenzschließung Ungarn-Serbien und Ungarn-Kroatien

Schließt Ungarn aufgrund des Flüchtlingstroms seine Grenzübergänge zu Serbien und Kroatien, so könnte eine mögliche Verlagerung der Balkanroute in Richtung Mittel- und Westeuropa, wie in Abbildung 4.10 dargestellt, aussehen. In grün sind die gemeinsamen Grenzen zwischen Ungarn und seinen Nachbarländern Serbien sowie Kroatien hervorgehoben. Die in blau dargestellte Verbindung, zwischen Serbien/Mazedonien und Salzburg entspricht dabei auch der in Abbildung 4.9 in blau dargestellten kürzesten Verbindung von Serbien/-Mazedonien, nach Salzburg. Auf dieser sich möglicherweise neu ergebenden Balkanroute (rote Linie), ist Graz die am nächstgelegene österreichische Landeshauptstadt.



1

Abbildung 4.10: Verlagerung der Route im Falle einer Grenzschließung Ungarn-Serbien und Ungarn-Kroatien

4.4.2 Auswirkungen der Routenänderung auf Österreich

Durch die in 4.4.1 dargestellte Grenzschließung und der sich daraus neu ergebenden Balkanroute, verlagert sich der Flüchtlingsstrom von der Ost- an die Südgrenze Österreichs. Nachfolgend werden die Grenzübergänge von den Bundesländern Steiermark und Kärnten, hinsichtlich ihrer Erreichbarkeit, analysiert. Ausgangspunkt der Routensimulation ist Zagreb, da die neue Balkanroute, egal ob sie über Graz oder Salzburg und dann weiter nach München führt, bis dahin den selben Verlauf aufweist (siehe Abbildung 4.10). Als Ziel dient jeweils die Landeshauptstadt, der zu untersuchenden Bundesländer. Für die Erreichbarkeitsanalyse werden aufgrund der Übersichtlichkeit nur die in Abbildung 4.11 dargestellten Grenzübergänge herangezogen. Einsatzkräften soll dadurch die Möglichkeit geboten werden, aus einer Vielzahl an denkbaren Routen, die wahrscheinlichsten herauszufiltern und dabei das Schwergewicht ihrer Arbeit darauf zu verlagern. Zusätzlich können dadurch Ausrüstungsgegenstände wie Feldlazarette, Zelte und mobile Küchen zielgerichtet und effektiv eingesetzt werden.

Abbildung 4.11: kärntnerisch-slowenische und steirisch-slowenische Grenzübergänge

Möglicher Routenverlauf nach Graz

Durch die neue Balkanroute könnte der Flüchtlingsstrom auch die steirische Landeshauptstadt erreichen. In Abbildung Abbildung 4.12 sind die vier Grenzübergänge Spielfeld, Mureck, Bad Radkersburg und Sicheldorf und die darüber verlaufenden Routen von Zagreb nach Graz farblich verschieden dargestellt. Je breiter die Linie, desto kürzer ist die Strecke zwischen Zagreb und Graz. Auffallend ist, dass bis Ptuj alle vier Route den selben Verlauf haben. Danach splitten sie sich auf.

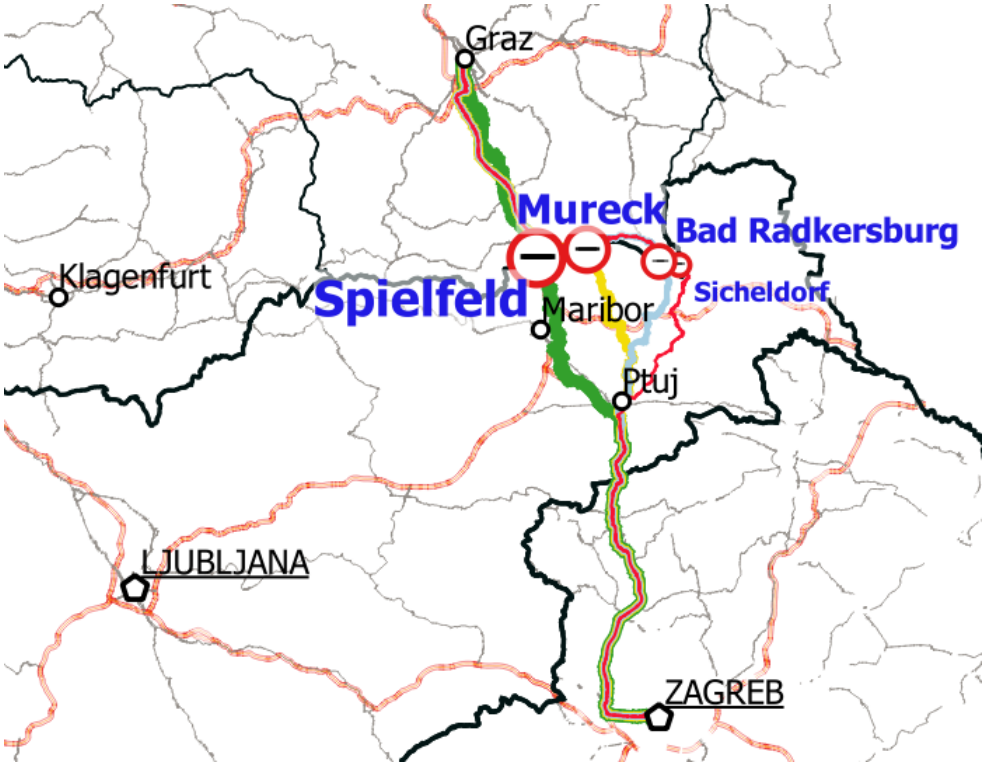


Abbildung 4.12: Erreichbarkeit Zagreb-Graz

In Tabelle 4.9 sind die Distanzen zu den in Abbildung 4.12 dargestellten vier Routen aufgelistet. Über den Grenzübergang Spielfeld beträgt die Distanz der kürzesten Verbindung Zagreb-Graz 178 km und über Sicheldorf 213 km. Aus dieser Simulation geht hervor, dass mögliche Flüchtlingsrouten von Zagreb nach Graz eher über Spielfeld und Mureck, als über Sichelsdorf und Bad Radkersburg verlaufen.

Tabelle 4.9: Zagreb-Graz über Grenzübergang

Nr.	Farbe	Grenzübergang	Distanz [km]
1	grün	Spielfeld	178
2	gelb	Mureck	188
3	hellblau	Bad Radkersburg	206
4	rot	Sicheldorf	213

Möglicher Routenverlauf nach Klagenfurt

In dieser Untersuchung soll gezeigt werden, wie sich der Flüchtlingsstrom, ausgehend von Zagreb nach Klagenfurt, auf die Grenzübergänge zwischen Kärnten und Slowenien auswirken kann. In Abbildung 4.13 sind die kürzesten Verbindungen von Zagreb nach Klagenfurt für jeden der sieben Grenzübergänge dargestellt. Die kürzeste, aller über einen der sieben Grenzübergänge, verlaufende Route, ist in grün und die längste in rot gezeichnet. Die zweitlängste Route durch den Karawankentunnel ist hellblau und die verbleibenden vier Strecken sind gelb eingefärbt.

Abbildung 4.13: Erreichbarkeit Zagreb-Klagenfurt

In Tabelle 4.10 ist die kürzeste Verbindung über jeweils einen der sieben in Abbildung 4.13 dargestellten Grenzübergänge aufgelistet. Über den Grenzübergang Loibltunnel beträgt die Distanz zwischen Zagreb und Klagenfurt 221 km und ist somit um 60 km kürzer als die Strecke Zagreb-Wurzenpass-Klagenfurt. Interessant ist, dass Strecken über die Grenzübergänge Nr. 2 bis Nr. 5 sich um maximal 3 km unterscheiden.

Tabelle 4.10: Zagreb-Klagenfurt über Grenzübergang

Nr.	Farbe	Grenzübergang	Distanz [km]
1	grün	Loibltunnel	221
2	gelb	Lavamünd	232
3	gelb	Bleiburg	233
4	gelb	Leifling	234
5	gelb	Raunjak	235
6	hellblau	Karawanken	253
7	rot	Wurzenpass	281

5 Resümee und Ausblick

Rückblickend auf die Arbeit ergeben sich einige Punkte an die mit dem jetzigen Wissensstand anders herangegangen werden könnte. Ein Beispiel dafür ist der Einsatz des osm2po-Konverters, durch welchen viele wichtige Metainformationen über die Straßen verloren gehen, die nachträglich wieder angefügt werden müssen. Im Speziellen soll hier das Tunnel- und Brückenattribut hervorgehoben werden. Hier könnte durch eine Eigenentwicklung eine individuell abgestimmte und maßgeschneiderte Lösung geschaffen werden, die nur routingrelevante Attribute beinhaltet. Auch sind die OSM-Daten von Geofabrik, welche als Basisdaten in der hier entwickelten Anwendung zum Einsatz kommen, nicht zur Gänze fehlerfrei und bereiten so manches Mal Kopfzerbrechen. Auch in diesem Fall wäre es die Mühe und Anstrengung wert, eine eigene Software zu entwickeln, welche die gewünschten Straßendaten samt Metainformationen direkt von einem der OSM-Server lädt. Mit viel Erfahrung in der Bildverarbeitung, Algorithmik und speziellen Kenntnissen von Filtertechniken ließe sich vielleicht auch das Eine oder Andere in Richtung steigungsabhängigen Routings erreichen. Abschließend bleibt noch die Überlegung bezüglich einer Erweiterung in Richtung Mehrbenutzersystem, sodass auch zwei oder mehr Benutzer zur gleichen Zeit mit ein und demselben Datensatz unterschiedliche Routingoperationen ausführen können.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ALS	Airborne Laserscanning
API	Application programming Interface
ASTER	Advanced Spaceborne Thermal Emission and Reflection
BSD	Berkley Software Distribution
CC	Creative Commons
CSW	Catalogue Service for the Web
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
DGM	Digitales Geländemodell
EPSG	European Petroleum Survey Group Geodesy
EVU	Energieversorgungsunternehmen
FS	Fremdschlüssel
GDAL	Geospatial Data Abstraction Library
GIS	Geoinformationssystem
GPL	General Public License
GPS	Global Positioning System
GUI	Graphical User Interface
LIS	Landinformationssystem
LKW	Lastkraftwagen
NASA	National Aeronautics and Space Administration
NIS	Netzinformationssystem
OGC	Open Geospatial Consortium
OGR	OpenGIS Simple Features Reference Implementation
OODB	Objektorientierte Datenbank
OOP	Objektorientierte Programmierung
ORDB	Objektrelationale Datenbank
ORDBMS	Objektrelationales Datenbankmanagementsystem

OSM	OpenStreetMap
PKW	Personenkraftwagen
PS	Primärschlüssel
QGIS	Quantum Geoinformationssystem
QL	Query Language
RDB	Relationale Datenbank
RDMBS	Relationales Datenbankmanagementsystem
SQL	Structured Query Language
SRID	Spatial Reference Identifier
SRTM	Shuttle Radar Topography Mission
StVO	Straßenverkehrsordnung
TIFF	Tagged Image File Format
TRSP	Turn restriction Shortest Path
TSP	Traveling Salesman Problem
UIS	Umweltinformationssystem
UTM	Universal Transverse Mercator
WFS	Web Feature Service
WGS84	World Geodetic System 1984
WKB	Well Known Binary
WKT	Well Known Text
WMS	Web Map Service
WPS	Web Processing Service
XML	Extensible Markup Language

Literaturverzeichnis

N. Bartelme. *Geoinformatik: Modelle · Strukturen · Funktionen.* Springer Berlin Heidelberg, 2013. ISBN 9783662074381.

E.W. Dijkstra. *A Note on Two Problems in Connection with Graphs: (numerische Mathematik, _1(1959), P 269-271).* Stichting Mathematisch Centrum. Rekenafdeling, 1959.

B. Hofmann-Wellenhof, H. Lichtenegger, and K. Legat. *Navigation: Principles of Positioning and Guidance.* Springer Vienna, 2011. ISBN 9783709160787.

Lott R. Iliffe J. *Datums and Map Projections for Remote Sensing, GIS, and Surveying.* Datums and Map Projections for Remote Sensing, GIS, and Surveying, 2nd Edition. CRC Press, 2008. ISBN 9781904445470.

Ramm und Topf. *OpenStreetMap: Die freie Weltkarte nutzen und mitgestalten.* Lehmanns, 2010. ISBN 9783865416353.

M. Schubert. *Datenbanken: Theorie, Entwurf und Programmierung relationaler Datenbanken.* Vieweg+Teubner Verlag, 2007. ISBN 9783322921130.

A. Steger. *Diskrete Strukturen 1: Kombinatorik, Graphentheorie, Algebra.* Springer-Lehrbuch. Springer Berlin Heidelberg, 2013. ISBN 9783662081334.

N. Stones, R. und Matthew. *Beginning Databases with PostgreSQL: From Novice to Professional.* Books for professionals by professionals. Apress, 2005. ISBN 9781430200185.

T. Theis. *Einstieg in Python: Ideal für Programmieranfänger geeignet.* Galileo Computing. Rheinwerk, 2014. ISBN 9783836228619.