Routingfunktionalitäten in einer WebMapping-Anwendung basierend auf OpenStreetMap-Daten

Kai Behncke und Manfred Ehlers

Zusammenfassung
Routinganwendungen im Web gibt es mittlerweile »en Masse«. In der Regel jedoch sind diese proprietärer Art und gewähren keinen Einblick in dahinter liegenden Quellcode bzw. entsprechende Algorithmen. Eine individuelle Anpassung ist zumeist unmöglich. Zudem basieren diese häufig auf proprietären Geodaten. Zum Zwecke der Lehre und Ausbildung – und auch um Unabhängigkeit von externen Anbietern zu erlangen – bietet es sich an, eine webbasierte Routinganwendung auf Basis freier Geodaten von OpenStreetMap und Open Source Software zu nutzen. Innerhalb dieses Artikels wird beschrieben, wie mittels OpenStreetMap-Daten, der »pgRouting«-Bibliothek und der Datenbank PostgreSQL/PostGIS sowie unterschiedlicher Skriptsprachen ein individuelles Routing in WebMapping-Anwendungen realisiert werden kann.

Summary
Within the web a large number of routing applications exist. The majority of these applications are proprietary developments and do not provide the possibility to explore the background/source code and/or the specific algorithms. For most cases, an individual adaptation of these applications is not possible. Furthermore, the most of these applications are based on proprietary geodata. For the purpose of teaching and education – and also to be independent from external providers – a web based routing application based on OpenStreet Map data und Open Source software is of great advantage. This article describes how an individual routing in webmapping applications can be realized using OpenStreetMap data, the pgRouting library and the Open Source database management system PostgreSQL/PostGIS together with different scripting languages.

Schlüsselwörter: OpenStreetMap, Routing, MapFish, OpenLayers

1 Einleitung und Hintergründe

Im Rahmen einer Dissertations-Fallstudie (Behncke 2011) wurde unter anderem untersucht, welche Funktionalitäten innerhalb einer WebMapping-Anwendung für den Anwender besonders wichtig sind. Es wurde beispielsweise evaluiert, dass Routingfunktionen innerhalb der entwickelten Applikation eine besonders hohe Wichtigkeit für die Nutzer aufweisen (61,3 % von 88 antwortenden Personen gaben an, dass diese Funktion »sehr wichtig/wichtig« sei, siehe Abb. 1, eine ausführliche Schilderung der Ergebnisse findet sich auch bei Behncke, Brinkhoff, Ehlers 2012).

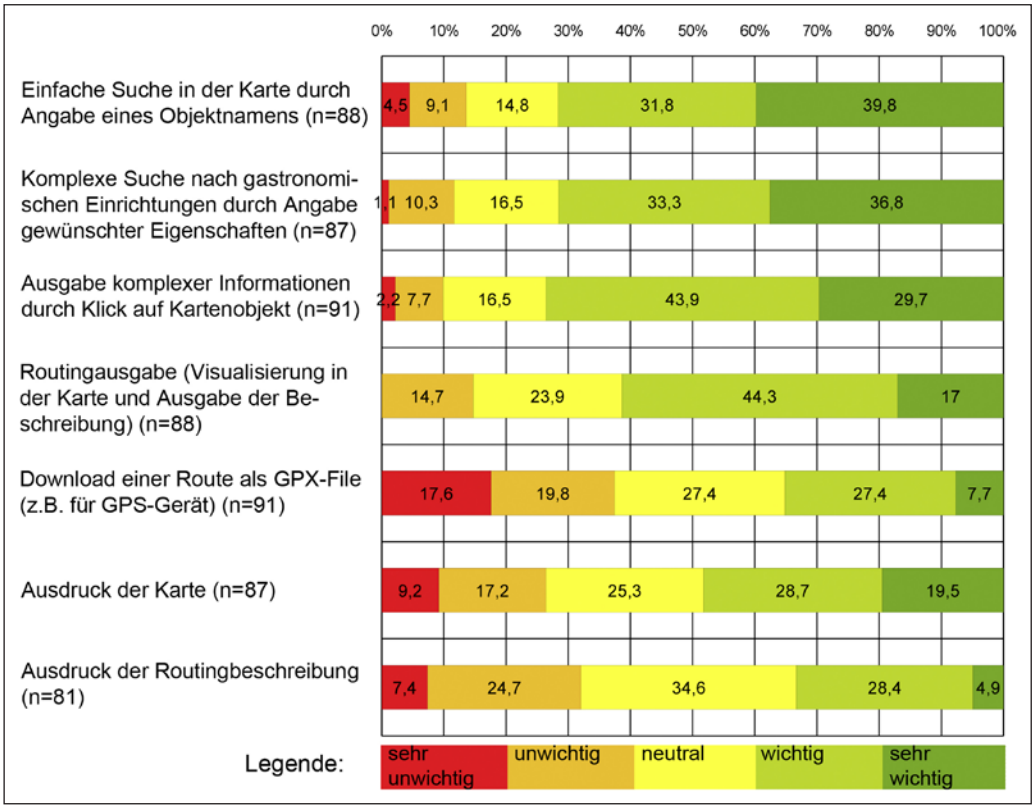

Abb. 1: Wichtigkeit verschiedener Funktionen innerhalb einer WebMapping-Anwendung

Dieses Ergebnis deckt sich mit einem Teilergebnis der wenigen verfügbaren empirischen Untersuchungen über WebMapping-Anwendungen (vgl. Wachowicz et al. 2008; Nivala et al. 2008).

Es existieren aktuell diverse Routinganwendungen unterschiedlicher Anbieter in WebMapping-Anwendungen (z. B. Google Maps, Map24, Falk.de, Bing Maps), welche sich zwar leicht nutzen lassen, dem Anwender jedoch aufgrund des proprietären »Closed Source«-Prinzips keine Möglichkeit geben, in dahinter liegende Algorithmen Einblick zu erlangen, bzw. diese technisch an eigene Daten bzw. Ansprüche anzupassen.

Eine (partielle) Ausnahme ist der Routenplaner »Open RouteService«. Dieser freie Dienst (der allerdings nicht als Open Source Software anzusehen ist) ermöglicht ein Routing basierend auf OpenStreetMap-Daten. »Dieser Service ist sowohl für den Endbenutzer über ›openrouteservice.org‹ als auch für andere Anwendungen und Clients über eine entsprechende Api verfügbar« (Franke et al. 2011). Basierend auf dem OpenRouteService wurde über einen erweiterten Routing-Algorithmus beispielsweise die effizienteste Route bezüglich einer Streckensteigung errechnet (sinnvoll für Elektrofahrzeuge) (Franke et al. 2011). Ein weiteres Projekt, basierend auf dem OpenRouteService und OpenStreetMap-Daten, stellt ein Routenplaner für Rollstuhlfahrer dar (Müller et al. 2010).

Möchte man nun allerdings Unabhängigkeit von externen Anbietern bzw. Services erlangen und zudem Freie Geodaten und Open Source Software nutzen, so bieten sich OpenStreetMap-Daten und die Routingbibliothek »pgRouting« an (www.pgrouting.org). PgRouting offeriert Routingfunktionalitäten für Daten, welche in einer PostgreSQL/PostGIS-Datenbank gehalten werden. PgRouting unterstützt beispielsweise den Dijkstra-Algorithmus (ein Algorithmus, welcher den kürzesten Pfad zwischen einem gegebenen Startknoten und weiteren Knoten in einem (gewichteten) Graphen ermittelt).

Ein wichtiges Argument für die Verwendung von OpenStreetMap-Daten ist neben der Tatsache, dass für die Datennutzung keine monetären Investitionen nötig sind, die hohe Quantität und Qualität der Daten (vgl. Ludwig et al. 2010; Zielstra, Zipf 2010; Amelunxen 2010; Schoof et al. 2011; Heiken, Peyke 2011; Stengel, Pomplun 2011).

Innerhalb der erwähnten Dissertation wurde eine entsprechende Routing-Anwendung für Fußgänger in eine WebMapping-Applikation (www.osnago-karte.de) des gastronomischen Webportals »OsnaGo« (www.osnago.de) integriert (vgl. zu dem OsnaGo-System und seinen Komponenten Behncke 2010a; Behncke 2010b). Die entsprechende Umsetzung soll nachfolgend beschrieben werden. Der Quellcode dieser Lösung kann unter der »GNU LGPL«-Lizenz unter www.osnago.de heruntergeladen werden.

Ein Nutzer kann in der Applikation einen Start- und Endpunkt setzen und sich darauf basierend die kürzeste Route anzeigen lassen. Ergänzend erhält der Nutzer eine Routingbeschreibung sowie die Möglichkeit, sich die Route als GPX-Datei herunterzuladen.

Um eine funktionierende Routing-Anwendung auf der Basis von pgRouting zu erstellen, sind einige »Fallstricke« und Besonderheiten zu beachten, welche nachfolgend beschrieben werden.

2 Umsetzung einer Routinganwendung basierend auf pgRouting und OpenStreetMap-Daten

Prinzipiell müssen OpenStreetMap-Daten, welche originär keine Topologie enthalten, zuerst entsprechend optimiert werden. Aus dem Datenpool von Niedersachsen wurde zunächst einmal der Ausschnitt von Osnabrück herausgefiltert (über das Tool »Osmosis«) und die Daten dann über die Software »osm2pgrouting« in eine PostgreSQL/PostGIS-Datenbank transferiert.

Entscheidend für den Routingalgorithmus und die oben beschriebenen Funktionalitäten sind dann die Spalten »gid« (eindeutige Id), der Name einer Straße (wichtig für die Routingbeschreibung), die Liniengeometrie selber (an dieser Stelle als WKT dargestellt) sowie die den Straßen zugewiesenen Source- und Target-Werte (siehe Abb. 2).

Die Kanten (Straßen) sind durch Anfangs- (Source) und Endwert (Target) gerichtet. Sich berührende Kanten besitzen jeweils dieselbe Id (Source- oder Target-Id) (siehe Abb. 3).

	gid integer	name character(20	source integer	target integer	astext text
230	612	Nitzestraße	343	344	MULTILINESTRING((893161.78204227 6852720.31969063,893221.794379757 6852794.1549593,893281.80
231	616	Henschelstraße	339	345	MULTILINESTRING((898540.439274776 6851317.25051212,898524.698698778 6851250.65260546,898507.
232	618		346	347	MULTILINESTRING((893992.503742315 6852149.44319625,894001.542884967 6852168.78780264,894061.
233	620	Lüstringer Straß	348	349	MULTILINESTRING((897980.157145665 6850266.41423966,898110.423213791 6850255.98916847,898187.
234	628	Berghoffstraße	330	343	MULTILINESTRING((892860.562632133 6852909.10418703,892975.633589766 6852853.35833882,893025.
235	629	Berghoffstraße	343	341	MULTILINESTRING((893161.78204227 6852720.31969063,893251.216121173 6852646.61249274))
236	630	Berghoffstraße	341	350	MULTILINESTRING((893251.216121173 6852646.61249274,893393.749597185 6852526.91707888,893426.
237	631	Berghoffstraße	350	346	MULTILINESTRING((893755.315303282 6852285.7299393,893853.799656786 6852222.76368544,893931.0
238	635		351	352	MULTILINESTRING((897857.048920796 6843458.14683716,898072.46326743 6843490.14106469,898222.0
239	637		346	353	MULTILINESTRING((893992.503742315 6852149.44319625,893985.000808635 6852057.28853081,893962.
240	639		354	351	MULTILINESTRING((897781.874868663 6843669.72920013,897852.662932859 6843489.94110037,897857.
241	641	Schinkelstraße	302	355	MULTILINESTRING((898351.351987714 6850487.3998097,898269.409710541 6850488.94632975))
242	642	Schinkelstraße	355	356	MULTILINESTRING((898269.409710541 6850488.94632975,898233.27540383 6850481.55942485,898126.7
243	647	Kanonenweg	357	349	MULTILINESTRING((898128.557158841 6850094.3022244,898187.200266591 6850249.85785017))
244	648	Kanonenweg	349	358	MULTILINESTRING((898187.200266591 6850249.85785017,898207.393622221 6850316.06541813))

Abb. 2: Datenstruktur der Routingdaten

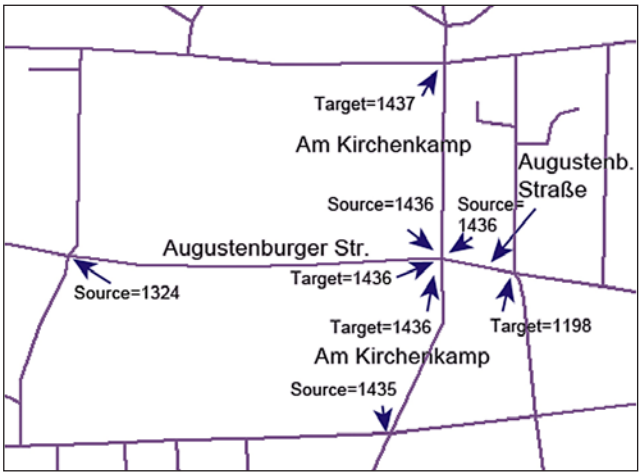

Abb. 3: Source- und Target-Ids innerhalb vorhandener Routingdaten

Abb. 4 zeigt eine Beispielgeometrie mit den für die Routingausgabe in OsnaGo relevanten Attributen.

Der Buchstabe »S« stellt den per Maus angeklickten Routing-Startpunkt in der Anwendung dar. »Z« visualisiert den angeklickten Zielpunkt. Die entsprechende Route wird in Rot dargestellt. Die grünen Dreiecke stehen für die Start- und Endpunkte der Kanten und sind mit der jeweiligen Source- bzw. Target-Id beschriftet (in grüner Schrift). Die grauen Zahlen stellen die Werte der Kanten-Id bzw. deren fest definierte Anfangs-(Source) und End-Id (Target) dar. In Abb. 4 steht somit die Angabe 27188 für die Kanten-Id, der Wert 13030 für die Source-Id

Abb. 4: Routingrelevante Attribute für »pgRouting«

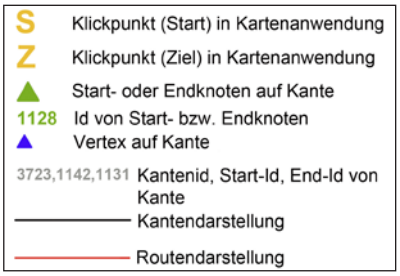

Abb. 5: Legende

der Kante und die 6641 für die Target-Id 102 (siehe dazu auch Abb. 5).

Die Routingberechnung innerhalb der Anwendung www.osnago-karte.de funktioniert nach folgendem Verfahren: Ein Nutzer setzt per Mausklick einen Start- bzw. Zielpunkt. Mittels AJAX wird eine PHP-Datei aufgerufen, welche die Koordinaten des Start- und Zielpunktes als übergebene Parameter erhält.

Über eine Datenbankabfrage wird dann errechnet, welche Kante am nächsten am gesetzten Start- bzw. Zielpunkt liegt. Von diesen zwei ermittelten Kanten wird dann für pgRouting zunächst die Source-Id der Startkante und die Target-Id der Zielkante ausgelesen. Über eine in pgRouting bereits existente Funktion (dijkstra_sp_delta) errechnet der Dijkstra-Algorithmus die kürzeste Route zwischen diesen Punkten. Die entsprechenden Geometrien werden in der Anwendung aus der PostgreSQL/PostGIS-Datenbank im WKT-Format ausgelesen und innerhalb einer XML-Vorlage ausgegeben. Anschließend werden diese Geometrien aus der AJAX-XML-Antwort gefiltert und innerhalb des OpenLayers-Frontend im Kartenklienten MapFish gezeichnet.

2.1 Zwischenstücke zwischen Klickpunkten und Anfangs- und Endknoten der Kanten

Standardmäßig wird in pgRouting immer die Strecke vom Source-Wert der Startkante bis zum Target-Wert der Zielkante errechnet. Dies kann zu einer falschen Routingdarstellung führen, da, anders als in Abb. 4, gesetzte Start- bzw. Endpunkte nahezu nie direkt auf einem Start- oder Endknoten bzw. der entsprechenden Kante selber liegen. In Abb. 6 besitzt die dem Routing-Startpunkt nächstgelegene Kante die Kanten-Id 3723 und die Source-Id 1142. Die dem Routing-Endpunkt nächstgelegene Kante trägt die Id 1999 und besitzt als Target-Id den Wert 1129. Somit

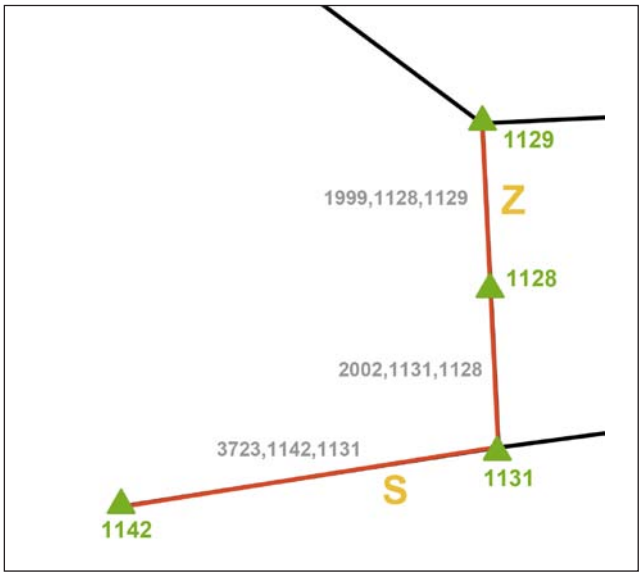

Abb. 6: Ausgabe von »zu viel« Routingstrecke

wird sowohl auf der Startkante als auch auf der Zielkante »zu viel« Strecke ausgegeben. Denkbar ist natürlich auch der »umgekehrte« Weg (die Ausgabe einer »zu kurzen« Strecke).

Um solche Fehler zu vermeiden, wurde nachfolgend beschriebene Lösung entwickelt:

Es ist zunächst zwingend notwendig, die genaue Geometrie zwischen gesetztem Routing-Startpunkt und dem nächsten zu passierenden Vertex (dem geometrischen Zwischenpunkt, siehe dazu auch Abb. 7) zu ermitteln. Dies gilt auch für den Routing-Zielpunkt und den zuvor zu passierenden Vertex.

Über eine in PL/pgSQL geschriebene Funktion wird zunächst ermittelt, welches der nächstgelegene Punkt vom gesetzten Routing-Startpunkt (Startbutton) auf der nächstgelegenen Kante ist. Zudem wird die Id der Kante ermittelt, welche die Startkante berührt (»Nebenkante« in Abb. 7).

Wurde nun ermittelt, welche Koordinaten (x-Wert, y-Wert) auf der Startkante am nächsten am Routing-Startbutton gelegen sind (P1 in Abb. 7) und welche Koordinaten der Berührungspunkt der Startkante mit der »Nebenkante« besitzt (P2), dann muss »nur noch« die Geometrie zwischen diesen beiden Punkten erfasst werden. Um diese Aufgabe zu lösen, wurde eine Funktion in PL/pgSQL entwickelt.

Die Funktion erhält als übergebene Werte die Id's der Start- und Nebenkante sowie die Koordinaten des Punktes auf der Startkante, welcher dem Klickpunkt am nächsten liegt. Darüber hinaus wird analysiert, an welcher Koordinate sich Start- und Nebenkante berühren. In einer Schleife werden zudem die Geometrien der Teilstücke zwischen den einzelnen Vertices der Startkante ermittelt.

Für jedes Teilstück wird untersucht, ob es die Koordinate des dem Klickpunkt nächstgelegenen Punktes auf der Startkante schneidet. Trifft dies an einer Stelle zu, so wird im WKT-Format eine Geometrie (Multilinestring) temporär erstellt, die aus den Koordinaten des dem Klickpunkt nächstgelegenen Punktes auf der Startkante und dem nächsten ermittelten Vertex besteht. Die Schleife muss also optional in zwei Richtungen durchlaufen werden können.

Diese Funktion wird analog auch für die Ermittlung der Routing-Geometrie auf der Zielkante verwendet. Das Ergebnis ist letztlich eine korrekte Routingvisualisierung der Teilstücke auf Start- und Zielkante.

2.2 Ausgabe der korrekten Routinglänge

Trotz dieser geschilderten Erweiterung liefert pgRouting an dieser Stelle in der WebMapping-Anwendung teilweise noch ein falsches Ergebnis (siehe Abb. 8).

Es ist zu erkennen, dass der kürzeste Weg eigentlich über die Kante mit der Id 9757 und der Source-Id 4981 sowie der Target-Id 1777 führen müsste. Der Grund für die »falsche« Ausgabe der Strecke liegt darin, dass die Startkante als Source-Id den Wert 4980 aufweist und die Target-Id der Zielkante den Wert 1778 besitzt. Außerdem kommt hinzu, dass an dieser Stelle noch keine Berücksichtigung der Streckenlänge der Teilgeometrien auf Start- und Zielkante erfolgt. Die Zwischenstücke auf Start- und Zielkante werden in die Berechnung noch nicht integriert. Die Funktion errechnet somit, dass die Strecke von der Source-Id 4980 auf der Kante 9739 zur Target-Id 1778 der Kante 3171 am kürzesten über die Kante 9761 führt.

Letztlich wird zur Lösung des Problems nun ein Weg über eine Vierfachberechnung angewandt:

Variante 1: Source-Id auf Startkante bis hin zur Target-Id auf Zielkante

Variante 2: Source-Id auf Startkante bis hin zur Source-Id auf Zielkante

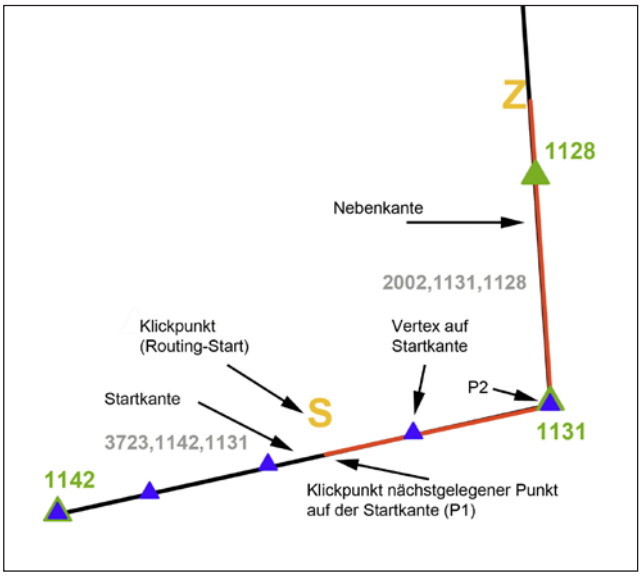

Abb. 7: Darstellung der Routing-Startkante sowie der »Nebenkante«

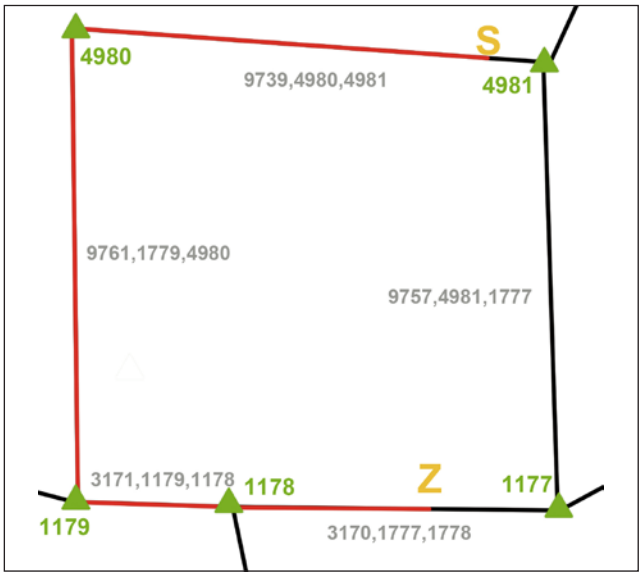

Abb. 8: Falsche Darstellung der Routingstrecke

Variante 3: Target-Id auf Startkante bis hin zur Source-Id auf Zielkante
Variante 4: Target-Id auf Startkante bis hin zur Target-Id auf Zielkante

Die Längen der Strecken werden errechnet und die kürzeste Strecke gezeichnet. Im Rahmen der Dissertation wurde dafür eine Lösung entwickelt, sodass eine korrekte Routingausgabe stattfindet.

Die Lösung wird technisch mit Hilfe von PHP und SQL umgesetzt.

Zwar mag das oben beschriebene Verfahren auf den ersten Blick als etwas umständlich erscheinen, dennoch ist es für eine WebMapping-Anwendung einer einzelnen Stadt für eine Praxisanwendung akzeptabel. Die Berechnung und Darstellung der Route dauert für eine Routingausgabe vom Nordosten des Osnabrücker Stadtgebietes bis zum Südwesten der Stadt unter Nutzung eines Servers (Strato-High-Q-Server Mr-7 (Version 9.6), AMD Opteron. 1385-Prozessor, CPU mit vier mal 2,7 GHZ, vier Gigabyte Ram) nur etwa 1,1 Sekunden (inklusive Berechnung der Abbiegevorschrift, textueller Ausgabe und Erstellung einer GPX-Datei mit integrierter Routingausgabe).

2.3 Ausgabe der Routingbeschreibung

Innerhalb der WebMapping-Anwendung wird eine Routingbeschreibung inklusive Angabe der Abbiegerichtung ausgegeben. Die Berechnung der Routingausgabe erfolgt über den Azimuth in Bogenmaß und wird anschließend in Grad umgerechnet. Die pgRouting-Bibliothek liefert die Kanten der Routingstrecke zwar nach Berechnung der Route in einer korrekten Reihenfolge, allerdings nicht automatisch in korrekter Richtung. Dies ist aber für eine korrekte Berechnung der Abbiegerichtung zwingend notwendig.

Abb. 9 verdeutlicht das Problem. Um eine Route zu visualisieren, ist es zwar völlig unerheblich, in welche Richtung einzelne Kanten gezeichnet werden, für die Azimuthberechnung ist dies jedoch von Bedeutung. Theoretisch könnte in dem obigen Beispiel auch die Berechnung des Azimuths von Anfangs- und End-Vertex des Teilstücks A3 (Kante A) mit dem Schnittpunkt von Kante A/B und dem dann vorletzten Vertex (Koordinate 896393, 6850249) der Kante B geschehen, jedoch ist es einfacher, zunächst alle Geometrien in die richtige Richtung zu bringen und dann eine Winkelberechnung durchzuführen. Hierfür wird geprüft, ob der erste Punkt auf Teilstück 2 (Kante B) identisch ist mit dem letzten Punkt auf Teilstück 1 (Kante A). Wenn dies nicht der Fall ist, dann wird Kante B mit der PostGIS-Funktion »st_reverse()« gedreht (siehe Abb. 10).

Dieses Procedere erfolgt in Anlehnung an eine im Rahmen eines Studienprojektes am Institut für Geoinformatik und Fernerkundung (IGF) entwickelte Lösung (vgl. Cakmak et al. 2010; Papenfuß et al. 2010). Die technische Umsetzung innerhalb der Dissertation stellt jedoch eine Eigenentwicklung dar.

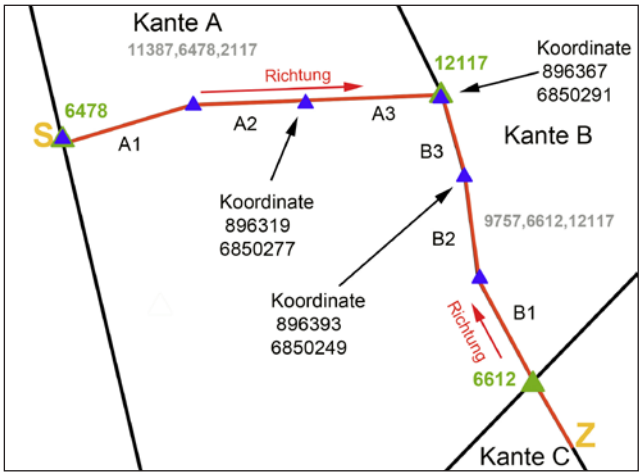

Abb. 9: Falsche Richtung der Kanten

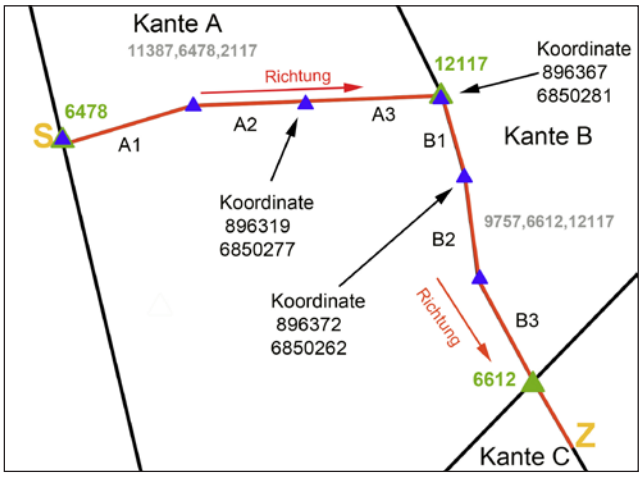

Abb. 10: Korrekte Richtung der Kanten

Die »unsortierten« Geometrien werden also sortiert. Der Anfangspunkt einer Folgekante entspricht nun dem Endpunkt der vorhergehenden Kante. Für jede Kante wird anschließend in einer Schleife ermittelt, aus wie vielen Vertices die Kante besteht. Dann werden die Koordinaten des vorletzten sowie des letzten Vertices einer Kante sowie der Azimuth dieser beiden Vertices ermittelt. Dies geschieht mit der PostGIS-Funktion ST_Azimuth(geometry1, geometry2). Das Ergebnis wird in Bogenmaß zurückgegeben und mittels der Formel »Bogenmaß/2 Pi + 360« in Grad transformiert.

Anschließend wird auf der nächstfolgenden Kante die Koordinate des ersten Vertices (somit des Schnittpunktes zwischen Kante 1 und Kante 2) sowie des nächstfolgenden Vertices ermittelt. Mit den beiden Azimuthwerten kann dann am Schnittpunkt der Kanten die Richtung bestimmt werden. Für die entsprechende Ermittlung wird der Winkel der zweiten Kante subtrahiert mit dem Winkel der ersten Kante. Wenn die Subtraktion zu einem negativen Wert führt, so wird dieser zu 360 addiert, da negative Abbiegewinkel nicht logisch sind.

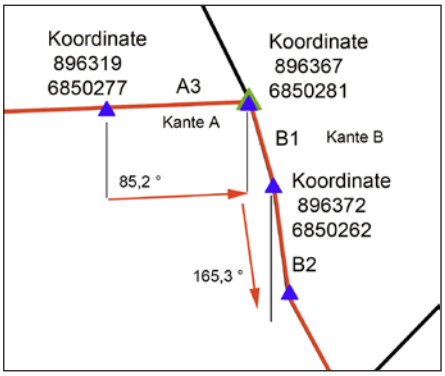

Abb. 11: Abbiegevorschrift »rechts«

In dem Beispiel entsteht dabei ein Wert von 80,1° (Abb. 11). Auf Grundlage dieses Wertes kann dann angegeben werden, in welche Richtung eine Person abbiegen soll. Abb. 12 verdeutlicht, welche Winkelspannen welchen Abbiegerichtungen zuzuordnen sind. Bei einem Winkel von 80,1° lautet die Abbiegevorschrift somit »rechts«.

Zu beachten ist, dass eine Abbiegeanweisung immer erst dann erfolgen darf, wenn eine Kante verlassen wird. Es muss hierbei berücksichtigt werden, dass einzelne Straßen oftmals aus mehreren Kanten bestehen, die wiederum verschiedene Multiline-Teilgeometrien aufweisen können.

Durch eine JavaScript-Funktion wird innerhalb einer Schleife ermittelt, bei welcher Kante eine Namensänderung (im Vergleich zur Vorgängerkante) stattfindet (zudem werden bei gleichnamigen Kanten entsprechende Distanzen addiert). An dieser Stelle erfolgt dann die Ausgabe der Abbiegeanweisung.

In der Anwendung selber erfolgt eine Darstellung beispielsweise der Art wie in Abb. 13.

Das Flussdiagramm in Abb. 14 fasst die routingrelevanten Verfahrensschritte der Anwendung www.osnagokarte.de zusammen.

3 Fazit

Routingfunktionen genießen in verschiedenen WebMapping-Anwendungen eine hohe Bedeutung. Innerhalb der erwähnten Dissertation ist es gelungen, mittels AJAX, PHP, PL/pgSQL unter Nutzung der Datenbank PostgreSQL/PostGIS eine Anwendung zu erstellen, welche eine performante und korrekte Routingberechnung basierend auf pgRouting und OpenStreetMap-Daten gewährleistet, und insbesondere völlig unabhängig von externen Services, Diensten oder API arbeitet. Die Erstellung der Routen funktioniert ausschließlich mit Open Source Software und freien Geodaten. Entsprechender Quellcode steht interessierten Anwendern jederzeit zur Verfügung. In diesem Artikel wurde eine entsprechende Umsetzung sowohl auf Daten- als auch auf Software-(Skript)-Ebene ausführlich beschrieben. Die dargestellte Lösung ermöglicht eine einfache Anpassung an individuelle Bedürfnisse (z. B. Erstellung besonderer Routingprofile für Fußgänger, Radfahrer etc.).

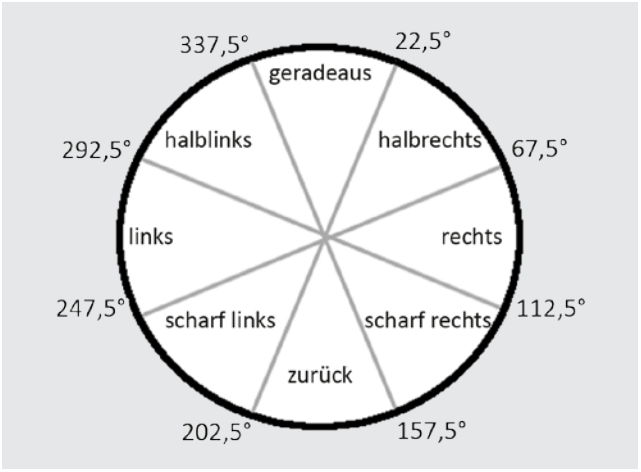

Abb. 12: Winkelangaben für das Routing

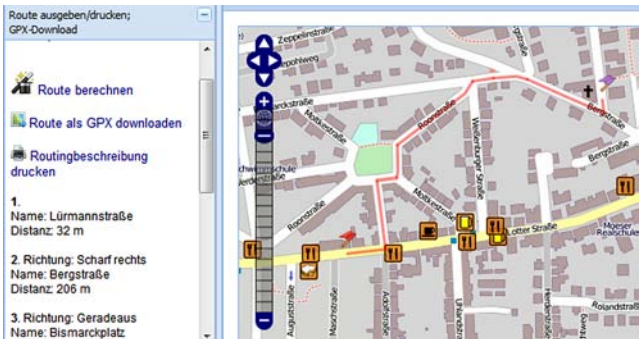

Abb. 13: Routingausgabe in OsnaGo

Literatur

Amelunxen, C.: An Approach to Geocoding based on Volunteered Spatial Data. In: Zipf, A.; Behncke, K.; Hillen, F.; Schaefermeyer, J. (Hrsg.): Geoinformatik 2010. Die Welt im Netz. Konferenzband 17.–19. März 2010, Kiel. Heidelberg, AKA. S. 7–12, 2010.

Behncke, K.: Gastronomap: Gastronomieführer mit Open Source Software und freien Geodaten. GIS Science, 23(1). S. 22–25, 2010a.

Behncke, K.: Gastronomieförderung durch OSM-Daten, WebMapping, mobile Navigation & City2Click-Codes. In: Tagungsband FOSSGIS – Anwenderkonferenz für Freie und Open Source Software für Geoinformationssysteme, 2.–5. März 2010, Osnabrück, S. 61–68, 2010b.

Behncke, K.: Fallstudie zur Nutzungsbedeutung von WebMapping-Anwendungen innerhalb eines Webportals. Eine empirische Untersuchung eines Open Source-Gastronomieportals für die Stadt Osnabrück. Dissertation am Fachbereich Mathematik/Informatik der Universität Osnabrück, 2011.

Behncke, K.; Brinkhoff, T.; Ehlers, M.: WebMapping-Inhalte in einem Webportal: Nur Spielerei oder wichtige Informationskomponente? Eine empirische Untersuchung. Kartographische Nachrichten, 62(1), S. 9–16, 2012.

Cakmak, M.; Giebel, N.; Gräuler, S.; Hempen, B.; Hurink, H.; Hütten, L.; Lüttmann, S.; Mescheder, A.; Schiller, I.; Schoof, M.; Ströer, S.; Thielscher, M.; Tischer, C.: Entwicklung eines Radroutenplaners unter Verwendung von freien Geodaten und Open-Source-Software. Projektbericht Studienprojekt 2009/2010. Institut für Geoinformatik und Fernerkundung, Universität Osnabrück, 2010.

Franke, D.; Dzafic, S.; Weise, C.; Kowalewski, S.: Konzept eines mobilen OSM-Navigationssystems für Elektrofahrzeuge. In: Strobl, J.; Blaschke, T.; Griesebner G. (Hrsg.): Angewandte Geoinformatik 2011. Beiträge zum 23. AGIT-Symposium Salzburg. Heidelberg, Wichmann, S. 148–157, 2011.

Heiken, A.; Peyke, G.: Zur Interoperabilität und Benutzerfreundlichkeit von OSM-Daten. In: Strobl, J.; Blaschke, T.; Griesebner G. (Hrsg.): Angewandte Geoinformatik 2011. Beiträge zum 23. AGIT-Symposium Salzburg. Heidelberg, Wichmann, S. 158–163, 2011.

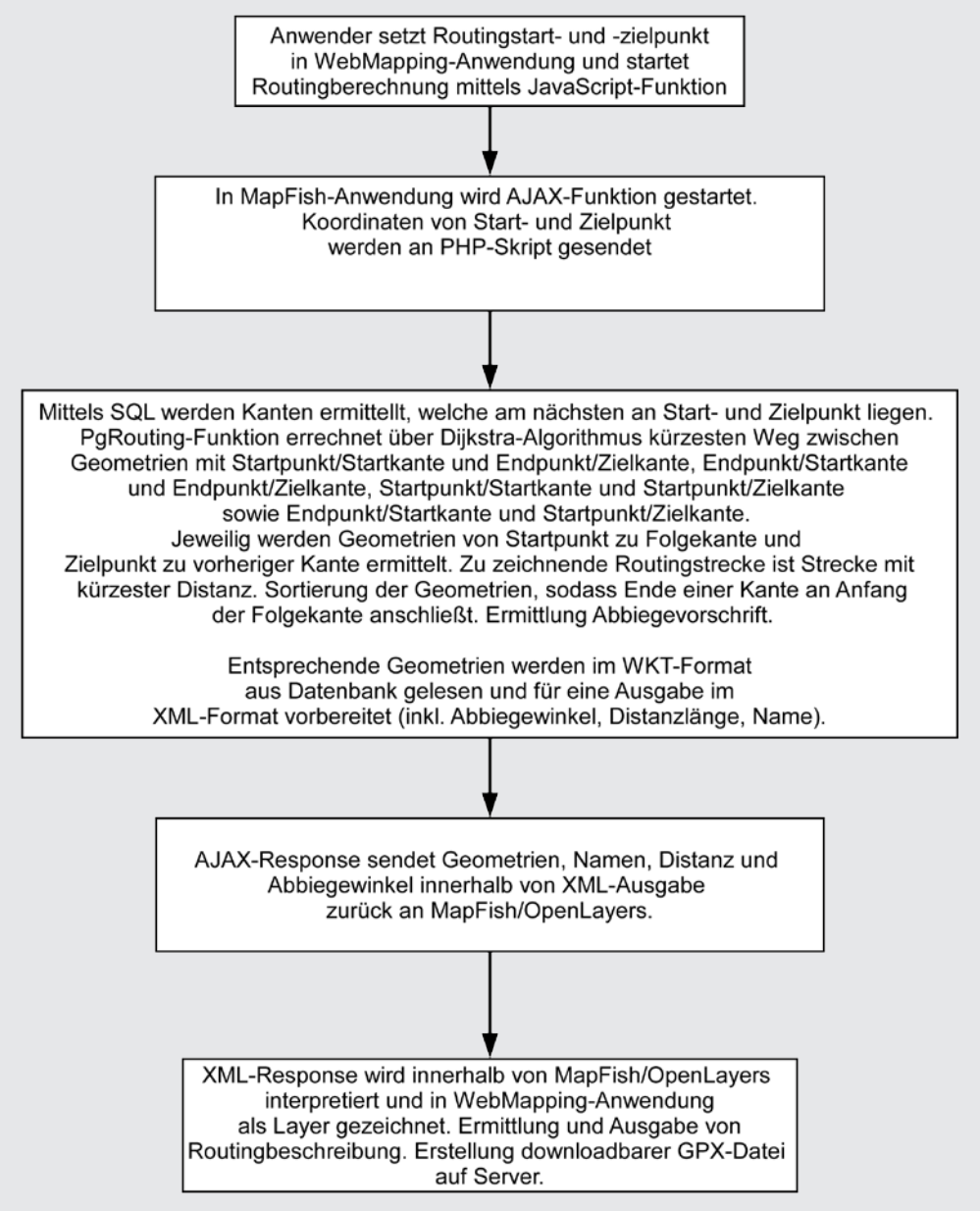

Abb. 14: Routingverfahren innerhalb von OsnaGo

Ludwig, I.; Voss, A.; Krause-Traudes, M.: Wie gut ist OpenStreetMap? Zur Methodik eines automatisierten objektbasierten Vergleiches der Strassennetze von OSM und NAVTEQ in Deutschland. GIS Science, 23(4). S. 148–158, 2010.

Müller, A.; Neis, P.; Auer, M.; Zipf, A.: Ein Routenplaner für Rollstuhlfahrer auf der Basis von OpenStreetMap-Daten – Konzeption, Realisierung und Perspektiven. In: Strobl, J.; Blaschke, T.; Griesebner, G. (Hrsg.): Angewandte Geoinformatik 2010. Beiträge zum 22. AGIT-Symposium Salzburg. Heidelberg, Wichmann, S. 258–261, 2010.

Nivala, A.-M.; Brewster, S.; Sarjakowski, L.T.: Usability Evaluation of Web Mapping Sites. The Cartographic Journal, 45(2). S. 129–138, 2008.

Papenfuß, K.; Westerholt, R.; Zimmer, B.: Konzeption eines Radroutenplaners unter Verwendung von Open-Source-Software. Projektbericht Studienprojekt 2009/2010. Institut für Geoinformatik und Fernerkundung, Universität Osnabrück, 2010.

Schoof, M.; Behncke, K.; Ehlers, M.: ATKIS-Basis-DLM und OpenStreetMap – Ein Datenvergleich anhand ausgewählter Gebiete in Niedersachsen. In: FOSSGIS e.V. (Hrsg.): Tagungsband FOSSGIS – Anwenderkonferenz für Freie und Open Source Software für Geoinformationssysteme. Heidelberg 5.–7. April 2011. S. 118–125, 2011.

Stengel, S.; Pomplun, S.: Die freie Weltkarte OpenStreetMap – Potenziale und Risiken. Kartographische Nachrichten, 61(3), S. 115–120, 2011.

Wachowicz, M.; Cui, L.; Vullings, W.; Bulens, J.: The effects of web mapping applications on user satisfaction: an empirical study. In: Peterson, M.P. (Hrsg.): International Perspectives on Maps and the Internet. Berlin et al., Springer. S. 397–415, 2008.

Zielstra, D.; Zipf, A.: A Comparative Study of Proprietary Geodata and Volunteered Geographic Information for Germany. In: AGILE 2010. The 13th AGILE International Conference of Geographic Information Science. Guimares, 2010.

Anschrift der Verfasser
Dr. Kai Behncke
behncke@creativista-solutions.de

Prof. Dr. Manfred Ehlers
Institut für Geoinformatik und Fernerkundung, Universität Osnabrück
Barbarastraße 22, 49076 Osnabrück
mehlers@igf.uni-osnabrueck.de