

Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Diplomarbeit Nr. 3702

Anpassung der Routenberechnung zur Stressreduktion beim Autofahren

Tobias Weißer

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. Albrecht Schmidt
Betreuer/in:	Dipl.-Inf. Bastian Pfleging, Pascal Knierim, M.Sc.
Beginn am:	28. November 2014
Beendet am:	6. Februar 2016
CR-Nummer:	H.5.2

Kurzfassung

Navigation hat in den vergangenen Jahren an Bedeutung zugenommen und ist durch die Verbreitung von Smartphones zu einer alltäglichen Sache geworden. Meistens werden die berechneten Routen dabei auf einen bestimmten Parameter hin optimiert, z. B. die Fahrzeit, die Distanz oder den Verbrauch von Treibstoff. Ein noch sehr wenig beachteter Faktor ist Stress, welchen eine Route beim Fahrer auslösen kann. Stress kann durch unterschiedliche Faktoren beeinflusst werden, beispielsweise das Verkehrsaufkommen, die Fahrbahneigenschaften oder Zeitdruck. In dieser Arbeit wurde eine neue Routenoption untersucht, welche das Ziel hat, den Fahrer möglichst stressfrei und entspannt an sein Ziel zu bringen. Hierfür wurden verwandte Arbeiten zu Routenberechnung, Stress und Erlebniserfassung untersucht. Basierend auf digitalen Vektorkarten von OpenStreetMap (OSM) wurden, mithilfe der Navigation-Application (App) OsmAnd für Android Smartphones, Bewertungen der Benutzer und Sensordaten der Geräte zu einzelnen Streckensegmenten gesammelt und ausgewertet. Die Navigation-App wurde für diese Arbeit um ein Plugin erweitert, über welches der Benutzer die Bewertung der Streckensegmente vornehmen kann. Es wurde eine eigene Version von OsmAnd namens Happy Navi (OsmAnd) erstellt und über den Google Play Store sowie die eigene Homepage zur Verfügung gestellt. Die ausgewerteten Bewertungen werden von der App in zukünftige Routenberechnungen miteinbezogen und zeigen dem Fahrer eine alternative, stressreduzierte Route als zweite Option neben der normalen Route an. Mit den sensorisch erfassten Daten wurde versucht Aussagen darüber zu treffen, in welchen Situationen der Fahrer gestresst ist und welche Faktoren dabei eine Rolle spielen. Hierbei konnte gezeigt werden, dass es Korrelationen zwischen den Beschleunigungs- und Geschwindigkeitswerten im Bezug auf die Bewertung der Streckensegmente geben könnte. Die gewonnenen Erkenntnisse können in Zukunft dafür genutzt werden, den Stress einer Route möglicherweise automatisch anhand der sensorischen Daten zu bestimmen.

Abstract

Navigation has gained importance in recent years and has become a ubiquitous thing through the spread of smart phones. Most of the time, calculated routes are thereby optimized for a specific parameter, like travel time, distance or consumption of fuel. Stress, a driver might be exposed to on a route, is yet a factor with very little attention. Stress can be influenced by different factors, such as the volume of traffic, road characteristics or through time pressure. In this work, a new route option is examined, which aims to bring drivers relaxed to their destination and with as little stress as possible. For this, related work to route calculation, stress and experience sampling has been investigated. Based on digital vector maps of OSM, using the navigation App OsmAnd for Android smart phones, opinions of users and sensor data of the devices about individual route segments were collected and then evaluated. The navigation app has been extended for this work with a plugin through which the user can make the evaluation of route segments. An adapted version of OsmAnd called Happy Navi (OsmAnd) was created and made available via the Google Play Store and home page. The evaluated ratings are then included in future route calculations by the app and show drivers an alternative, stress reduced route as a second option in addition to the normal route. Using recorded sensor data the goal was to identify situations in which the driver is stressed and what factors play a role therefor. It could be shown that there might be correlations between the acceleration and velocity values with respect to the evaluation of the route segments. These findings could be useful in the future for determining the stress of a route automatically based on sensory data.

Inhaltsverzeichnis

1	Einleitung	13
2	Verwandte Arbeiten	15
2.1	Algorithmen zur Routenberechnung	16
2.1.1	Dijkstra-Algorithmus	16
2.1.2	A*-Algorithmus	17
2.1.3	Ameisenkolonie-Optimierung	18
2.1.4	Unidirektional und bidirektional	20
2.2	Alternative Routenberechnung	20
2.3	Erlebniserfassung im Auto	22
2.4	Stress und kognitive Last beim Autofahren	23
2.5	Ergebnisse der Literaturrecherche	23
3	Grundlagen	25
3.1	Digitale Karten	25
3.1.1	Entstehung digitaler Karten	25
3.1.2	Raster- und Vektorkarten	26
3.1.3	OpenStreetMap	27
3.1.4	Datenstruktur	27
3.2	Android	29
3.2.1	Android Versionen	30
3.2.2	Kompatibilität	32
3.2.3	Architektur	33
3.2.4	Software Development Kit	33
3.2.5	Native Development Kit	34
3.2.6	Manifest	35
3.2.7	Signierung	35
3.2.8	Android Studio	35
4	Konzept	37
4.1	Erfassung von relevanten Daten	37
4.2	Verwendung und Nutzung der Daten	38
5	Implementierung	41
5.1	Anforderungen	41

5.2	OsmAnd	41
5.2.1	Funktionen	42
5.2.2	Plugins	42
5.2.3	Kompilieren von OsmAnd	43
5.3	Happy Navi (OsmAnd)	44
5.4	Stressreduzierung Plugin	44
5.4.1	Aufbau	44
5.4.2	Datenstruktur	48
5.4.3	Einstellungen	48
5.5	Erfassung von Stresswerten	49
5.6	Native Core	49
5.7	Server	50
5.8	Ausführung einer beispielhaften Routenplanung	50
5.9	Versionen	51
6	Ergebnisse	53
6.1	Nutzung von Happy Navi (OsmAnd)	53
6.2	Datenanalyse	54
6.2.1	Ortsdaten	54
6.2.2	Straßenabschnitte	54
6.3	Routenplanung und Routenvergleiche	56
7	Diskussion	59
7.1	Datenerfassung	59
7.2	Korrelation zwischen Stresswerten und Daten	59
7.3	Automatische Stresswert-Bestimmung	62
7.4	Stressreduzierte Routen	63
8	Zusammenfassung und Ausblick	65
8.1	Zusammenfassung	65
8.2	Weiterführende Arbeiten und Themen	66
	Literaturverzeichnis	69

Abbildungsverzeichnis

2.1	Vergleich des schematischen Suchraums zwischen Dijkstra und A*	18
3.1	Unterschied zwischen Vektor- und Rasterkarte	28
3.2	Marktanteilentwicklung verschiedener Betriebssysteme für mobile Endgeräte	30
3.3	Aufbau von Android	34
4.1	Beispielhafter Systemaufbau zur Datenerfassung	39
4.2	Beispiel für verschiedene Routen mit ähnlichen Eigenschaften	40
5.1	Achsen eines Android Smartphones	47
5.2	Dialog zur Bewertung von Straßensegmenten	49
5.3	Schema der Datenbank	51
5.4	Beispiel einer normalen und stressreduzierten Route	52
6.1	Anzahl Nutzungen von Happy Navi (OsmAnd) pro Tag	53
6.2	Übersicht der gesammelten Ortsdaten und deren Häufigkeit als Heatmap in Europa	55
6.3	Bewertungen der Streckensegmente im deutschen Raum	57
7.1	Beschleunigungswerte der Fahrt 1	60
7.2	Geschwindigkeiten der Fahrt 1	61
7.3	Beschleunigungswerte und Bewertungen der Fahrt 2	62
7.4	Geschwindigkeiten und Bewertungen der Fahrt 2	63

Tabellenverzeichnis

5.1	Liste der verfügbaren Plugins in OsmAnd	43
-----	---	----

Verzeichnis der Listings

3.1	Beispiel für den Aufruf einer nativen Methode aus Java heraus	35
5.1	Beispiel für die Normalisierung von Beschleunigungssensordaten	46

Verzeichnis der Algorithmen

2.1	Dijkstra-Algorithmus	17
2.2	A*-Algorithmus	19

Abkürzungsverzeichnis

ADB	Android Debug Bridge (S. 33)
API	Application Programming Interface (S. 33)
App	Application (S. 3)
CAD	Computer Aided Design (S. 25)
CAN	Controller Area Network (S. 38)
DDMS	Dalvik Debug Monitor Server (S. 33)
ESM	Experience Sampling Methode (S. 23)
FCD	Floating Car Data (S. 21)
GPS	Global Positioning System (S. 15)
NDK	Native Development Kit (S. 34)
OBD	On-Board-Diagnose (S. 38)
OHA	Open Handset Alliance (S. 29)
OSM	OpenStreetMap (S. 3)
POI	Point of Interest (S. 15)
SDK	Software Development Kit (S. 33)
TMC	Traffic Message Channel (S. 21)
USB	Universal Serial Bus (S. 33)
XML	Extensible Markup Language (S. 26)

1 Einleitung

Navigationsgeräte sind in Deutschland weit verbreitet. Knapp 50% aller Haushalte besitzen ein Navigationsgerät¹. Zudem verfügt fast jeder zweite Deutsche über ein Smartphone², mit welchem auch navigiert werden kann. Hinzu kommen noch die in Autos verbauten Navigationssysteme. Alle diese Navigationsgeräte bringen den Benutzer in der Regel auf dem schnellsten oder kürzesten Weg an sein Ziel, egal wo er sich gerade befindet und an welches Ziel er kommen möchte.

Das Navigationsgerät wählt die Route hierbei in der Regel statisch nach voreingestellten Parametern aus. Die einzige dynamische Komponente stellt in der Regel das Verkehrsaufkommen dar, vorausgesetzt diese Informationen können vom Gerät abgerufen und verarbeitet werden. Häufig kann es jedoch vorkommen, dass sich Ortskundige Fahrer in bekannten Gegenden einen anderen Streckenabschnitt, als den vom Navigationsgerät vorgeschlagenen, aussuchen und befahren. Die Gründe hierfür können stressige, anstrengende, unübersichtliche oder auch schlecht ausgebaute Streckenabschnitte sein, welche durch eine alternative Route gut substituiert werden können und sich dabei in Länge und Zeit oft nur sehr wenig unterscheiden. Persönliche Einschätzungen und Wissen zu den Streckenabschnitten werden in bisherigen Navigationssystemen aber noch nicht in die Berechnung der Route miteinbezogen. Dabei sind vor allem Ortskundige Fahrer eine gute Informationsquelle dafür, welche Streckenabschnitte gut oder schlecht zu befahren sind. Dieses Wissen zu Straßeneigenschaften in bekannten Gegenden wird oft angewendet, ohne sich dessen bewusst zu sein. So kann es vorkommen, dass einem Bekannten der Weg zu sich selbst nach Hause anders beschrieben wird als der Weg, welchen das Navigationsgerät vorschlagen würde. Hierbei spielen die Ortskenntnisse von Personen, beispielsweise zu engen, kurvigen oder verkehrsreichen Straßen, ungünstig getakteten Lichtsignalanlagen oder temporäre Baustellen, eine große Rolle.

Der Stress, den eine Route beim Fahrer auslösen kann, ist ein bisher noch sehr wenig beachteter Faktor. Viele Streckenabschnitte einer Route können stressig und anstrengend zu befahren sein, was sich negativ auf die Fahrerbelastung auswirken kann. Die Gründe dafür können vielfältig sein und von verschiedenen Faktoren abhängen. Dichtes Verkehrsaufkommen ist einer der größten Stressfaktoren im Straßenverkehr [HW99]. Gerade in lang andauernden

¹Statistisches Bundesamt, Wiesbaden https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/AusstattungGebrauchsgueter/Tabellen/Infotechnik_D.html

²Statista GmbH, Hamburg <http://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonennutzer-in-deutschland-seit-2010/>

Situationen mit viel Verkehr kann der Fahrer schnell gestresst werden und ermüden [GMG+89]. Auch Tunnel können zu erhöhtem Stress führen, welcher durch die Enge, Dunkelheit und den erhöhten Lärmpegel verursacht wird [MR14].

Ziel der Arbeit

Im Rahmen dieser Arbeit soll eine neue Routenoption untersucht werden, welche den Fahrer möglichst entspannt und stressfrei an sein Ziel bringt. Die Route soll hierbei so gewählt werden, dass der Fahrer möglichst wenig Stress verursachende Streckensegmente befahren muss. Hierfür gilt es in einem ersten Schritt herauszufinden, wo sich gute oder schlechte, beziehungsweise entspannte oder stressige Streckenabschnitte befinden und diese mit einer Bewertung zu versehen. Mithilfe dieser Bewertungen sollen in einem zweiten Schritt zukünftige Fahrten verbessert werden. Hierfür sollen bei der Berechnung der Route die Bewertungen der einzelnen Streckenabschnitte mit in die Kalkulation einfließen und, wenn möglich, die besser bewerteten Abschnitte in die Route mit aufgenommen werden. Die Bewertungen für die einzelnen Streckenabschnitte sollen mithilfe von Crowdsourcing gesammelt werden. Hierfür ist der Einsatz einer Navigation-App für Android vorgesehen, über welche der Fahrer Bewertungen zu Streckenabschnitten tätigen kann. Diese fließen wiederum in zukünftige Berechnungen mit ein und verbessern so die Routenwahl für zukünftige Fahrten.

Aufbau der Arbeit

Die vorliegende Arbeit ist folgendermaßen aufgebaut: in Kapitel 2 werden verwandte Arbeiten zu den relevanten Themen vorgestellt. Wichtige Grundlagen werden in Kapitel 3 erläutert. Danach wird das Konzept dieser Arbeit in Kapitel 4 erklärt und in Kapitel 5 die Implementierung der Anwendung beschrieben. Die Ergebnisse der Arbeit werden in Kapitel 6 präsentiert, eine Diskussion der Ergebnisse findet in Kapitel 7 statt. Zum Schluss wird eine Zusammenfassung und ein Ausblick in Kapitel 8 gegeben.

2 Verwandte Arbeiten

Die ersten Navigationssysteme wurden zu Beginn der 80er Jahre entwickelt [AGB13]. Mit dem Gyrocomp von Honda, welcher noch komplett ohne das Global Positioning System (GPS) auskam, und dem von Toyota entwickelten NAVICOM, welches lediglich die Richtung in Form eines Pfeils darstellen konnte, wurden erstmals Navigationssysteme in Fahrzeugen verbaut. Mit der offiziellen Verfügbarkeit von GPS Anfang der 90er Jahre stieg die Verwendung von GPS-basierten Systemen rasant an. Jeder Automobilhersteller entwickelte sein eigenes System und versuchte die neue Technologie in sein Fahrzeugkonzept zu integrieren. Häufig wurde die Chance genutzt, um mehrere Multimediasysteme zusammenzufassen und somit die ersten Infotainmentsysteme zu schaffen. Eine große Herausforderung bestand darin, die Systeme so miteinander zu verknüpfen, dass sie auch während der Fahrt bedient werden konnten. Die Benutzerschnittstelle sollte möglichst intuitiv konstruiert sein und der Fahrer damit auch ohne lange Einlernzeit zurechtkommen. Durch die zusätzlichen Aufgaben, neben dem eigentlichen Fahren des Fahrzeugs, wurden auch zum ersten Mal Studien bezüglich der mentalen und kognitiven Last (Workload) durchgeführt, um herauszufinden, inwiefern der Fahrer während der Fahrt durch den Einsatz dieser Systeme abgelenkt wird.

In den letzten Jahren hat die Bedeutung von mobiler Navigation stark zugenommen. Mit den ersten Smartphones wurde es möglich, zu jeder Zeit standortbezogene Daten zu generieren und zu verarbeiten. Zuvor war die Navigation ausschließlich mit genau für diesen Zweck konzipierten mobilen oder fest verbauten Navigationsgeräten möglich. Gleichzeitig wurde die Navigation jedoch auch zunehmend komplexer und herausfordernder. Neben der ohnehin schon anspruchsvollen Aufgabe der optimalen Routenberechnung kommen heutzutage auch noch weitere Kriterien zur Routenauswahl hinzu, wie z. B. das Optimieren der Route bezogen auf einen bestimmten Parameter oder eine bestimmte Eigenschaft, welche die Route besitzen soll. Oft wird auch das Anzeigen von erweiterten Inhalten zusätzlich zur berechneten Route gefordert, wie z. B. das Anzeigen von Points of Interest (POIs). Weiterhin spielt auch das Fahrerlebnis (Driving Experience) eine bedeutende Rolle. Automobilhersteller werben damit für bestimmte Aspekte ihrer Fahrzeuge, beispielsweise das Fahrverhalten oder die Beschleunigung. In [PSMT14] wird beschrieben, dass auch die Wahl der Route einen Einfluss auf das Fahrerlebnis haben kann. Faktoren für das Fahrerlebnis sind hiernach unter anderem das Verkehrsaufkommen, der Straßentyp und die Landschaft, durch welche gefahren wird. Das Fahrerlebnis auf der berühmten Route 66 ist sicherlich nicht mit den überfüllten Straßen einer Großstadt zu vergleichen.

Im Folgenden wird ein Überblick über die wichtigsten Routenberechnungsalgorithmen, sowie deren Abwandlungen zur Optimierung von ein oder mehreren Parametern gegeben. Danach

werden die Themen Erlebniserfassung (Experience Sampling), sowie kognitive Last und Stress beim Autofahren beschrieben und Einblicke in die Anforderungen und Schwierigkeiten bei der Messung dieser Parameter gegeben.

2.1 Algorithmen zur Routenberechnung

Die Berechnung der optimalen Route in kurzer Zeit ist, besonders auf mobilen Geräten mit begrenzten Ressourcen, eine sehr schwierige Aufgabe. In diesem Kapitel werden einige der wichtigsten Algorithmen betrachtet, welche im Zusammenhang mit Routenberechnungen gegenwärtig von Interesse sind. Viele der entwickelten Algorithmen basieren auf dem im folgenden Abschnitt beschriebenen Dijkstra-Algorithmus. Allen Algorithmen gemeinsam ist, dass sie als Grundlage einen einfachen, gewichteten, zusammenhängenden Graphen mit Knoten V und Kanten E besitzen, auf welchem sie den optimalen Pfad von einem Startknoten zu einem Endknoten suchen. Der Graph kann hierbei sowohl gerichtet als auch ungerichtet sein, darf aber keine negativen Kantengewichte besitzen.

2.1.1 Dijkstra-Algorithmus

Der Algorithmus von Dijkstra [Dij59] findet in einem kantengewichteten Graphen, wie ihn beispielsweise ein Straßennetz darstellt, einen optimalen Pfad zu einem gegebenen Start- und Endknoten. Genauer gesagt, findet der Algorithmus den optimalen Pfad zu jedem anderen Knoten vom Startknoten aus. Bei der Routenberechnung ist aber lediglich der Pfad von Startknoten zu Zielknoten von Interesse. In einem ersten Schritt wird die Distanz zum Startknoten für alle Knoten im Graphen auf unendlich gesetzt und die Distanz des Startknotens auf null festgelegt. Danach werden alle Knoten des Graphen in eine Prioritätswarteschlange eingefügt und solange, ausgehend vom Startknoten, abgearbeitet, bis der Zielknoten gefunden wurde. Die Kosten für jeden Knoten setzen sich dabei immer aus den Kosten des Vorgängers und den Kosten für den Weg zwischen Vorgänger und aktuellem Knoten zusammen. Somit sind die Kosten für die gesamte Strecke zwischen Start- und Endknoten am Ende im Endknoten gespeichert. In Algorithmus 2.1 wird die Funktionsweise des Dijkstra-Algorithmus gezeigt. Die Laufzeit des Algorithmus beträgt $O(|E| + |V| \cdot \log |V|)$, wobei nur besuchte Knoten und Kanten miteinbezogen werden müssen [Büh13].

Um die Laufzeit zu verbessern, wurde mit dem A*-Algorithmus eine Erweiterung des Dijkstra-Algorithmus vorgestellt, welcher zusätzlich auf eine Heuristik setzt. Durch eine geeignete Heuristik kann der Suchraum des Graphen stark eingeschränkt und damit die Laufzeit beschleunigt werden (siehe Abbildung 2.1). Im folgenden Abschnitt 2.1.2 wird der A*-Algorithmus genauer betrachtet.

Algorithmus 2.1 Dijkstra-Algorithmus

```

procedure DIJKSTRA(Graph  $G$ , Knoten  $start$ , Knoten  $ziel$ )
  for all Knoten  $v \in V$  in  $G$  do
     $distanz[v] = \infty$ 
  end for
   $distanz[start] = 0$ 
  besuchte Knoten  $S \leftarrow \emptyset$ 
  Prioritätswarteschlange  $Q \leftarrow V$ 

  while  $Q \neq \emptyset$  do
    Knoten  $u \leftarrow \text{minDistanz}(Q, distanz)$ 
     $S \leftarrow S \cup \{u\}$ 
    entferne  $u$  aus  $Q$ 
    if  $u == ziel$  then
      return
    end if
    for all Knoten  $v \in \text{adjacent}[u]$  do
       $d[v] = distanz[u] + |u, v|$ 
      if  $d[v] < distanz[v]$  then
         $distanz[v] = d[v]$ 
      end if
    end for
  end while
end procedure

```

2.1.2 A*-Algorithmus

Der A*-Algorithmus (Algorithmus 2.2) ist eine Erweiterung des bekannten Dijkstra-Algorithmus und wurde erstmals von Hart, Nilsson und Raphael [HNR68] vorgestellt. Er stellt für viele aktuellen Navigationsdienste die Basis der Routenberechnung dar, was vor allem an der effizienten und zielgerichteten Suche liegt, welche durch die Hinzunahme einer Heuristik erreicht wird. Der Faktor der Beschleunigung hängt dabei direkt von der Wahl des Heuristikparameters ab. Wichtig ist hierbei, dass die Heuristik die Kosten nie überschätzen darf. Bei der Berechnung der kürzesten Route wird hierfür in der Regel die minimal Entfernung zwischen dem zu untersuchenden Knoten und dem Endknoten in Form der Luftlinie genommen.

Der Algorithmus operiert mit zwei Listen, einer offenen und einer geschlossenen. In der offenen Liste befinden sich alle Knoten des Graphen, welche bereits gefunden, aber noch nicht abschließend untersucht wurden. Die abgearbeiteten Knoten werden in die geschlossene Liste eingefügt. Außerdem werden die Kosten eines Knotens u als Kostenfunktion durch $f(u) = g(u) + h(u)$ beschrieben. Die Gesamtkosten $f(u)$ setzen sich hierbei zusammen aus

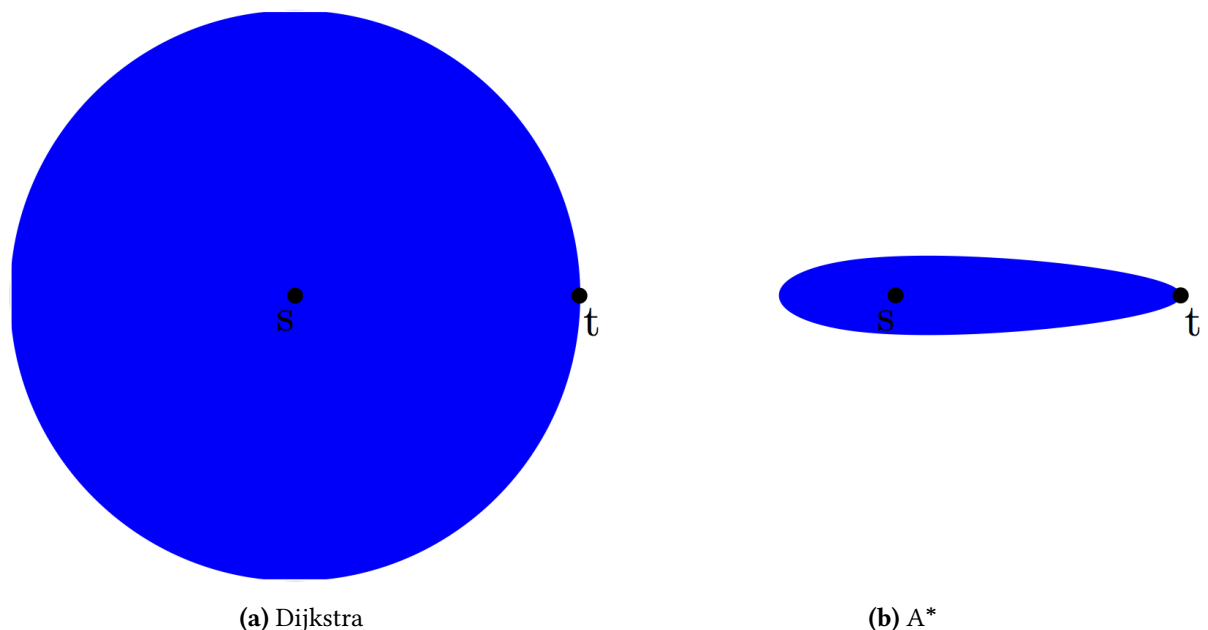


Abbildung 2.1: Vergleich des schematischen Suchraums zwischen Dijkstra und A* Algorithmus (Quelle: Daniel Delling - Algorithmen für Routenplanung, KIT)

den bisherigen minimalen Kosten vom Startknoten zu Knoten u als $g(u)$ plus den geschätzten Kosten vom Knoten u zum Zielknoten als $h(u)$. Zu Beginn wird der Startknoten in die offene Liste eingefügt und von diesem ausgehend immer derjenige Knoten als nächstes untersucht, welcher die geringsten Gesamtkosten verspricht. Dies wird solange fortgeführt, bis der Zielknoten gefunden wurde. Die Funktionsweise wird in Algorithmus 2.2 dargestellt. Inwiefern sich die Laufzeit des A*-Algorithmus gegenüber dem Dijkstra-Algorithmus verbessert, hängt sehr stark von der Genauigkeit der verwendeten Heuristik ab. Je besser die Heuristik die Kosten für den restlichen Weg abschätzt, desto besser wird auch die Laufzeit des Algorithmus. Die Wahl einer bestmöglichen Heuristik ist deshalb ein Kernpunkt für die Optimalität des A*-Algorithmus.

2.1.3 Ameisenkolonie-Optimierung

Eine relativ neue Art von Algorithmen setzt auf die Ameisenkolonie-Optimierung [BM04], welche erstmals von Dorigo, Maniezzo und Colorni [DMC91] als Ameisen System beschrieben wurde. Diese Algorithmen sind Ameisen bei der Futtersuche nachempfunden und nutzen das Prinzip der Schwarmintelligenz. In der Natur setzen Ameisen auf dem Weg vom Nest zur Futterquelle Pheromone entlang des Weges frei. Auf dem kürzesten Weg steigt dabei die Pheromonkonzentration schneller an, als auf längeren Wegen zur Futterquelle. Wege mit höheren Konzentrationen werden von nachfolgenden Ameisen bevorzugt und es bildet sich, auf

Algorithmus 2.2 A*-Algorithmus

```
procedure A*(Graph  $G$ , Knoten  $start$ , Knoten  $ziel$ )
  offene Liste  $OL \leftarrow \emptyset$ 
  geschlossene Liste  $CL \leftarrow \emptyset$ 
   $OL \leftarrow start$ 

  while  $OL \neq \emptyset$  do
    Knoten  $u \leftarrow \text{minKosten}(OL, f(u) = g(u) + h(u))$ 
    entferne  $u$  aus  $OL$ 
    if  $u == ziel$  then
      return
    end if
     $CL \leftarrow CL \cup \{u\}$ 
    for all Knoten  $v \in \text{adjacent}[u]$  do
      if  $v \in CL$  then
        continue
      end if
       $g'(v) \leftarrow g(u) + |u, v|$ 
      if  $v \in OL \wedge g'(v) \geq g(v)$  then
        continue
      end if
       $g(v) \leftarrow g'(v)$ 
       $h(v) \leftarrow |v, ziel|$  //geschätzt
       $f(v) = g(v) + h(v)$ 
      if  $v \in OL$  then
        setze  $f(v)$  für  $v$  in  $OL$ 
      else
         $OL \leftarrow OL \cup \{v\}$ 
      end if
    end for
  end while
end procedure
```

dem kürzesten Weg, eine Ameisenstraße. Im Straßennetz werden die Ameisen durch künstliche Ameisen simuliert, welche auf dem Graphen von Knoten zu Knoten laufen. Dabei wird der genutzten Kante jeweils ein Pheromonwert zugewiesen, welcher von anderen Ameisen gelesen werden kann. Jede Ameise wählt für ihren nächsten Schritt einen Knoten aus, den sie zuvor noch nicht besucht hat. Die Wahrscheinlichkeit für die Auswahl einer Kante wird durch eine stochastische Funktion gepaart mit der Pheromonkonzentration bestimmt. Auf diese Weise bilden sich schon nach kurzer Laufzeit auf den kürzeren Wegen höhere Konzentrationen und das Verfahren konvergiert sehr schnell in Richtung der optimalen Lösung.

2.1.4 Unidirektional und bidirektional

Alle unidirektionalen Algorithmen (darunter Dijkstra und A*) können auch in einer bidirektionalen Variante implementiert werden, wodurch die Laufzeiten noch einmal verringert werden können. Bei der bidirektionalen Suche wird von Start- und Endknoten ausgehend gleichzeitig der jeweilige Gegenpart gesucht. Die Suche erfolgt so lange, bis ein gemeinsamer Knoten gefunden wurde. Als Ergebnis wird dann die zusammengesetzte Strecke der beiden Teilstrecken ausgegeben. Besonders Prozessoren mit mehreren Kernen können hier durch Parallelisierung einen beträchtlichen Geschwindigkeitszuwachs erreichen.

2.2 Alternative Routenberechnung

Die Art und Weise, wie eine Routenberechnung erfolgt, hängt stark davon ab, welcher Parameter optimiert werden soll. So kann bei fast allen Navigationsdiensten zwischen der schnellsten und der kürzesten Route gewählt werden. Einige bieten auch eine dritte Möglichkeit für die am meisten ökonomische und Treibstoff sparende Route an. Neben diesen gängigen Parametern gibt es einige, zum Teil auf spezielle Probleme ausgelegte, Optionen.

Montealegre, Haase und Boufelliga [MHB12] reichten 2012 ein Patent ein, welches Routen anhand ihrer Sportlichkeit auswählt. Die Navigationssoftware analysiert hierfür die Straßendaten zwischen Start- und Endpunkt bezüglich der Wendepunkte eines kurvigen Streckenabschnitts und wählt danach die sportlichste Route aus. Ähnliches gilt auch für die von TomTom speziell für Motorradfahrer konzipierten Navigationsgeräte der Rider-Serie¹. Zusätzlich zur Sportlichkeit einer Route in Form von möglichst vielen kurvigen Straßenelementen können Routen hier auch nach Anzahl der Höhenmeter ausgewählt werden. Eine weitere Besonderheit ist die Berechnung einer Rundreise mit frei wählbaren Gebieten, welche besucht werden sollen. Für diese Strecke kann eine Reisedauer angegeben werden und die Rundreise wird danach mit den festgelegten Parametern ausgewählt.

¹TomTom - Product Lineup http://www.tomtom.com/en_gb/drive/motorcycle/#productLineup

Für Autofahrer gibt es verschiedene Systeme, welche Verkehrsdaten während der Fahrt abrufen und eine Route dynamisch an das Verkehrsaufkommen anpassen können. Die wichtigsten Technologien sind hierbei Traffic Message Channel (TMC), Floating Car Data (FCD) und an Straßen verbaute Verkehrssensoren sowie Kameras. Google Maps und Apple Maps verwenden FCD, indem sie die Positionsdaten von Smartphones nutzen, um die Geschwindigkeit von Fahrzeugen zu bestimmen und Aussagen über Verkehrsbehinderungen treffen zu können. Mit dem Navigationsdienst Waze hat sich Google zudem einen Partner an Bord geholt, um die Verkehrslage noch genauer einschätzen zu können. Bei Waze können Nutzer direkt Angaben zu Verkehrssituationen vornehmen und diese mit anderen Teilnehmern teilen. Viele erhältliche Navigationssysteme können TMC Informationen abrufen und verarbeiten. Allerdings wird TMC nur alle 15 Minuten aktualisiert und es kann passieren, dass man sich schon im Stau befindet bevor dieser vom System gemeldet wird. Einen anderen Ansatz verfolgt TomTom HD Traffic and IQ Routes [Tom10]. Hier werden nicht nur aktuelle Verkehrsinformationen für die Routenführung herangezogen, sondern auch historische Verkehrsaufkommen beachtet, um Straßenabschnitte mit hohem Verkehrsaufkommen zu bestimmten Tageszeiten schon von vornherein zu meiden.

Mit dem vermehrten Aufkommen von rein elektrisch betriebenen Fahrzeugen und ihrer oft noch geringen Reichweite besteht die Notwendigkeit, bei der Routenplanung auch die Lage von Tankstellen für Elektrofahrzeuge sowie deren unterschiedliche Aufladezeiten zu berücksichtigen [YCC15]. Zudem ist die schnellste Route für diese Art von Fahrzeugen aufgrund des erhöhten Stromverbrauchs oft nicht die beste Wahl [BDH+14]. Der hier verfolgte Ansatz besteht darin, die Routenplanung mit Hilfe einer Multikriterien-Optimierung für die Geschwindigkeit sowie den Energieverbrauch zu optimieren. Auch die Umweltbelastung einer Route kann ein Faktor für die Routenwahl sein. Boriboonsomsin et al. [BBZV12] suchen hierfür in ihrem Ansatz die am meisten Treibstoff sparende oder am wenigsten Schadstoff produzierende Route.

In [PSMT14] wird noch ein anderer Aspekt für die Routenbestimmung genannt, welcher die Grundlage für diese Arbeit legt. Das Fahrerlebnis ist ein Faktor, welcher bisher noch sehr wenig Beachtung findet. Anders als bei Optimierungen bezüglich der üblichen Faktoren soll hier die Route so gewählt werden, dass sie möglichst wenig Stress und kognitive Last enthält. Die Route soll stattdessen einfach zu befahren sein, Spaß machen und den Fahrer möglichst wenig belasten. Es wird vorgeschlagen, dies durch den Einsatz zusätzlicher Gewichtungen der Straßensegmente zu erreichen und diese zu den verwendeten Faktoren hinzuzufügen. In einer Umfrage wurde herausgefunden, dass es für viele Menschen ein bedeutender Faktor ist, möglichst stressfrei und auf einfach zu befahrenden Routen an ihr Ziel zu kommen.

2.3 Erlebniserfassung im Auto

Definition 2.3.1 (Experience Sampling)

Experience sampling is a systematic way of having participants provide samples of their ongoing behavior.

(Barbara A. Sommer, University of California, Department of Psychology)²

Mit Hilfe von Erlebniserfassung kann ein besserer Einblick für in situ Vorgänge gewonnen werden. Atz [Atz13] untersuchte hierfür drei verschiedene Methoden genauer. Die erste von ihnen untersuchte Methode ist die Experience Sampling Methode (ESM). Hierbei werden Probanden in regelmäßigen Abständen zu ihrem Befinden befragt, während sie gerade eine vorgesehene Aufgabe ausführen. Die zweite Methode ist die Day Reconstruction Methode (DRM), bei welcher sie Teilnehmer dazu aufforderten am Ende des Tages Notizen zur Aufgabenstellung zu machen. Als dritte Methode wurde von ihnen der Herzschlag pro Minute über einen Brustgürtel gemessen und aufgezeichnet.

Mit dem Trip Experience Sampling (TES) stellten Wilfinger et al. [WMB+13] eine Methode vor, bei der Fahrer nach einer Fahrt ihre Erlebnisse in einem Logbuch festhielten. Durch die kurze Zeitspanne zwischen dem Erlebten und der Befragung sind den Teilnehmern mehr Details in Erinnerung geblieben und die Gefühlslage wird nicht durch spätere Ereignisse beeinflusst. Ihr Ziel war es herauszufinden, was sich im Auto während der Fahrt abspielt und wie sich kontextbezogene Faktoren auf das Befinden während der Fahrt auswirken.

Im Automotive-Kontext ist die Erlebniserfassung mit einigen Schwierigkeiten und Herausforderungen verbunden. Die Sicherheit des Fahrers darf durch den Einsatz von Methoden zur Erlebniserfassung nicht gefährdet werden. Es sollen keine lang andauernden Erlebniserfassungen während der Fahrt durchgeführt werden, welche den Fahrer von der eigentlichen Fahraufgabe abhalten, ihn durch umfangreiche Verkabelung in seiner Bewegungsfreiheit einschränken oder durch Aufbauten sein Sichtfeld einschränken. Nach Meschtscherjakov et al. [MTD+13] besteht die Herausforderung darin, die Erlebnisse und Gefühle von Fahrern in einer unauffälligen und nicht ablenkenden Art und Weise festzuhalten. Ihr Ansatz besteht darin, die Befragung direkt nach erfolgreichem Abschließen der Aufgabe über ein Smartphone durchzuführen. Diese Möglichkeit bietet sich jedoch nur bei Aufgaben mit einer kurzen Zeitspanne an. Die Benutzung von Smartphones wird als sehr hilfreich für diese Art von Studien eingeschätzt. Zum einen kann die Befragung direkt über das Smartphone erfolgen, zum anderen können die im Gerät eingebauten Sensoren während der Studie zum Erfassen von Daten beitragen.

²University of California - Experience Sampling Method (ESM) <http://psc.dss.ucdavis.edu/sommerb/sommerdemo/sampling/behavior.htm>

2.4 Stress und kognitive Last beim Autofahren

Es gibt viele verschiedene Faktoren für Stress, dem Autofahrer unterliegen können. In [GMG+89] wird der Fahrerstress als ein komplexes Syndrom beschrieben, welches durch situationsbezogene und externe Faktoren bestimmt wird. So kann Stress durch unterschiedliche Auswirkungen sichtbar werden, wie z. B. emotional als Folge großer Besorgnis, physiologisch durch eine erhöhte Herzschlagfrequenz oder durch aggressives Fahrverhalten. Zudem wird unterschieden zwischen situationsbedingtem und langanhaltendem Stress. Ersterer kann durch plötzliche Ereignisse während der Fahrt ausgelöst werden, wie z. B. Unfälle, wohingegen letzterer sich in lang andauernden Situationen zeigt, etwa wenn der Fahrer schon längere Zeit im Stau steht. Nach Hennessy und Wiesenthal [HW99] sind besonders Verkehrsstauungen ein häufiger Auslöser für erhöhten Stress.

In [WMB+13] wurde in der durchgeführten Studie herausgefunden, dass Fahrer bei ungefähr jeder fünften Fahrt mit Stress zu kämpfen hatten. Die Gründe lagen hierbei in der Verkehrsdichte, Zeitdruck und anderen Verkehrsteilnehmern. Auch unbekannte Straßen, das Wetter und private sowie geschäftliche Vorkommnisse hatten Einfluss auf die Höhe des Stresses. Des Weiteren kann auch das Befahren eines Tunnel kann zu erhöhtem Stress führen [MR14]. Gründe hierfür können z. B. die Enge, Dunkelheit, Monotonie oder der erhöhte Lärmpegel sein.

In der heutigen Zeit ist das Fahren des Autos nicht zu jeder Zeit die primäre Aktion, welche durch den Fahrer ausgeführt wird. Durch viele Nebentätigkeiten im Auto, wie z. B. das Bedienen des Infotainmentsystems, des Navigationsgeräts oder des Mobiltelefons wird der Fahrer von seiner eigentlichen Aufgabe leicht abgelenkt [SPB+13]. Oft sind Fahrer nicht alleine unterwegs und können durch andere Passagiere beeinflusst und abgelenkt werden. Alle diese Faktoren tragen zu einer Erhöhung der kognitiven Last bei und erschweren die eigentliche Aufgabe des Fahrens. Die Höhe der kognitiven Last kann entweder objektiv durch Sensoren und Messungen oder subjektiv durch Befragung des Fahrers bestimmt werden.

2.5 Ergebnisse der Literaturrecherche

In diesem Kapitel wurden verwandte Arbeiten zu den Themen Routenberechnung, Erlebniserfassung und Stress untersucht. Bei den Algorithmen stellte sich der A*-Algorithmus als der geeignetste heraus, wenn ein kürzester Weg gefunden werden soll. Dies liegt an seiner Einfachheit und zielgerichteten Suche. Im Bereich der alternativen Routenberechnung ist besonders die Arbeit von Pfleging et al. [PSMT14] von Interesse. Die hier beschriebenen Aspekte für die Bestimmung der Route anhand des Fahrerlebnisses bilden die Ausgangsbasis dieser Arbeit. Bei der Erlebniserfassung wurden verschiedene Herangehensweisen betrachtet, von denen besonders die Experience Sampling Methode (ESM) vielversprechend aussieht. Zuletzt wurden die Faktoren für das Auslösen von Stress untersucht, um besser zu verstehen, in welchen Situationen Stress beim Autofahren auftreten kann. Hier waren dichter Verkehr und andere

Verkehrsteilnehmer ein oft genannter Auslöser. Mit diesen Erkenntnissen wird im weiteren Verlauf dieser Arbeit ein System geschaffen, welches zum Ziel hat, den Fahrer mit einem besseren Fahrerlebnis und weniger Stress an sein Ziel zu bringen.

3 Grundlagen

In diesem Kapitel werden die Grundlagen zu den verwendeten Werkzeugen und Technologien erläutert. Im ersten Abschnitt werden Informationen zu digitalen Karten und deren Entstehung gegeben. Diese stellen die Basis für die Beschaffung von Daten und ortsbezogenen Informationen dar. Im zweiten Abschnitt wird das mobile Betriebssystem Android beschrieben. Die während der Arbeit entwickelte Anwendung basiert auf Android und setzt diverse Technologien des Betriebssystems und der Geräte, welche mit Android als Betriebssystem ausgestattet sind, ein.

3.1 Digitale Karten

Viele Anwendungen basieren auf der Verwendung von digitalen Karten, welche von Diensten wie Google Maps, Microsoft Bing Maps, Nokia Here Maps und OSM bereitgestellt werden. In diesen Karten lassen sich je nach Verwendungszweck unterschiedlich viele Informationen speichern. Häufig werden diese in verschiedenen Schichten (Layers) gespeichert und können dann je nach Bedarf übereinander gelegt werden. Basierend auf den hinterlegten Daten können Auswertungen der geographischen Informationen erfolgen. Typische Informationen sind hierbei z. B. der Längen- und Breitengrad, die örtliche Höhe, der Untergrund des Gebiets oder Informationen zu Straßen und Gebäuden.

Im Folgenden wird die Entstehung digitaler Karten erklärt. Danach werden die beiden grundlegenden Kartentypen (Rasterkarten und Vektorkarten) beschrieben und zum Schluss mit OSM ein Kartendienst genauer betrachtet.

3.1.1 Entstehung digitaler Karten

Um eine digitale Karte erstellen zu können werden geographische Daten zu Objekten in der realen Welt benötigt. Eine Möglichkeit besteht darin, alle möglichen Arten von Kartenmaterialien, z. B. Land-, Gelände- oder Seekarten, mit Hilfe von Computer Aided Design (CAD)-Programmen zu digitalisieren oder einfach einzuscannen. Auch Luftbilder von Satelliten und Flugzeugen können auf diese Weise umgewandelt werden. Durch die Digitalisierung können Vektorkarten und durch Scannen Rasterkarten erzeugt werden. Diese beiden verschiedenen Formate werden im nächsten Abschnitt 3.1.2 genauer betrachtet.

Bei Open Source Projekten wie OSM werden die Informationen von Hand durch die Benutzer eingetragen. Hierbei setzt man auf die Ortskenntnisse der Teilnehmer oder diese laufen bzw. fahren noch nicht eingetragene Regionen mit spezieller Ausrüstung zum Erfassen von geographischen Informationen ab. Teilweise werden auch spezielle Veranstaltungen und Treffen zur Erstellung von noch nicht oder schlecht erfassten Regionen durchgeführt. Ein Problem bei dieser Form der Kartenerstellung ist die Zuverlässigkeit und Korrektheit der Karteninformationen. In dünn besiedelten Gebieten sind diese Karten zudem oft unvollständig. In [WGS10] werden als mögliche Fehlerquellen in OSM vor allem unerfahrene Editoren und schlechte Ausgangsdaten genannt.

3.1.2 Raster- und Vektorkarten

Digitale Karten können auf zwei unterschiedliche Weisen erzeugt werden. Eine Möglichkeit stellen die sogenannten Rasterkarten dar. Diese bestehen aus Bild-Kacheln mit einer Größe von normalerweise 256x256 Pixeln und werden in einer Gitterstruktur als Karte angezeigt. Sie besitzen in der Regel lediglich eine zweidimensionale Ebene und kommen damit klassischen Papierkarten am nächsten. Der Nachteil dieses Kartentyps liegt darin, dass die Karten beim Vergrößern unscharf und beim Verkleinern unübersichtlich werden. Um diesem Effekt entgegenzuwirken, müssen die Bild-Kacheln in verschiedenen diskreten Zoom-Stufen erstellt werden. Der Vorteil dabei ist, dass dieser Kartentyp sehr schnell angezeigt werden kann, besonders wenn eine geeignete Caching-Strategie zum Einsatz kommt. Zudem können die Kacheln schon vorher gerendert werden, wodurch sichergestellt wird, dass die eingetragenen Informationen in jeder Zoom-Stufe gut leserlich dargestellt werden.

Im Gegensatz dazu stehen die Vektorkarten. Hier werden die Karten mit Hilfe von Vektordaten direkt beim Anzeigen generiert. Die Vektordaten können in den unterschiedlichsten Formaten gespeichert werden, z. B. im Extensible Markup Language (XML)- oder im Binärformat. Sie bestehen aus Informationen zu Knotenpunkten, Wegen und Relationen und werden erst auf dem Gerät in eine graphische Karte transformiert. Ein großer Vorteil von Vektorkarten ist die Möglichkeit, den enthaltenen Elementen Metadaten zuordnen zu können. Dadurch kann der Informationsgehalt der Karte nahezu beliebig gesteigert werden und es lassen sich sehr einfach spezielle Karten erstellen, welche nur die benötigten Informationen enthalten. Auch das Aussehen der Karte kann sehr leicht geändert werden, z. B. von Tag- auf Nachtsicht. Ein weiterer Vorteil ist die, im Vergleich zu Rasterkarten, geringere Dateigröße des Kartenmaterials, da die Informationen zu den verschiedenen Datentypen lediglich einmal gespeichert werden müssen. Hierdurch sind sie besonders im Bereich von mobilen Anwendungen auf Geräten mit begrenztem Speicherplatz von Interesse. Auch gibt es bei diesem Kartentyp keine diskreten Zoom-Stufen, sondern die Karte kann stufenlos vergrößert und verkleinert werden. Hier liegt aber gleichzeitig eines der Hauptprobleme von Vektorkarten. Wenn die Karte vergrößert oder verkleinert wird, muss entschieden werden, ab wann welche Details ein- und ausgeblendet werden. Ansonsten kann die Karte schnell unübersichtlich werden oder recht leer aussehen. Ein Vergleich von Vektor- und Rasterkarten ist in Abbildung 3.1 zu sehen.

Die Möglichkeit des Routings ist aber nur mit Vektorkarten möglich, d. h. alle Dienste, mit deren Hilfe eine Route geplant werden kann, arbeiten mit Vektorkarten oder sind mit Vektordaten überlagerte Rasterkarten. Auch die OSM-Karten werden als Vektorkarten angeboten. Auf diesem Kartentyp arbeitet die in dieser Arbeit verwendete Anwendung. Im folgenden Abschnitt wird OSM als Beispiel für digitale Karten genauer erklärt.

3.1.3 OpenStreetMap

Neben Kartendiensten wie Google Maps oder Bing Maps gibt es ein Open Data Projekt namens OSM. Ein großer Vorteil von OSM ist die freie Nutzbarkeit der geographischen Daten ohne zusätzliche Kosten oder Lizenzbeschränkungen. Das OSM-Projekt wurde 2006 von Steve Coast ins Leben gerufen und zählt aktuell über zwei Millionen Nutzer¹, welche an der Erstellung der Karten mitarbeiten.

Der Detailgrad der Karten ist hierbei teilweise um ein vielfaches höher als bei Google Maps oder Bing Maps, da von Benutzern nicht nur Straßen- und Gebäudekarten, sondern auch spezielle Karten, wie z. B. Fahrrad-, Wander-, Skigebiet- und Seekarten, erstellt werden können. Zudem können in den Karten auch kleine POIs erstellt werden, wie z. B. Parkbänke, Briefkästen, Mülleimer und viele mehr, welche in den meisten anderen Karten nicht eingetragen sind. Leider ist der Detail-Grad in den OSM-Karten regional sehr unterschiedlich, da die Detailmenge sehr davon abhängt, wie viele Menschen an einem bestimmten Ort Daten eintragen und wie deren Präferenzen im Bezug auf die Wichtigkeit bestimmter Aspekte sind.

3.1.4 Datenstruktur

Die Datenstruktur von OSM ist sehr einfach aufgebaut und besteht aus lediglich drei Datentypen als Basis: Knoten, Wege und Relationen. Aus diesen wird alles, was in den Karten angezeigt werden kann, abgeleitet. Im Folgenden werden die drei Datentypen kurz beschrieben.

Knoten

Knoten stellen die kleinste Informationseinheit in OSM dar². Sie sind die einzigen Elemente, welche Koordinaten mit Längen- und Breitengrad besitzen. Jeder Knoten besitzt eine eindeutige ID, über welche er identifiziert werden kann. Optional kann ein Knoten auch mit weiteren Informationen versehen werden. Hierzu gehören z. B. Höhenangaben, Tags oder die Schicht, in welcher sich der Knoten befindet, bei Unter- oder Überführungen. Manche Informationen, wie z. B. POIs, werden in der Regel mit einem Knoten und den für den POI relevanten zusätzlichen

¹OpenStreetMap - Stats <http://wiki.openstreetmap.org/wiki/Stats>

²OpenStreetMap - Node <http://wiki.openstreetmap.org/wiki/Node>

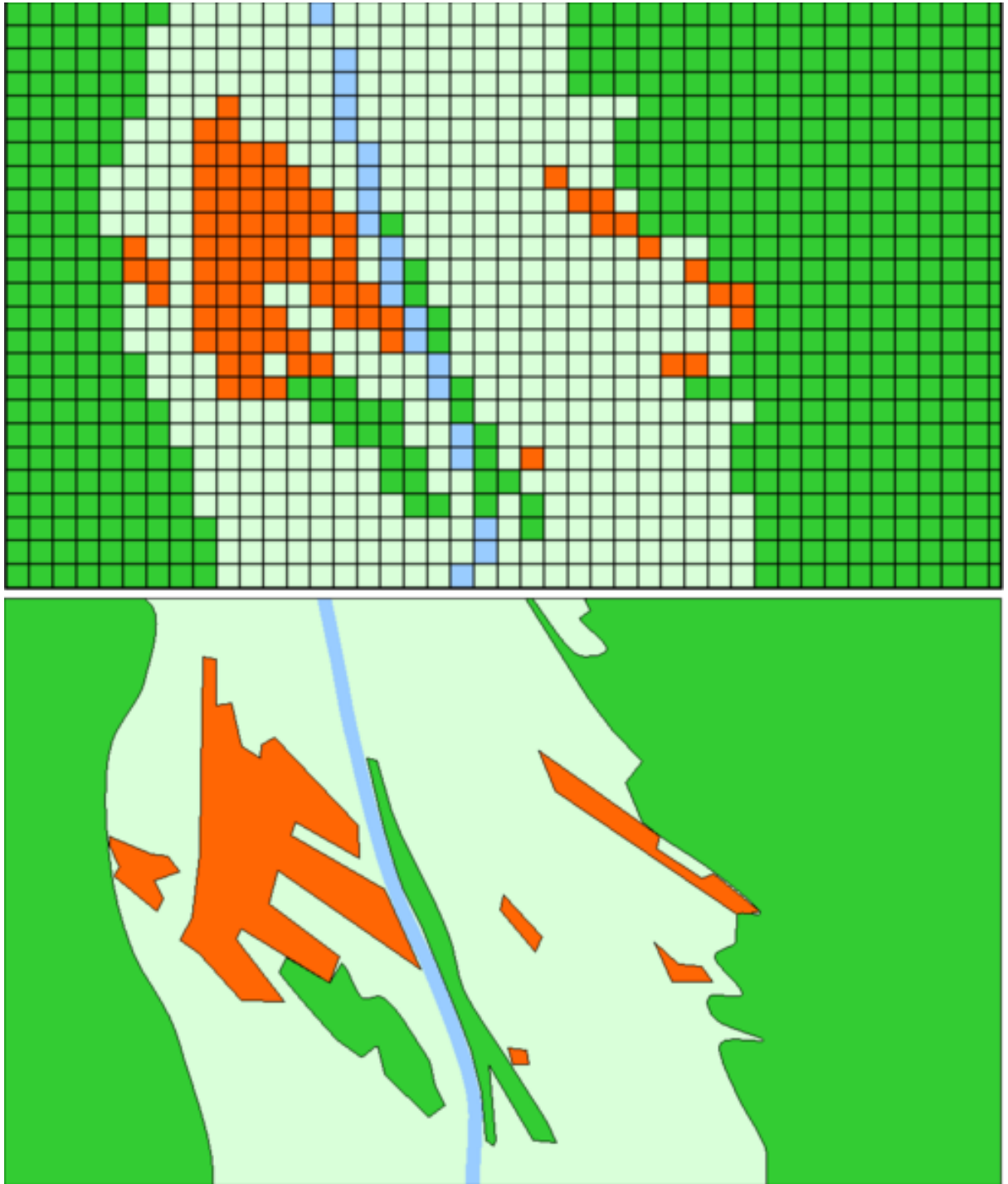


Abbildung 3.1: Unterschied zwischen Vektorkarte (unten) und Rasterkarte (oben) (Quelle: Wikimedia)

Informationen gespeichert [Bah11]. Derzeit sind in den OSM-Daten rund 2,75 Milliarden Knoten eingetragen.

Wege

Aus mehreren geordneten und miteinander verknüpften Knoten werden Wege erstellt und definieren gleichzeitig die Form des Weges³. An den Schnittpunkten von Wegen teilen diese sich auch den Knoten, welcher den Schnittpunkt definiert. Jeder Weg besitzt eine eindeutige ID und sollte zusätzlich einen Tag besitzen, welcher die Bedeutung des Weges beschreibt. Es wird in OSM zwischen offenen und geschlossenen Wegen unterschieden. Offene Wege besitzen unterschiedliche Anfangs- und Endknoten. Die meisten Straßen, Bahngleise oder Flüsse sind als offene Wege definiert. Geschlossene Wege können entweder einen Weg oder ein Gebiet beschreiben. Welche Eigenschaft dem Weg zugewiesen wird, hängt von den angegebenen Metadaten in Form von Tags ab. So wird durch einen passenden Tag ein Kreisverkehr als Weg und ein Park als Gebiet gekennzeichnet. In seltenen Fällen können beide Attribute miteinander kombiniert werden, z. B. wenn ein Kreisverkehr eine Grünfläche umschließt.

Relationen

Um mehrere Knoten, Wege oder Relationen selbst miteinander in Beziehung zu setzen werden Relationen⁴ verwendet. Teil der Relation sind ein oder mehrere Tags, sowie eine geordnete Liste von Elementen. Jedem der Elemente kann zusätzlich eine Rolle zugeordnet werden, welche den Zweck des Elements innerhalb der Relation beschreibt. Ein einfaches Beispiel für eine Relation ist eine Busroute. Diese würde aus Wegen, welcher der Bus befährt, sowie Knoten, welche die Haltestellen definieren, bestehen.

3.2 Android

Android ist ein Betriebssystem für mobile Endgeräte, welches seit 2008 von der Open Handset Alliance (OHA), ein von Google gegründetes Konsortium von aktuell 87 Unternehmen, entwickelt wird. Das Konsortium besteht aus diversen Netzbetreibern, Software-Unternehmen, Geräte-Herstellern, Marketing-Unternehmen und Unternehmen aus der Halbleiterindustrie. Alle Mitglieder haben sich verpflichtet nur Geräte herzustellen, welche zu Android kompatibel sind. Die erste Version von Android wurde im Jahr 2008 veröffentlicht und bildete die Ausgangsbasis für alle darauf folgenden Versionen. Seither hat sich Android enorm entwickelt und

³OpenStreetMap - Way <http://wiki.openstreetmap.org/wiki/Way>

⁴OpenStreetMap - Relation <http://wiki.openstreetmap.org/wiki/Relation>

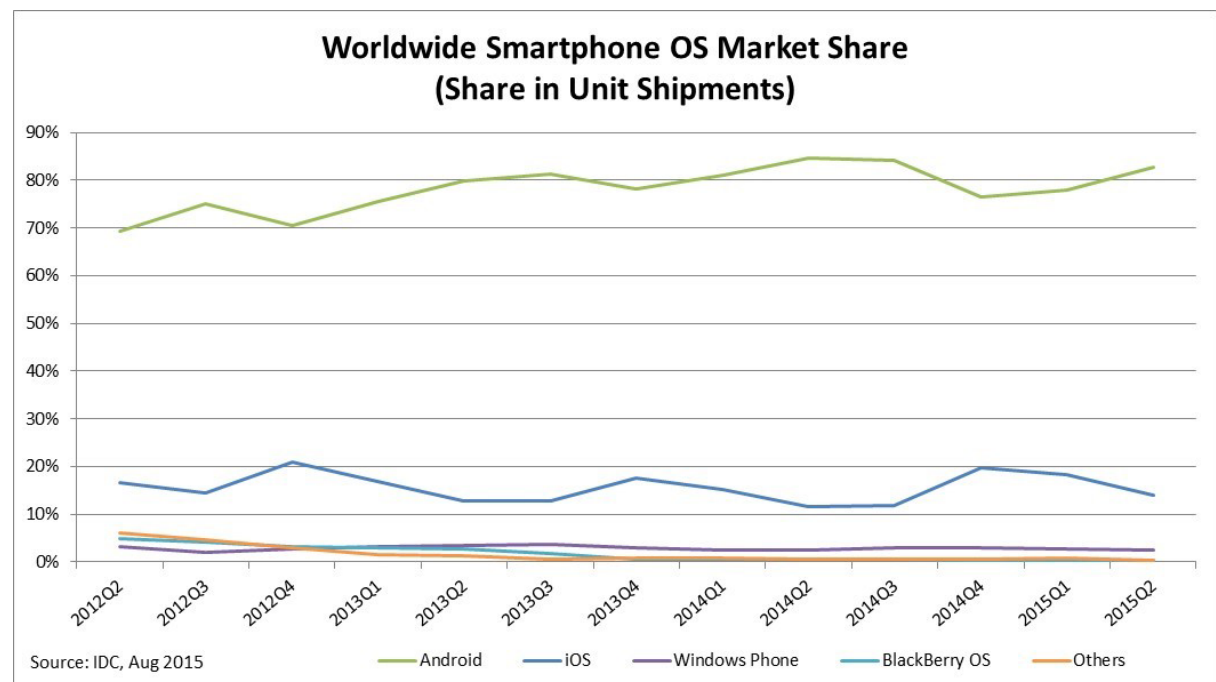


Abbildung 3.2: Marktanteilentwicklung verschiedener Betriebssysteme für mobile Endgeräte.

ist das am meisten verwendete Betriebssystem für mobile Endgeräte mit einem Marktanteil von über 80% (Abbildung 3.2)⁵.

3.2.1 Android Versionen

Die erste Version von Android (1.0) wurde im Oktober 2008 für das T-Mobile G1 veröffentlicht, welches das erste Mobiltelefon mit Android als Betriebssystem war⁶. Wenige Monate später wurde Android 1.1 vorgestellt, welches primär einige Fehler ausbesserte.

Ein großer Schritt wurde mit Android 1.5 (Cupcake) im April 2009 gemacht, in welchem viele neue Funktionen eingeführt wurden, wie z. B. eine On-Screen Tastatur, Unterstützung für Videoaufnahmen und -wiedergabe, die Möglichkeit für Drittanbieter-Widgets und vieles mehr. Mit Android 1.6 (Donut) wurde ein weiterer großer Schritt nach vorne gemacht. Neben der Unterstützung für Code-Division Multiple Access (CDMA), welches Android vor allem in Amerika und Asien interessanter machte, wurde hier der wichtige Grundstein für die Unterstützung von verschiedenen Display-Größen und -Auflösungen gelegt. Dieser Schritt war enorm wichtig, um eine breite Palette von unterschiedlichen Geräten unterstützen zu können.

⁵IDC - Smartphone OS Market Share, 2015 Q2 <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

⁶The Verge - Android: A visual history <http://www.theverge.com/2011/12/7/2585779/android-history>

Im November 2009 wurde mit Android 2.0 (Eclair) die nächste große Version vorgestellt. Besonders die Unterstützung für mehrere Benutzerkonten und die Einführung von Google Maps Navigation bedeuteten einen großen Schritt in der Entwicklung von Android. Mit Android 2.1 wurde ein kleineres Update vorgestellt. Das eigentlich Neue war, dass Google in Zusammenarbeit mit HTC zum ersten Mal ein eigenes Smartphone namens Nexus One vertrieb. Ziel des Projekts war es, Android in seiner Reinform, ohne die von vielen Geräteherstellern überarbeitete Oberfläche, zur Verfügung zu stellen. Als Mitte 2010 mit Android 2.2 (Froyo) die nächste Version vorgestellt wurde, konnte man den Vorteil eines Geräts mit unverändertem Android deutlich erkennen. Denn während die diversen Gerätehersteller noch damit beschäftigt waren, die Oberfläche anzupassen, konnte man für das Nexus One bereits das Update auf die neueste Version herunterladen. Daneben wurde mit Froyo hauptsächlich das Design aufgefrischt und die Möglichkeit eingeführt, einen mobilen WLAN-Hotspot zu erstellen. Auch in Android 2.3 (Gingerbread) wurde hauptsächlich das Aussehen der Oberfläche aufpoliert. Dazu kamen Verbesserungen bezüglich der Energieverwaltung und die Unterstützung von Frontkameras und Near-Field-Communication (NFC). Zusammen mit Samsung wurde zudem ein weiteres eigenes Smartphone, das Nexus S, produziert.

Bislang gab es jedoch keine Unterstützung für Tablets. Diese wurde durch Android 3.0 (Honeycomb) erstmals gewährleistet. Neu waren hier der Wegfall des physischen Home-Buttons, welcher durch einen Software-Button ersetzt wurde, sowie ein neues Paradigma bezüglich der Gestaltung des App-Layouts. So konnten Apps nun eine Action Bar einsetzen, über welche die gängigsten Optionen und Funktionen direkt erreicht werden konnten. Mit Android 3.1 und 3.2 wurden in Honeycomb kleinere Verbesserungen eingeführt, wie z. B. in der Größe veränderbare Widgets.

Honeycomb stellte jedoch nur eine Übergangsversion dar, bevor mit Android 4.0 (Ice Cream Sandwich) wieder eine große Revolution des Betriebssystems bevorstand. Hier wurde erstmals eine Version vorgestellt, die sowohl Smartphones als auch Tablets unterstützte. Die Funktionen und Verbesserungen aus Honeycomb wurden in Ice Cream Sandwich integriert und um einige Funktionalitäten erweitert, wie z. B. ein verbessertes Benachrichtigungssystem, Android Beam, Entsperrung des Geräts durch Gesichtserkennung, eine neue Schriftart für bessere Lesbarkeit auf hochauflösenden Displays und vieles mehr. In Android 4.1 (Jelly Bean) wurde das Hauptaugenmerk auf die Geschwindigkeit von Android gelegt. Hierfür wurde unter dem Namen Project Butter an der Reaktionsgeschwindigkeit auf Eingaben und visuelle Darstellungen gearbeitet. Damit wurde auf den Eindruck der schlechteren Reaktionsgeschwindigkeit gegenüber Apples iOS reagiert. Weiter wurde mit Google Now ein Assistent integriert, der dem Nutzer in alltäglichen Dingen zur Seite steht und sehr intelligent Informationen bereitstellt, welche der Nutzer gerade benötigt - oftmals ohne dass dieser dafür noch extra eine Eingabe machen muss. Dazu gesellten sich einige weitere Verbesserungen bezüglich Benachrichtigungen, Widgets und Textvorhersagen. In den Versionen 4.2 und 4.3 wurde Jelly Bean weiter mit kleineren Verbesserungen versehen. Unter anderem konnten jetzt mehrere Benutzerprofile angelegt werden und die Unterstützung von OpenGL ES 3.0 für das Darstellen von 3D-Inhalten wurde integriert. Mit Android 4.4 (KitKat) wurde die Oberfläche abermals komplett neu gestaltet. Apps konnten von nun an mit einer ausziehbaren Seitenleiste, dem sogenannten Drawer, versehen werden

und Google Now wurde direkt in den Home Screen integriert. Neu war auch der Vollbildmodus, wodurch das eintauchen in eine App ohne störende Statusleiste ermöglicht wurde. Darüber hinaus wurde das gesamte Betriebssystem schneller, effizienter und ressourcenschonender entworfen, womit auch ältere Geräte auf KitKat aktualisiert werden konnten.

Seit Android 5.0 (Lollipop) setzt Google auf ein neues Designparadigma namens Material Design. Damit wird das Ziel verfolgt eine einheitliche visuelle Sprache für den Benutzer zur Verfügung zu stellen, welche die klassischen Prinzipien von gutem Design mit modernen technologischen Möglichkeiten vereint. Nachdem in Jelly Bean mit Project Butter die Reaktionsgeschwindigkeit verbessert wurde, widmete man sich in Lollipop mit Project Volta der Verbesserung der Batterielaufzeit. Teil des Projekts sind einige Werkzeuge, um die Nutzung der Batterie zu optimieren, sowie ein neuer Stromsparmodus. Des Weiteren dehnte Google die Unterstützung von Android auf mehr Geräte aus. Dazu gehören jetzt neben Smartphones und Tablets auch Fernseher und vor allem Smartwatches. Ein weiterer Schritt, um Android auch in Bereichen abseits von mobilen Handgeräten zu verbreiten, ist Android Auto. In Zusammenarbeit mit verschiedenen Herstellern der Automobilbranche bringt Google hiermit oft genutzte Funktionen von Smartphones direkt ins Auto. Mit Android 5.1 wurden einige Fehler und Probleme von Lollipop behoben, bevor Google im Oktober 2015 seine aktuelle Version von Android veröffentlichte.

Die aktuelle Android Version ist 6.0 (Marshmallow)⁷. Diese baut auf Lollipop auf und bringt einige neue Funktionen und Verbesserungen mit sich. Insbesondere die Unterstützung für Fingerabdruckererkennung, das bessere Batterie-Management und die Möglichkeit, die Rechte für jede App einzeln zu setzen, sind hierbei zu nennen.

3.2.2 Kompatibilität

Google unterscheidet zwischen Android- und App-Kompatibilität. Ob ein Gerät zu Android kompatibel ist, entscheidet Google durch eine Reihe von Tests [And15], welche jeder Hersteller durchführen und bestehen muss. Die Kompatibilität einer App zu einem Gerät hängt davon ab, ob das jeweilige Gerät alle in der App verwendeten Hard- und Software Komponenten besitzt, wie z. B. benötigte Sensoren.

Da eine App auf einer Vielzahl unterschiedlicher Geräte mit unterschiedlichen Versionen lauffähig sein soll, ist es notwendig, eine hohe Kompatibilität zu gewährleisten. Hier bietet Android die Möglichkeit an, für verschiedene Konfigurationen jeweils spezifische Ressourcen zur Verfügung zu stellen. Ein Beispiel hierfür sind die unterschiedlichen Displaygrößen und -auflösungen, für welche mit Hilfe von XML Dateien verschiedene Anordnungen der Elemente auf dem Display festgelegt werden können. Das System erkennt die Gerätespezifikation und wählt dann die passenden Ressourcen für genau diese Spezifikation aus. Es ist jedoch zwingend

⁷GottaBeMobile - Android 6.0 vs Android 5.1 Lollipop: What's New in Marshmallow <http://www.gottabemobile.com/2015/10/05/android-6-0-vs-android-5-1-lollipop-whats-new-in-marshmallow/>

notwendig, dass bei der Entwicklung von Apps diese unterschiedlichen Anforderungen bedacht und die nötigen Ressourcen bereitgestellt werden.

3.2.3 Architektur

Die Android Architektur ist als Schichtenmodell aufgebaut, wie in Abbildung 3.3 dargestellt. Die unterste Schicht ist ein angepasster Linux-Kernel, welcher die grundlegende Speicher- und Prozessverwaltung regelt und Treiber für diverse Hardware-Komponenten beinhaltet. Dieser stellt eine Abstraktionsebene zwischen den Hardwarekomponenten und der Software dar.

Auf der nächsten Schicht befinden sich verschiedene Bibliotheken von Linux zusammen mit einigen Android spezifischen Bibliotheken. Zudem befindet sich hier noch die Android Runtime. Diese beinhaltet die Dalvik Virtual Machine, welche eine für Android angepasste Java Virtual Machine ist. Zusätzlich beinhaltet die Android Runtime einige Core Libraries, die es Entwicklern ermöglichen, Android Apps in reinem Java zu schreiben.

Darüber befindet sich das Application Framework, welches Entwicklern Zugriff auf verschiedene APIs gibt. Ein großer Vorteil hiervon ist die Wiederverwendbarkeit von Komponenten. So kann jede App ihre Fähigkeiten und Ressourcen veröffentlichen, welche dann wiederum von anderen Apps genutzt werden können.

Die oberste Schicht stellen die Apps selbst dar. Einige Apps sind bereits vorinstalliert, teilweise von Google selbst, teilweise von den jeweiligen Geräteherstellern und Providern. Jede App, die ein Nutzer zusätzlich installiert, findet sich in dieser Schicht wieder.

3.2.4 Software Development Kit

Das Android Software Development Kit (SDK) stellt Entwicklern alle Werkzeuge und Bibliotheken zur Verfügung, welche für die Entwicklung von Android Apps benötigt werden. Zu den Werkzeugen gehören unter anderem die Android Debug Bridge (ADB), Dalvik Debug Monitor Server (DDMS), der Android Emulator und der SDK Manager.

Mit Hilfe der ADB kann per Universal Serial Bus (USB)-Verbindung vom Computer auf das Smartphone zugegriffen werden. Dies ist notwendig, um selbst erstellte Apps zu installieren und diese zu debuggen. Hierfür steht mit dem DDMS ein Werkzeug zum Auslesen und Auswerten von Log Nachrichten, Speicher- und Prozessorzuständen zur Verfügung. Dies funktioniert sowohl mit realen Geräten als auch mit dem Android Emulator. Mit diesem können Geräte simuliert werden, was den großen Vorteil hat, dass mit wenig Aufwand verschiedene Gerätespezifikationen getestet werden können. Hierfür können mit dem SDK Manager verschiedene Versionen von Android, sowie zur Erstellung von Android Apps benötigte Werkzeuge heruntergeladen und installiert werden.



Abbildung 3.3: Aufbau von Android. (Quelle: Documentation for Android SDK)

3.2.5 Native Development Kit

Während das SDK primär die Entwicklung von Apps mit Java ermöglicht, können mit Hilfe des Native Development Kit (NDK) Teile einer App in Programmiersprachen wie C oder C++ geschrieben werden. Dies ist besonders bei rechenintensiven Aufgaben im Bereich von Grafik-Rendering, physikalischen Berechnungen oder Signalverarbeitung performanter. Oftmals gibt es zudem schon Bibliotheken die in einer nativen Sprache geschrieben wurden. Dank des NDK müssen diese nicht noch einmal neu geschrieben werden, sondern lassen sich direkt als externe Bibliotheken einbinden. Listing 3.1 ist ein Beispiel für den Aufruf einer nativen Methode aus der Java Umgebung heraus.

⁸Android Developer - NDK <http://developer.android.com/ndk/index.html>

Listing 3.1 Beispiel für den Aufruf einer nativen Methode aus Java heraus⁸

```
public class MyActivity extends Activity {  
    /**  
     * Native method implemented in C/C++  
     */  
    public native void computeFoo();  
}
```

3.2.6 Manifest

Jede Android App muss eine Manifest Datei im XML-Format in ihrem Wurzelverzeichnis besitzen. In dieser werden essentielle Informationen über die App gespeichert und vom Android System ausgelesen. Unter anderem wird hier in der Regel der Package Name festgelegt, über welchen eine App eindeutig identifiziert werden kann. Zudem werden hier alle Komponenten und Prozesse einer App aufgelistet, sowie alle Berechtigungen die benötigt werden. Auch die minimale Anforderung an die Android Version wird hier festgelegt, um zu verhindern, dass es zu Kompatibilitätsproblemen kommt.

3.2.7 Signierung

Bevor eine App auf einem Android Gerät installiert werden kann, muss sie digital mit einem Zertifikat signiert werden. Über dieses kann der Autor bestimmt werden, welcher eine bestimmte App anbietet. Es wird zwischen Debug- und Release-Signierung unterschieden. Für die Debug-Signierung wird die App mit einem vom Android SDK generierten Schlüssel (Key) signiert. Eine App, welche mit einem Debug-Key signiert ist, kann jedoch nicht in Googles Play Store veröffentlicht werden. Hierfür ist die Signierung mit einem Release-Key erforderlich. Nötig sind dafür die Erstellung einer Keystore-Datei, welche alle privaten Keys enthält, sowie ein privater Key, welcher dem Keystore hinzugefügt wird.

3.2.8 Android Studio

Die Entwicklung der in Abschnitt 5.3 beschriebenen Anwendung Happy Navi (OsmAnd) wurde mit Hilfe von Android Studio umgesetzt. Diese plattformunabhängige Entwicklungsumgebung wird seit dem Jahr 2013 von Google angeboten und basiert auf IntelliJ IDEA. Die zuvor unterstützte Eclipse IDE wurde durch Android Studio als offizielle Entwicklungsumgebung von Google ersetzt.

4 Konzept

In diesem Kapitel wird das Konzept für ein System zur Erfassung, Analyse und Verarbeitung von Daten bezüglich Stresswerten beschrieben. Hierfür wird im ersten Abschnitt auf die verschiedenen Möglichkeiten zur Erfassung von relevanten Daten im Fahrzeug eingegangen. Im zweiten Abschnitt werden Möglichkeiten für die Verwendung und Nutzung der gewonnenen Daten beschrieben.

4.1 Erfassung von relevanten Daten

Das Ziel der Datenerfassung ist es zum einen die Stresswerte des Fahrers mit geeigneten Methoden zu ermitteln und zum anderen diese mit Ort und Zeit zu verknüpfen.

Stress wird vom menschlichen Körper mit unterschiedlichen Signalen ausgesendet. Viele können mit geeigneten Sensoren direkt am Körper gemessen werden. In [HP05] werden zur Messung ein Elektrokardiogramm, Elektromyogramm, sowie Sensoren für die elektrodermale Aktivität und die Atmung eingesetzt. Mit diesen lassen sich die Pulsfrequenz, Herzratenvariabilität, Atmung und der emotionale Stress aufzeichnen. Mit Hilfe von auf den Fahrer und die Straße gerichteten Kameras können die gewonnenen Werte in Kontext mit den vorhandenen Umgebungsinformationen gesetzt werden. Wie in [Hei12] beschrieben, können diese Messungen aber auch ohne störende Verkabelung des Fahrers über in den Fahrersitz und Gurt eingebaute Sensoren erfolgen. Eine andere Möglichkeit zur kontaktlosen Messung bieten Wärmebildkameras, welche die Durchblutung der Haut auf der Stirn messen und hier die durch den Blutfluss bedingte erhöhte Temperatur erfassen können [POP+05]. Auch GPS Aufzeichnungen können zur Erfassung von Stress benutzt werden. Mit dem GStress Modell [VAS+14] wurde ein Werkzeug entwickelt, welches anhand von gemessenen physiologischen Daten trainiert wird und damit, basierend auf verschiedenen Vorkommnissen während der Fahrt (bremsen, anhalten, abbiegen), den Stress des Fahrers bestimmen kann. Neben Messungen gibt es weiterhin die Möglichkeit der subjektiven Befragung des Fahrers. Dies kann sehr einfach in schriftlicher Form nach der Fahrt oder über eine Eingabe auf einem geeigneten Gerät während der Fahrt erfolgen. Für diesen Zweck können Fragebögen oder Dialoge mit Likert-Skalen zum Einsatz kommen.

Um die gewonnenen Werte mit Ort und Zeit zu verknüpfen, bieten sich in erster Linie GPS Daten an. Diese können leicht mit Smartphones, Tablets oder Navigationssystemen zur gleichen Zeit wie die physiologischen Stressmessungen generiert werden und enthalten Längen-

und Breitengrad, sowie die Zeit und Geschwindigkeit. Gleichzeitig können diesen auch direkt weitere Daten aus Kartenquellen hinzugefügt werden, wie z. B. Straßennamen und -typ, der Ort, die erlaubte Höchstgeschwindigkeit oder die Anzahl der Fahrspuren. Smartphones und Tablets bieten noch weitere Vorteile. Über die integrierten Sensoren können zusätzlich Beschleunigungswerte, Helligkeit, Umgebungstemperatur und -feuchtigkeit gemessen und aufgezeichnet werden. Der Touchscreen bietet eine hervorragende Möglichkeit um Eingaben des Nutzers zu erhalten und über das integrierte Mikrofon können Geräusche aus der unmittelbaren Umgebung des Geschehens aufgezeichnet werden.

Für die Daten des Fahrzeugs kann auf den Controller Area Network (CAN)-Bus zugegriffen werden. Dieser liefert z. B. Informationen zu Stärke der Beschleunigung und Bremse oder dem Lenkwinkel. Mit Hilfe einer On-Board-Diagnose (OBD) Schnittstelle können diese während der Fahrt ausgelesen und aufgezeichnet werden, um zusätzliche Einblicke zu Vorgängen während der Fahrt zu gewinnen.

Eine weitere Möglichkeit zur Informationsbeschaffung könnte in sozialen Netzwerken und Anwendungen liegen. Auf Facebook, Instagram, Twitter und Google News, um nur einige zu nennen, werden permanent Informationen zu allen möglichen Themen geschrieben. Tejaswin, Kumar und Gupta [TKG15] untersuchten die Möglichkeit Daten zu Verkehrssituationen von Twitter zu extrahieren. Sie bezogen hierfür kontinuierlich öffentliche, verkehrsbezogene Nachrichten von Twitter und analysierten diese in Echtzeit, um Aussagen über Verkehrsaufkommen und Hindernissen auf den Straßen treffen zu können. Die Nachrichten wurden zudem gespeichert, um damit statistische Vorhersagen zu Verkehr und Unfallwahrscheinlichkeiten machen zu können. Dieses Vorgehen wäre auch bei anderen sozialen Netzwerken denkbar und könnte auf stressbezogene Themengebiete erweitert werden. In [SKNI13] wird mit *Social Navigation* ein System vorgestellt, welches von Fahrern bereitgestellte Sprachnachrichten zu Verkehrssituationen an andere Fahrer in der Umgebung weiterleitet. Diese können sich die Sprachnachrichten anhören und darauf basierend ihre weitere Route personalisiert auswählen. Mit diesem System könnten auch sehr einfach Informationen zu Stress von Straßensegmenten bereitgestellt werden.

Ein möglicher Aufbau des Systems ist in Abbildung 4.1 zu sehen. In diesem Aufbau fungiert das Smartphone als zentrale Sammelstelle der Daten und leitet diese an einen Server weiter, auf welchem die Daten dann ausgewertet werden können. Nach der Auswertung schickt der Server die relevanten Daten zur Stressreduzierung zurück an das Smartphone, wo diese dem Benutzer in geeigneter Form präsentiert werden können.

4.2 Verwendung und Nutzung der Daten

Mit Hilfe der gewonnen Daten sollen Aussagen zu den Zusammenhängen zwischen Stress und den durch Sensorik aufgezeichneten Werten getroffen werden können. Hierfür könnten

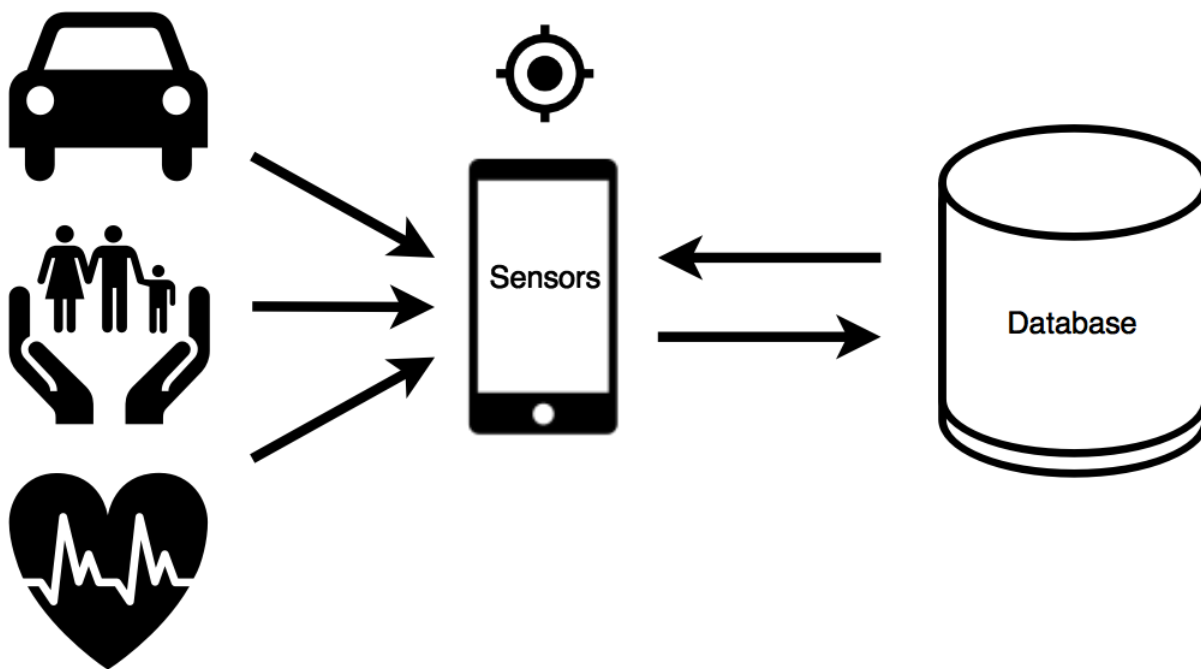


Abbildung 4.1: Beispielhafter Systemaufbau zur Datenerfassung

die Daten beispielsweise auf Korrelationen zwischen den physiologischen Werten, den manuell eingetragenen Stresswerten und den vom Fahrzeug und Smartphone erfassten Daten untersucht werden. Interessant sind hier die sensorisch erfassten Werte von Smartphone und Fahrzeug zu guten oder schlechten manuellen Bewertungen und den physiologischen Werten. Über diese könnten Erkenntnisse darüber gewonnen werden, in welchen Situationen welche Faktoren einen Einfluss auf das Empfinden des Fahrers haben. Beispielsweise kann anhand von Beschleunigungsdaten von Smartphones auf die Gleichmäßigkeit der Fahrt geschlossen werden. Das Gleiche gilt für die in den GPS-Daten gespeicherten Geschwindigkeitswerten.

Durch diese Daten könnte es beispielsweise möglich sein, allein anhand der sensorischen Werte Aussagen über den Stress auf einer Route bestimmen oder sogar vorhersagen zu können. In einem weiteren Schritt könnte damit die Route automatisiert dahingehend angepasst werden, dass ein Maximum an Stress nicht überschritten wird. Eine andere Möglichkeit bestünde darin, dem Nutzer verschiedene Routen mit deren Stressgehalt zu präsentieren und ihn selbst entscheiden zu lassen, welche der verfügbaren Optionen er gerne benutzen würde. In Abbildung 4.2 sind beispielhaft drei sehr ähnliche Routen von Stuttgart nach Berlin abgebildet. Die Routen unterscheiden sich in Länge und Zeit nur sehr wenig voneinander. Jeder Route wird hier beispielhaft ein Stresswert angefügt, welcher dem Fahrer signalisiert, wie hoch der Stress der jeweiligen Route ist.

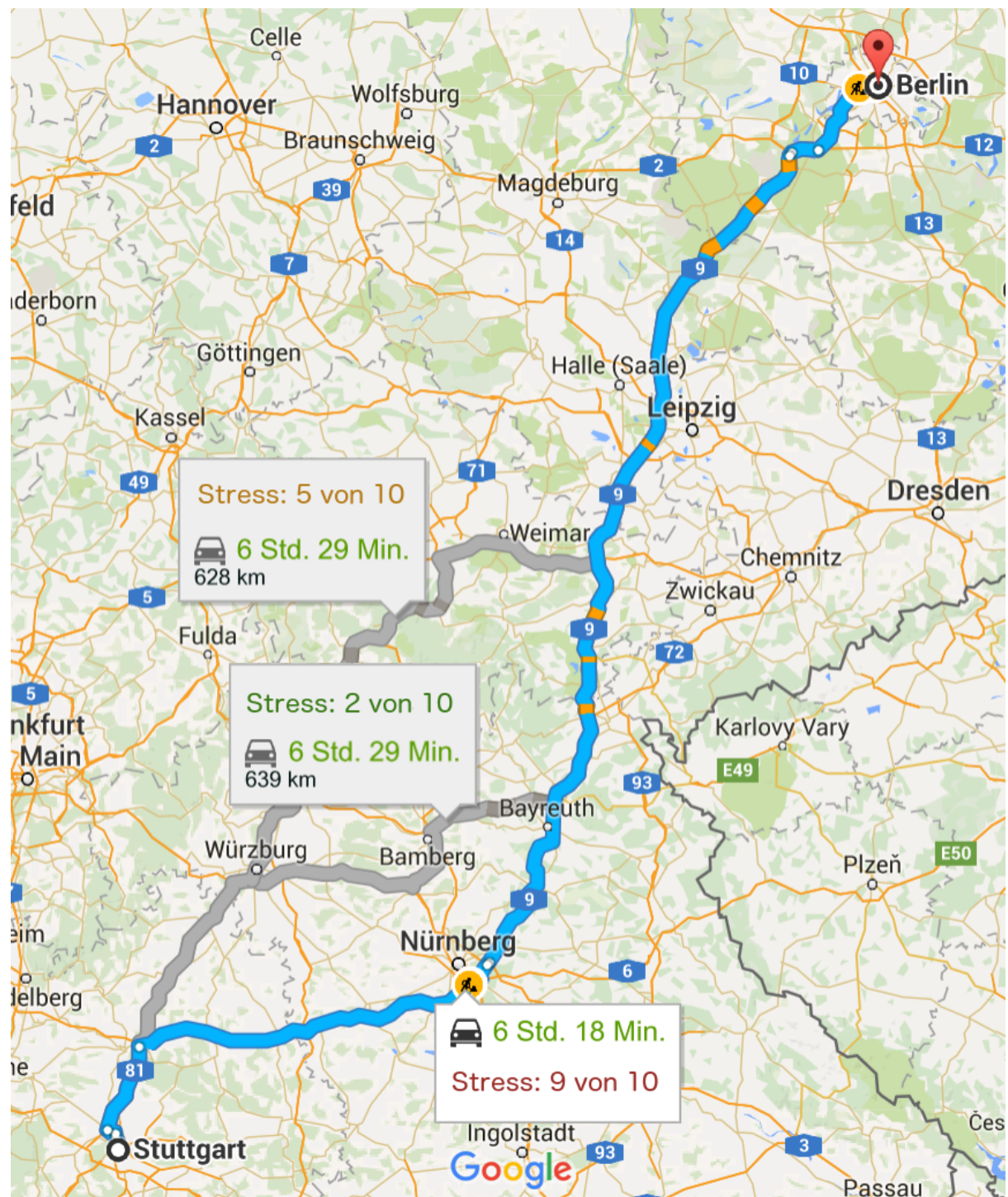


Abbildung 4.2: Beispiel für verschiedene Routen mit ähnlichen Eigenschaften (Quelle: Google Maps)

5 Implementierung

5.1 Anforderungen

Ziel der Implementierung war es, bestehend auf einem stabilen Open-Source Navigationsprogramm, die erforderlichen Änderungen für das Sammeln der Stresswerte sowie die Anpassung der Routenberechnung vorzunehmen. Als Plattform wurde hierfür Googles Android gewählt, welches das weltweit am weitesten verbreitete Betriebssystem für mobile Endgeräte ist. Bei den frei verfügbaren Navigation-Apps für Android wurden unter anderem die Programme OsmAnd¹, AndNav2², Navit³ und ZANavi⁴ betrachtet. Aufgrund der vielen Funktionen und der Möglichkeit Plugins zu entwickeln und zu benutzen, fiel die Wahl auf das weit verbreitete und erprobte OsmAnd für Android, welches seit 2011 über den Google Play Store erhältlich ist. Basierend auf dem frei verfügbaren Quellcode wurde daraufhin die in Abschnitt 5.3 beschriebene Version Happy Navi (OsmAnd)⁵ entwickelt.

5.2 OsmAnd

OsmAnd ist eine Open-Source Navigations-App für Android, welche von mehreren Entwicklern unter der Führung von Victor Shcherb entwickelt wird. Es dient als Grundlage für die während der Diplomarbeit ausgearbeitete Version von OsmAnd: Happy Navi (OsmAnd).

Für Android sind eine Vielzahl von Navigation-Apps über den Google Play Store verfügbar. Viele davon sind kostenpflichtig, wie z. B. Navigon von Garmin, oder bieten den vollen Funktionsumfang nur durch In-App-Käufe an, wie es beispielsweise bei TomTom Traffic der Fall ist. OsmAnd wird im Play Store in einer kostenlosen Version mit beschränkter Anzahl Kartendownloads und ohne Wikipedia-Artikel sowie in einer kostenpflichtigen Version mit vollem Funktionsumfang angeboten. Der Quellcode von OsmAnd ist bei GitHub⁶ öffentlich

¹OsmAnd <http://osmand.net/>

²AndNav2 <http://www.andnav.org/>

³Navit <http://www.navit-project.org/>

⁴ZANavi <http://info.zanavi.cc/>

⁵Happy Navi (OsmAnd) <https://maps.hci.simtech.uni-stuttgart.de/info>

⁶OsmAnd Quellcode <https://github.com/osmandapp/Osmand>

zugänglich und es kann in wenigen Schritten eine eigene Version mit vollem Funktionsumfang erstellt werden.

5.2.1 Funktionen

Der Funktionsumfang von OsmAnd ist, für eine frei verfügbare App, sehr umfangreich. Die Navigation bietet sowohl offline als auch online Karten für alle Regionen der Welt, Turn-by-turn Sprachanweisungen, verschiedene Ansagen zu Gefahrenstellen oder Geschwindigkeitsüberschreitungen, Fahrspuranweisungen, Zwischenhalte, automatische Routenneuberechnung und die Suche von Adressen nach Typen. Es werden somit alle Funktionen geboten, die sich in anderen käuflichen Navigation-Apps auch vorfinden lassen.

Darüber hinaus werden Routingoptionen für Fußgänger und Radfahrer angeboten, welche durch spezielle Fußgänger-, Wander- und Fahrradrouten-Karten optimal eingesetzt werden können. In den Karten können zusätzlich auch Haltestellen des öffentlichen Verkehrs angezeigt werden und es gibt die Möglichkeit, Wanderungen und Fahrradtouren lokal oder online als GPX-Datei abzuspeichern, um sie beispielsweise mit anderen zu teilen.

In der Kartenansicht können die eigene Position und Orientierung angezeigt werden. OsmAnd unterstützt POIs, Favoriten und verschiedene Überlagerungen für die Karten, wie z. B. spezielle Onlinekarten oder die Satellitenansicht von Bing Maps. Eine herausragende Option ist, dass neben Beschriftungen für Orte in englischer oder lokaler Sprache, diese auch in phonetischer Schreibweise angezeigt werden können. Dies ist besonders bei Aufenthalten im Ausland ein großer Vorteil gegenüber vielen anderen Navigation-Apps. Eine weitere Besonderheit ist die Integration von Wikipedia-Artikeln. Für viele POIs können zusätzliche Informationen direkt aus der Wikipedia-Datenbank abgerufen werden.

5.2.2 Plugins

Die Funktionalität von OsmAnd kann mit diversen Plugins erweitert werden. Diese sind teilweise schon integriert oder können nachträglich installiert werden. Teilweise sind diese Plugins jedoch kostenpflichtig, da sie von Drittanbietern zur Verfügung gestellt werden. Eine Übersicht über die in OsmAnd verfügbaren Plugins⁷ ist in Tabelle 5.1 aufgelistet.

⁷OsmAnd - Features <http://osmand.net/features>

Plugin	Funktion
Online Maps	zeigt Online Maps an und erlaubt das Übereinanderlegen von bis zu zwei Schichten
Höhenlinien und Schummierung	zeigt Höhe der Oberfläche und Flächentönungen an
Trip Aufnahme Werkzeug	Aufnahme von Ausflügen und Speicherung als GPX-Datei
Ski Maps	Karten mit Skiliften und Pisten
Seekarten	Informationen über Meere, Flüsse, Häfen und Segelrouten
Audio und Video Notizen	Erinnerungen von Ausflügen einfach aufnehmen und mit der Route verbinden
OSM Editierung	Daten in OSM hinzufügen oder entfernen
Entfernungsmesser und Planungswerkzeug	Messen von Entfernungen zwischen Orten und erstellen und editieren von GPX-Dateien
Parken	Information wo das Auto geparkt ist und wenn die Zeit abläuft
OsmAnd Entwicklung	für das Testen und Entwickeln von OsmAnd

Tabelle 5.1: Liste der verfügbaren Plugins in OsmAnd.

5.2.3 Kompilieren von OsmAnd

Um OsmAnd zu kompilieren sind mehrere Schritte notwendig. Zunächst müssen folgende Werkzeuge und Softwares installiert werden: Java 1.7, Android SDK und NDK, Python, Google Repo Tool, Git, SVN und Gradle.

Für das Android SDK müssen die SDK Tools und die SDK-Platform Tools sowie eine Version von Android installiert werden. Hier wurde Android 5.0.1 (API 21) verwendet. Zudem werden das Android Support Repository, die Android Support Library, die Google Play Services und das Google Repository benötigt. Das Android NDK wird benötigt um nativen Code zu kompilieren. Nachdem alle Werkzeuge installiert sind, kann mit Hilfe des Repo Tools der OsmAnd-Quellcode von GitHub geklont werden. Bevor nun OsmAnd kompiliert werden kann, müssen noch die Pfade zum Android SDK und Android NDK gesetzt werden. Danach kann OsmAnd mit Hilfe von Gradle erstellt werden, welches alle nötigen Dateien, Abhängigkeiten und die native Bibliothek erzeugt und schließlich daraus eine lauffähige Android-App generiert⁸.

⁸OsmAnd - Building OsmAnd

<https://code.google.com/archive/p/osmand/wikis/GradleCommandLineBuildEnvironment.wiki>

5.3 Happy Navi (OsmAnd)

Für Happy Navi (OsmAnd) wurde die offizielle Version von OsmAnd um ein Plugin namens *Stressreduzierung Plugin* erweitert, mit Hilfe dessen, durch Crowdsourcing, Stresswerte für Straßensegmente gesammelt werden. Die erste Version basierte auf OsmAnd 1.9.0 und die aktuellste Version auf OsmAnd 2.2.0, welche seit November 2015 offiziell verfügbar ist.

5.4 Stressreduzierung Plugin

Im Folgenden wird das während dieser Arbeit entwickelte Plugin zur Stressreduzierung vorgestellt. Dafür wird zuerst eine Übersicht über den Aufbau des Plugins gegeben, danach die Datenstruktur und Fragmente erklärt und zum Schluss die Einstellungsmöglichkeiten des Plugins erläutert.

5.4.1 Aufbau

Das Stressreduzierung Plugin ist in 7 Packages unterteilt:

Connectivity	Hier werden alle Internetverbindungen für Uploads und Downloads behandelt.
Database	Alle erzeugten Daten für die Datenbank werden hier verarbeitet.
Fragments	Die Fragmente für die Dialoge und Anzeigen werden hier erzeugt.
Sensors	In diesem Package werden die GPS- und Beschleunigungssensoren verwaltet.
Simulation	Zu Testzwecken kann mit dem Simulation Package die Fahrt mit dem Auto mit und ohne Navigation getestet werden.
Tools	Hier befinden sich einige Helferklassen für die Berechnungen, Präferenzen und das Erzeugen der UUID.
Voice	Die Spracheingabe wird hier behandelt.

Zusätzlich zu den Packages befinden sich noch drei weitere Klassen im Plugin. Zum einen die Plugin Hauptklasse, welche die OsmAndPlugin Klasse erweitert und die Settings-Klasse, welche die Einstellungsmöglichkeiten für das Plugin bietet. Zum anderen die Klasse Constants, in welcher alle konstanten Variablen hinterlegt sind.

Connectivity

Der ConnectionHandler verwaltet den ConnectionReceiver, DownloadService und UploadService. Mit Hilfe von Up- und DownloadService werden die Datenbank-Dateien vom Gerät

auf den Server geladen und umgekehrt. Hierfür wird ein `WakefulIntentService`⁹ verwendet, über welchen sichergestellt werden kann, dass der Datenaustausch auch bei Ausschalten des Geräts zu Ende geführt wird. Der `ConnectionReceiver` wird benutzt, falls in den Einstellungen die Option für den ausschließlichen Datenaustausch über WLAN aktiviert ist. Wenn die App geschlossen wird und kein WLAN zur Verfügung steht, überwacht der `ConnectionReceiver` die Verbindungen des Geräts und aktiviert den Datenaustausch, sobald sich das Gerät mit einem WLAN-Netz verbindet.

Database

Alle Zugriffe auf die Datenbank werden hier über den `DataHandler` geregelt. Dieser ist für das Schreiben der Datenbankeinträge in den verschiedenen Tabellen zuständig und implementiert das Interface für die Eingabe der Stresswerte über den im nächsten Abschnitt erläuterten Dialog für die Stresswerte.

Fragments

Ein Fragment ist ein Teil der Benutzeroberfläche, welches einen bestimmten Inhalt zur Verfügung stellt. Sie bieten den Vorteil, dass sie wiederverwendet werden können und sich in verschiedenen Benutzeroberflächen einbinden lassen. In Happy Navi (OsmAnd) gibt es zwei verschiedene Arten von Fragmenten. Die einen werden auf dem Dashboard als Karten angezeigt, die anderen erscheinen als Überlagerung des Bildschirminhalts. Über das Dashboard bekommt der Benutzer angezeigt, wenn eine neue Version der App zur Verfügung steht oder er weitere Angaben zu seiner Person tätigen kann. Die als Überlagerung erscheinenden Fragmente haben die Form von Dialogen mit welchen der Benutzer interagieren soll. Dazu gehört vor allem der Dialog, über welchen die Eingabe zur Bewertung der Straßensegmente erfolgt. Alle Fragmente werden hierbei über die `FragmentHandler` Klasse gesteuert.

Sensors

Im `Sensors Package` wird die Hauptaufgabe des Plugins umgesetzt. Hier werden zum einen die Daten des Beschleunigungssensors generiert und zum anderen die verfügbaren Informationen zu Straßeneigenschaften und Stresswerten erfasst. Die Handler Klasse ist hier lediglich für das Starten und Stoppen der Sensoren zuständig, diese arbeiten ansonsten autonom.

In der `SRAccelerometer` Klasse werden die Daten des Beschleunigungssensors mit einer Rate von 5 Hz erfasst und mit Hilfe eines Tiefpass-Filters¹⁰ (Listing 5.1) normalisiert. Durch die Normalisierung wird der Betrag der Erdbeschleunigung aus den Beschleunigungswerten

⁹GitHub - CWAC Wakeful <https://github.com/commonsguy/cwac-wakeful>

¹⁰Android Developers - Motion Sensors http://developer.android.com/guide/topics/sensors/sensors_motion.html

Listing 5.1 Beispiel für die Normalisierung von Beschleunigungssensordaten.

```
public void onSensorChanged(SensorEvent event){
    // In this example, alpha is calculated as  $t / (t + dT)$ ,
    // where  $t$  is the low-pass filter's time-constant and
    //  $dT$  is the event delivery rate.

    final float alpha = 0.8;

    // Isolate the force of gravity with the low-pass filter.
    gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
    gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
    gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];

    // Remove the gravity contribution with the high-pass filter.
    linear_acceleration[0] = event.values[0] - gravity[0];
    linear_acceleration[1] = event.values[1] - gravity[1];
    linear_acceleration[2] = event.values[2] - gravity[2];
}
```

entfernt, sodass nur noch die lineare Beschleunigung des Gerätes zu sehen ist. Für manche Geräte kann die lineare Beschleunigung auch Software-basiert bereitgestellt werden. Da dies jedoch nicht für jedes Gerät verfügbar ist und zudem noch einer Kalibrierung bedarf, wurde auf die Software-basierte Option verzichtet. Jede Erfassung wird um den Längen- und Breitengrad der aktuellen Position, die Bewegungsrichtung und die Geschwindigkeit ergänzt und mit einem Zeitstempel versehen. Die Achsen des Smartphones sind in Abbildung 5.1 abgebildet. Hier ist besonders die in Richtung Fahrer gerichtete Z-Achse von Interesse, welche Daten zu Beschleunigungs- und Bremsvorgängen liefern kann.

Die Daten in der `SRLocation` Klasse werden mit 1 Hz erfasst. Hier werden die Informationen zu den Straßensegmenten, Stresswerten und Durchschnittsgeschwindigkeiten generiert. In den Straßensegmenten sind unter anderem die Identifikationsnummer des Segments, der Name, der Straßentyp, die erlaubte Höchstgeschwindigkeit, die Anzahl an Fahrspuren und ein boolscher Wert für Einbahnstraßen enthalten.

Simulation

Zu Testzwecken befinden sich hier zwei Klassen, welche die Simulation einer Fahrt ermöglichen, sowohl mit als auch ohne Navigation. Diese finden sich nicht in der verfügbaren Version von Happy Navi (OsmAnd) wider, da die Simulation einer Fahrt zu unechten Ergebnissen bei den Stresswerten und somit zu falschen Bewertungen von Streckensegmenten führen könnte.

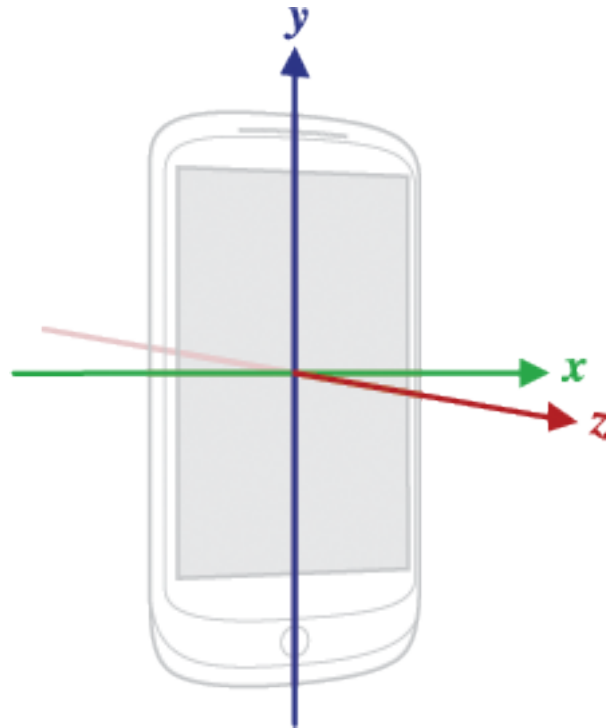


Abbildung 5.1: Achsen eines Android Smartphones. (Quelle: Documentation for Android SDK)

Tools

Im Tools Package befinden sich drei Klassen die zum Teil von mehreren anderen Packages verwendet werden. In der Calculation Klasse finden sich diverse Methoden zur Berechnung und Umwandlung von Werten. Verschiedene Präferenzen des Nutzers werden über die SRSharedPreferences Klasse verwaltet und die UUIDCreator Klasse erzeugt eine einmalige Identifikationsnummer für den Benutzer.

Voice

Um den Dialog zur Eingabe von Stresswerten per Sprache bedienen zu können, wurde hier die Klasse SRPocketSphinx implementiert. Das von der Carnegie Mellon Universität entwickelte PocketSphinx¹¹ ist ein Offline-Spracherkennungsprogramm, welches für verschiedene Plattformen zur Verfügung steht. Die Spracherkennung ist durch die begrenzten Ressourcen von mobilen Endgeräten jedoch nur eingeschränkt möglich und wurde daher als experimentelles Feature implementiert. Für die Eingabe per Sprache wird eine zweistufige Bestätigung

¹¹GitHub - PocketSphinx <https://github.com/cmusphinx/pocketsphinx>

durchgeführt, um die Eingabe ungewollter Bewertungen zu verhindern. Hierfür wird per Sprache zunächst die gewünschte Auswahl getroffen und in einem zweiten Schritt, ebenfalls per Sprache, bestätigt oder abgelehnt. Falls die Auswahl abgelehnt wird, kann der erste Schritt wiederholt werden. Die manuelle Eingabe ist hierbei weiterhin aktiv und kann parallel zur Spracheingabe genutzt werden. Der Schwellwert für die Spracherkennung kann über die Einstellungen angepasst werden. Hierdurch kann verhindert werden, dass die Spracherkennung auf Umgebungsgeräusche reagiert und ungewollte Eingaben getätigt werden.

5.4.2 Datenstruktur

Die Datenstruktur der Datenbank ist in sechs Tabellen gegliedert. Alle Tabellen werden über den Primary Key des Benutzers mit der User Tabelle verknüpft. In dieser werden die Informationen zu den Benutzern gespeichert. Jeder Benutzer besitzt eine eindeutige Identifikationsnummer und zusätzlich Einträge zu Alter, Geschlecht und Typ des genutzten Autos. Die zusätzlichen Angaben sind freiwillig und können über das Dashboard gemacht werden. In der Tabelle App Logs wird gespeichert, welcher Benutzer die App genutzt hat und zu welchem Zeitpunkt. Alle Informationen zu geplanten und gefahrenen Routen werden in der Routing Logs Tabelle gespeichert. Hier werden die Start- und Endkoordinaten hinterlegt sowie die Koordinaten, wo eine Route im Falle des Abbruchs geendet hat. Des Weiteren werden die Zeiten zu Start, vorberechnetem Ende, tatsächlichem Ende, Distanz bis zum Ziel bei Ende der Routenführung und gegebenenfalls die Zeit des Abbruchs der Route eingetragen. Für den Fall dass die Routenplanung mit aktivierter Stressreduzierung Option erfolgte, werden diese Informationen in die hierfür vorgesehenen Felder zu Nutzung der Stressreduzierung Option, Stärke der Stressreduzierung und den Distanzen und Zeiten der normalen und der stressreduzierten Route gespeichert. In der Tabelle Segment Infos werden all diejenigen Informationen gespeichert, welche die Straßensegmente alleinig besitzen. Dazu gehören die Durchschnittsgeschwindigkeit, der Stresswert und der Zeitstempel. Jedes Segment ist hierbei mit dem dazugehörigen OSM Segment verknüpft. Diese werden separat in der OSM Segments Tabelle gespeichert und beinhalten Informationen, welche sich in mehreren Segmenten wiederfinden können. Hierzu gehören die eindeutige Identifikationsnummer des Segments, der Name, der Straßentyp, die Anzahl der Fahrspuren, die erlaubte Maximalgeschwindigkeit und die Information, ob es sich um eine Einbahnstraße handelt. In der letzten Tabelle, Location Infos, werden die Informationen zu einzelnen Orten gespeichert. Jede Location Info besteht dabei aus einem Längen- und Breitengrad, Beschleunigungsdaten für die X-, Y- und Z-Achse, Bewegungsrichtung und Geschwindigkeit sowie einem Zeitstempel.

5.4.3 Einstellungen

Das Plugin kann über die Einstellungen aktiviert und je nach gewünschtem Verhalten angepasst werden. Zu den Einstellungsmöglichkeiten zählen die Aktivierung der stressreduzierten Routenberechnung und die Stärke des Stressreduzierungslevels, die Benachrichtung per Ton



Abbildung 5.2: Dialog zur Bewertung von Straßensegmenten

wenn ein neuer Evaluierungsdialog angezeigt wird, die Aktivierung der Spracheingabe und deren Erkennungsschwelle sowie die Option den Upload der Daten nur über WLAN durchzuführen.

5.5 Erfassung von Stresswerten

Um die Stresswerte zu erfassen wird mithilfe des Stressreduzierung Dialogs (Abbildung 5.2) eine Bewertung der zuletzt befahrenen Streckenabschnitte vorgenommen. Der Benutzer kann dafür eine gute, neutrale oder schlechte Bewertung abgeben, welche mit den Werten 2, 1, und 0 in die Datenbank eingetragen werden. Der Dialog erscheint jedes Mal, wenn das Auto zum Stehen kommt, es sei denn der Fahrer befindet sich immer noch auf dem gleichen Straßensegment, wie bei der letzten Bewertung oder die Distanz oder Zeit seit der letzten Bewertung ist noch nicht groß genug. Falls der Fahrer keine Bewertung abgibt, verschwindet der Dialog nachdem eine bestimmte, minimale Geschwindigkeit für fünf Sekunden überschritten wurde.

5.6 Native Core

Das Rendern von Grafiken sowie das komplexe Berechnen von Routen wird wegen Geschwindigkeitsvorteilen nicht in Java, sondern in C++ durchgeführt. Android bietet dafür die Möglichkeit Teile des Programmcodes in C++ auszulagern und diese als native Bibliotheken in die App einzubinden. In Happy Navi (OsmAnd) wird alles, was Zugriffe auf die Kartendaten

erfordert, der Berechnung der Route dient oder das Anzeigen von Texten und Objekten auf der Karte ermöglicht, über den sogenannten Native Core geregelt und ausgeführt. Für die Berechnung der Route wird der in Abschnitt 2.1.2 auf Seite 17 beschriebene A*-Algorithmus verwendet. Dieser wurde für die Benutzung der gesammelten Stresswerte so verändert, dass diese Werte als zusätzliche Gewichtung für die Heuristik verwendet werden. Die Formel für die Kostenfunktion eines Knotens sieht demnach wie folgt aus:

$$f(n) = g(n) + h(n) \times s(n) \quad \text{mit} \quad \begin{array}{l} f(n) : \text{Gesamtkosten} \\ g(n) : \text{bisherige Kosten} \\ h(n) : \text{geschätzte Kosten bis zum Ziel} \\ s(n) : \text{Kosten des Stresswerts} \end{array}$$

Die Stärke der Stresswerte wird über die Einstellungsoption für das Stressreduzierungslevel geregelt. Hierfür gibt es fünf Stufen, von minimal bis maximal. Bei der minimalen Stufe werden die geschätzten Kosten bis zum Ziel maximal um den Faktor 0,9 nach unten gestaucht oder um 1,2 nach oben gestreckt. Wenn die maximale Stufe ausgewählt wird hat der untere Faktor den Wert 0,1 und der obere Faktor den Wert 10,0. Hierdurch werden die Kosten für bewertete Streckenabschnitte bewusst über- oder unterschätzt, um die Auswahl der Segmente entweder zu verhindern oder zu forcieren.

5.7 Server

Die von den Geräten erfassten Daten werden entweder per WLAN oder Mobilfunknetz an einen zentralen Server gesendet. Hier werden die Daten in eine große SQLite-Datenbank (Abbildung 5.3) eingefügt und aggregiert. Aus dieser wird stündlich eine separate Datenbank generiert, in welcher nur die Identifikationsnummern der Segmente, zusammen mit deren Stresswert, gespeichert sind. Diese werden von der App bei jedem Neustart vom Server heruntergeladen.

5.8 Ausführung einer beispielhaften Routenplanung

Wenn eine Route mit aktivierter Stressreduzierung geplant wird, dann wird die Strecke zweimal berechnet, einmal normal und einmal mit Stressreduzierung. Bisher konnte in OsmAnd lediglich eine Route berechnet werden und es gab daher keine Auswahlmöglichkeiten. Wenn während der Berechnung eine stressreduzierte Route gefunden wurde, kann über den Button zum Wechseln der Route zwischen den beiden Optionen hin und her geschaltet werden. Im Informationsfenster unter der angezeigten Route können hierbei jeweils die Informationen zu Streckenlänge und Reisezeit der ausgewählten Route abgelesen werden. Der Fahrer kann

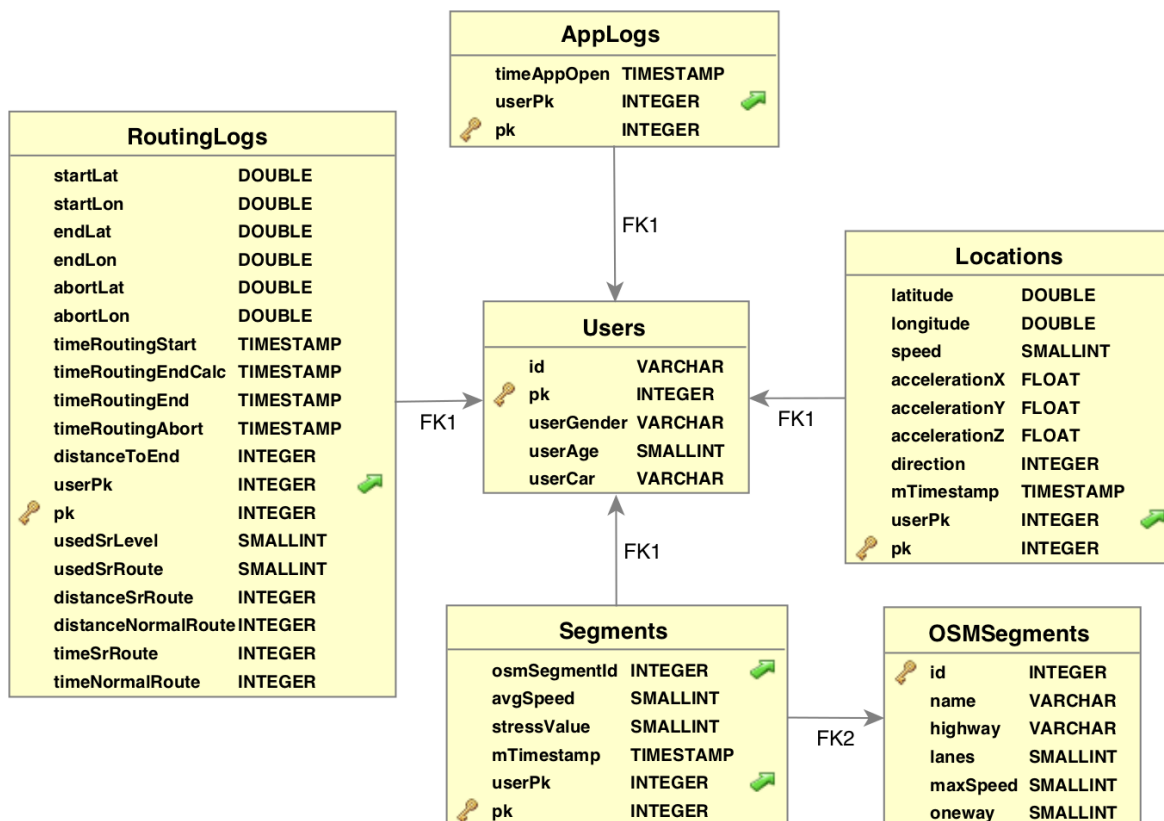


Abbildung 5.3: Schema der Datenbank

somit die beiden Optionen miteinander vergleichen und sich danach für eine der beiden entscheiden.

5.9 Versionen

Happy Navi (OsmAnd) wurde in einer freien Version und einer Vollversion zur Verfügung gestellt. Die freie Version wurde über den Google Play Store verteilt und besitzt eine Downloadbeschränkung für Offline-Karten sowie keine Unterstützung für Wikipedia Artikel. Die Vollversion wurde über die eigene Homepage zum Download angeboten und besitzt keinerlei Einschränkungen. Der Grund hierfür war die Bitte des Hauptentwicklers von OsmAnd nur die freie Version im Google Play Store anzubieten, da OsmAnd hier auch eine kostenpflichtige Version namens OsmAnd+ anbietet. Dies hatte zur Folge, dass permanent zwei verschiedene Version generiert und synchronisiert werden mussten. Zudem konnte die Vollversion nicht auf den Update-Mechanismus des Google Play Stores zurückgreifen. Aus diesem Grund wurde in der Vollversion die Benachrichtigung für eine neue Version auf dem Dashboard angezeigt. Leider muss der Benutzer die neue Version dadurch komplett neu herunterladen und manuell

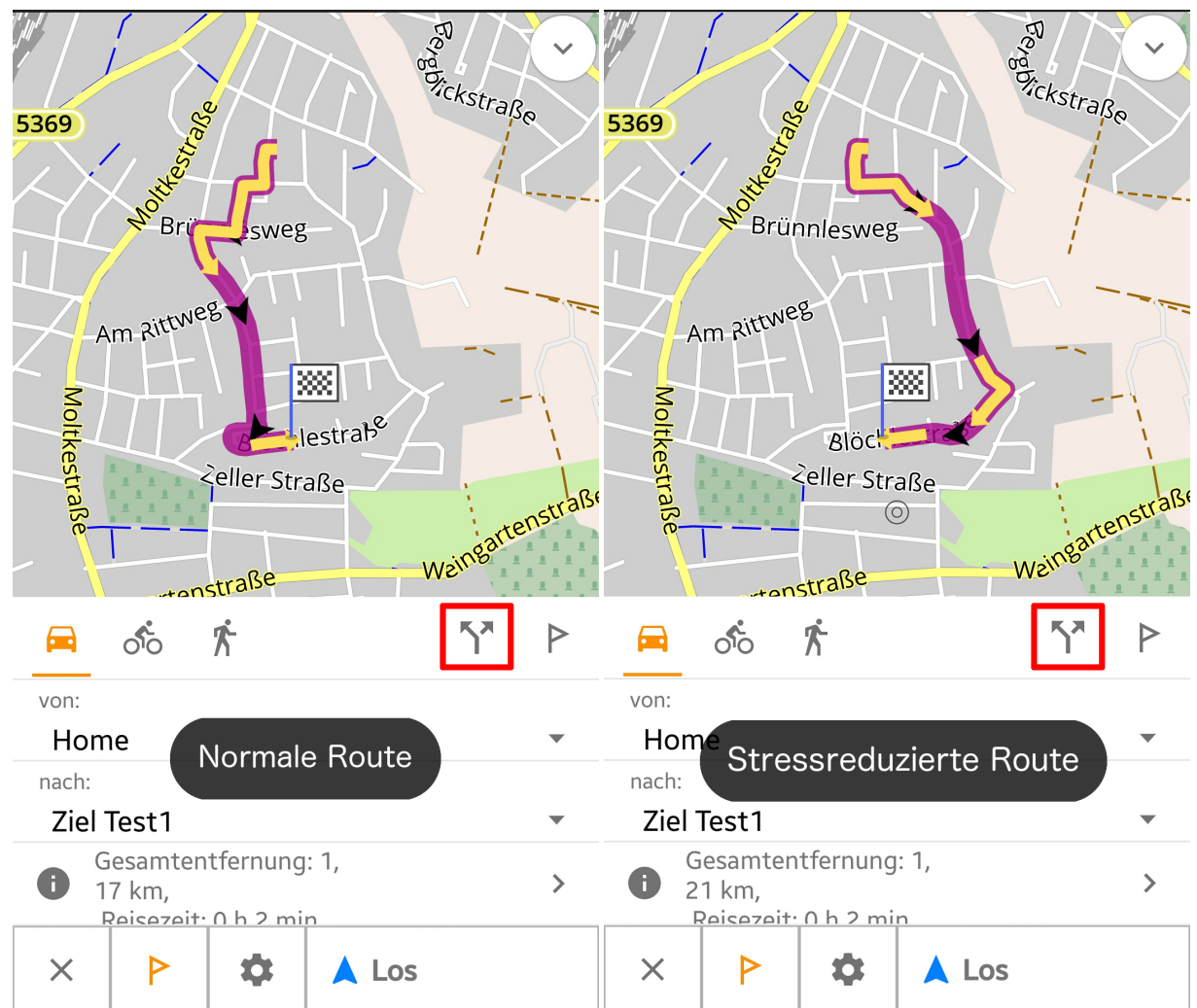


Abbildung 5.4: Beispiel einer normalen und stressreduzierten Route. Der Button zum Wechseln der Route ist rot markiert.

installieren. Auf der anderen Seite kann so eine kostenlose Version von OsmAnd erhalten werden, da sich das Stressreduzierung Plugin auch deaktivieren lässt.

6 Ergebnisse

In diesem Kapitel werden die Ergebnisse der Auswertung von den gesammelten Daten beschrieben. Zuerst wird auf die Nutzung der App eingegangen, danach die gesammelten Daten analysiert und zum Schluss verschiedene Routenplanungen untersucht. Alle Ergebnisse basieren auf den Daten und Zahlen, welche am 31. Januar 2016 vorlagen.

6.1 Nutzung von Happy Navi (OsmAnd)

Zum Zeitpunkt der Auswertung war die freie, über den Google Play Store erhältliche Version von Happy Navi (OsmAnd) auf 96 Geräten installiert und wurde insgesamt 720 mal von verschiedenen Benutzern heruntergeladen. Dazu kommen noch die Benutzer, welche die Vollversion installiert haben, jedoch nicht erfasst wurden. In der Datenbank befinden sich 86 verschiedene Benutzer, welche Daten an den Server übermittelt haben, davon lediglich ein Benutzer, welcher Zusatzangaben zu Geschlecht, Alter und Fahrzeugmodell machte. Die App wurde von den Benutzern 1074 mal an 60 verschiedenen Tagen geöffnet.

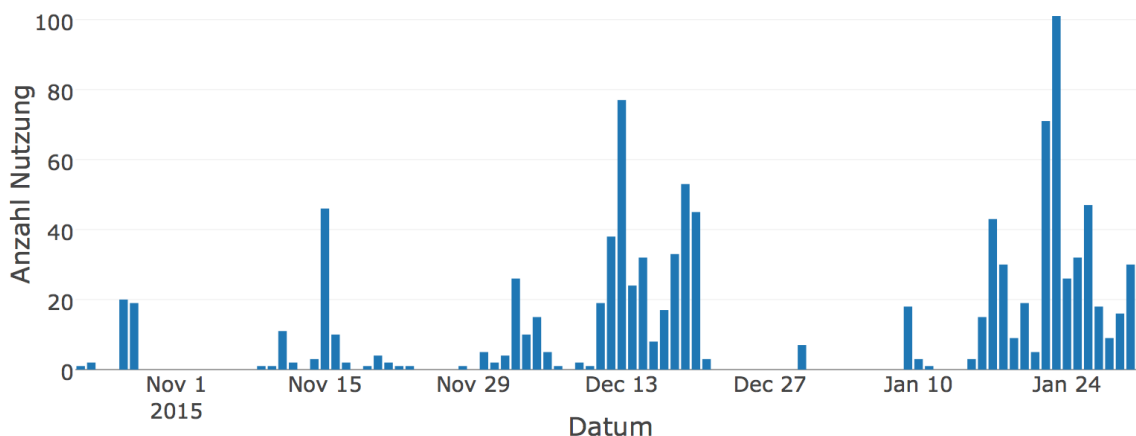


Abbildung 6.1: Anzahl Nutzungen von Happy Navi (OsmAnd) pro Tag.

Eine Übersicht über die Nutzung der App pro Tag ist in Abbildung 6.1 dargestellt. Es ist sehr gut zu erkennen, dass die Nutzung der App über den dreimonatigen Zeitraum seit der Veröffentlichung im Google Play Store stetig ansteigt. Die Nutzungsspitzen liegen dabei auf den Wochenenden, an denen die App deutlich häufiger genutzt wird als unter der Woche. Der höchste Nutzungswert wurde am 23. Januar erreicht, an dem die App über 100 mal geöffnet wurde.

6.2 Datenanalyse

Die bei der Nutzung generierten Daten umfassen die folgenden Einträge in der Datenbank: 1074 AppLogs, 3611681 Locations, 2498 OSMSegments, 4838 Segments, 103 RoutingLogs und 86 Users. Bei den Ortsdaten (Locations) und Straßenabschnitten (Segments) kam es vor, dass Datensätze mehrfach enthalten waren, aufgrund von Fehlern beim Hochladen der Datenbank von den Geräten auf den Server. Zudem wurden zu den meisten Ortsdaten mehrere Einträge geschrieben, da diese mit einer Rate von 5 Hz erfasst wurden. Diese Rate war jedoch nötig um auch kurzzeitige Spitzenwerte bei den Beschleunigungssensoren erfassen zu können. Im Folgenden werden die Schritte beschrieben, mit welchen die Daten bereinigt wurden.

6.2.1 Ortsdaten

In einem ersten Schritt war es notwendig die Einträge in den Ortsdaten zu bereinigen. Es wurde dabei darauf geachtet, dass nur direkt zusammenhängende Ortsdaten zusammengelegt wurden. Für jeden Ort wurde bei der Aggregierung der Durchschnittswert für die Beschleunigungswerte der X-, Y- und Z-Achse, der Geschwindigkeit und der Richtung bestimmt, sowie der Anfangs- und Endzeitstempel. Dies wurde für jeden Benutzer separat gemacht. Nach der Bereinigung befanden sich noch 61221 Ortsdaten in der Datenbank. Die Verteilung der Daten ist in ?? zu sehen. Der Großteil der Ortsdaten wurde für Orte im deutschen Raum erfasst. Daneben wurden auch Ortsdaten in Frankreich, Italien und den Niederlanden generiert. Die Häufigkeit der erhobenen Daten an den einzelnen Orten ist in Abbildung 6.2 als Heatmap zu sehen. Auf dieser ist zu erkennen, dass sich die meisten erfassten Orte auf einige wenige Städte konzentrieren, hauptsächlich Frankfurt, Hannover und Dresden.

6.2.2 Straßenabschnitte

Im zweiten Schritt wurden die Straßenabschnitte aggregiert und für jeden Abschnitt die durchschnittliche Bewertung, d. h. der durchschnittliche Stresswert bestimmt. Die einzelnen Straßenabschnitte konnten mit gut (2), mittel (1) oder schlecht (0) bewertet werden. Nicht bewertete Streckenabschnitte sind mit -99 eingetragen und wurden bei der Aggregierung nicht mit einbezogen. Insgesamt wurden Bewertungen für 2128 verschiedene Streckenabschnitte

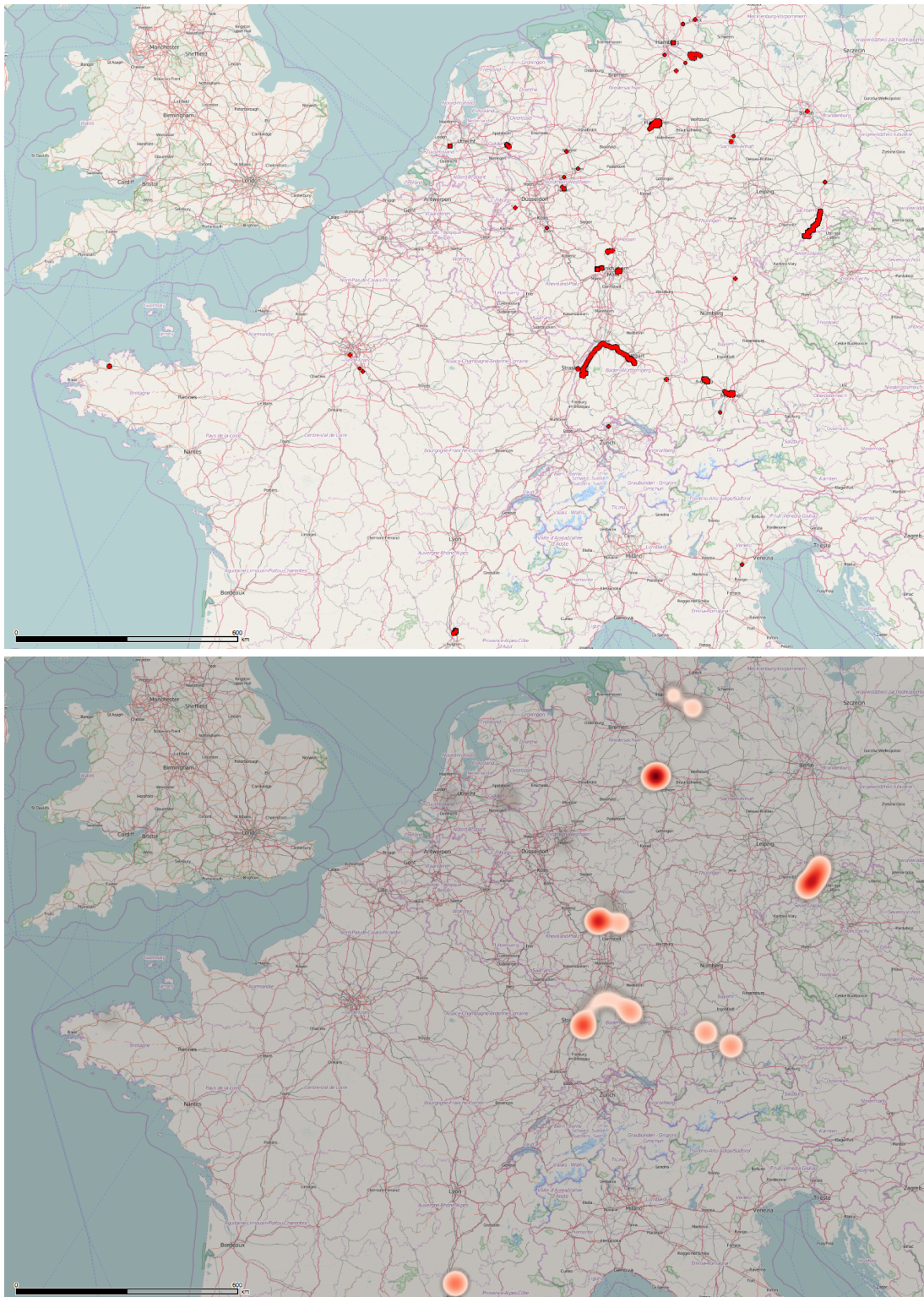


Abbildung 6.2: Übersicht der gesammelten Ortsdaten (oben) und deren Häufigkeit als Heatmap (unten) in Europa.

gesammelt. In Abbildung 6.3 sind die Bewertungen für alle Streckenabschnitte im deutschen Raum abgebildet. Neben einigen kurzen Streckenabschnitten im städtischen Raum, konnten auch zwei längere Fahrten mit durchgängigen Bewertungen erfasst werden. Bei der einen Fahrt handelt es sich um eine längere Fahrt auf der Autobahn, bei der anderen um eine Überlandfahrt auf Bundesstraßen. Die Bewertungen der Streckenabschnitte sind farblich gekennzeichnet. Rote und orange Markierungen stehen für hohen oder erhöhten Stress auf den jeweiligen Segmenten. Gelbe Abschnitte sind neutral bewertet worden. Hellgrüne und Dunkelgrüne Markierungen stehen für entspannte und sehr entspannte Straßensegmente.

6.3 Routenplanung und Routenvergleiche

Insgesamt wurden in den RoutingLogs 90 verschiedene Routenplanungen verzeichnet. Von diesen wurden 49 zu Ende geführt und 41 vorzeitig abgebrochen. Mit aktivierter Stressreduzierung wurden davon 15 Routen geplant, wovon lediglich 6 mal die stressreduzierte Route gewählt wurde. Von den 6 stressreduzierten Routen unterschieden sich 4 in Distanz und Fahrzeit gegenüber den normalen Routen. Im Schnitt waren die stressreduzierten Routen 2741 Meter länger und benötigten 381 Sekunden mehr Zeit gegenüber den normalen Routen.

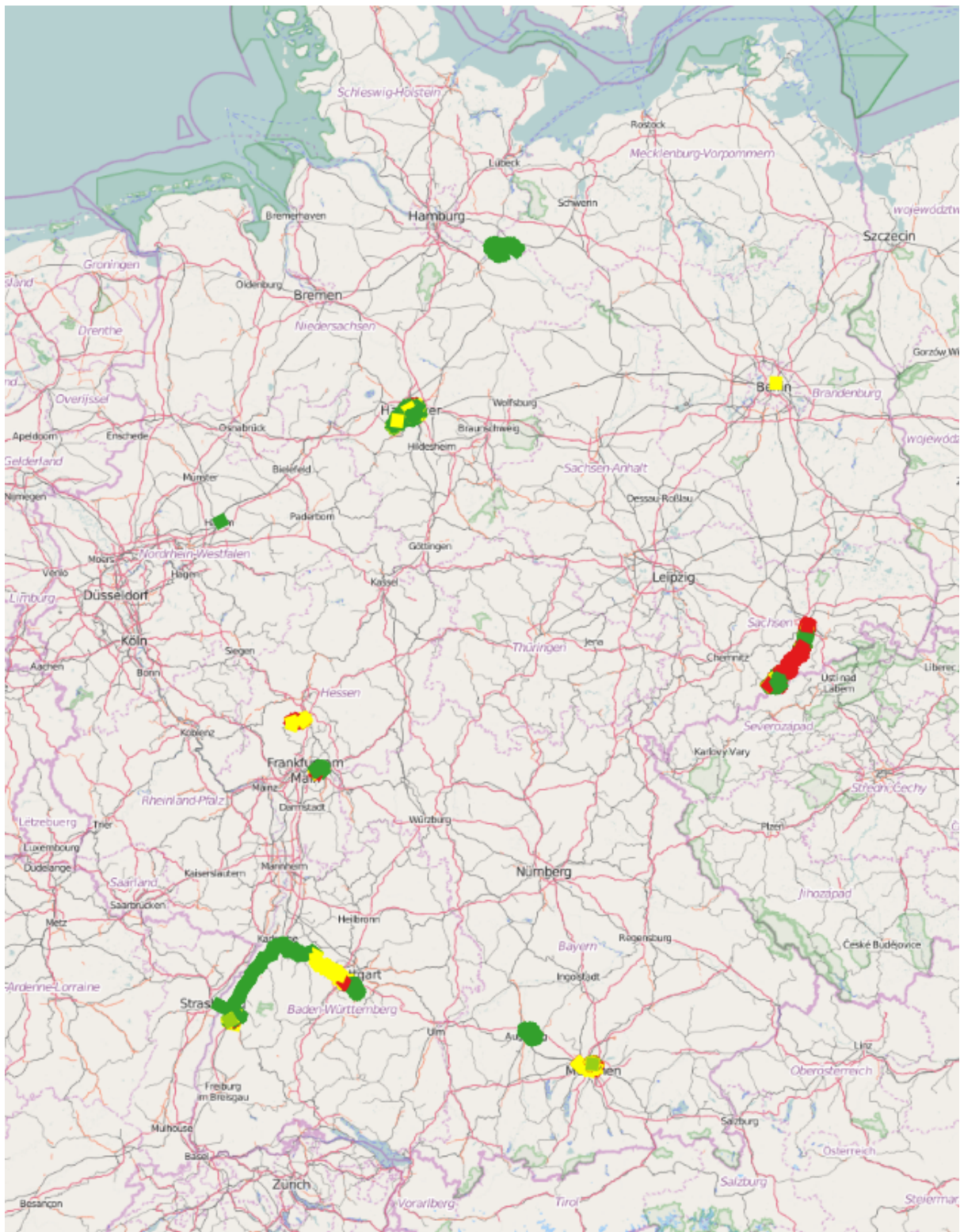


Abbildung 6.3: Bewertungen der Streckensegmente im deutschen Raum.

7 Diskussion

In diesem Kapitel werden die im vorherigen Abschnitt erfassten Daten und Ergebnisse diskutiert. Im ersten Abschnitt werden die Schwierigkeiten und Probleme beim Sammeln der Daten erklärt. Danach wird im zweiten Abschnitt auf Zusammenhänge zwischen Stresswerten und den vorhandenen Ortsdaten eingegangen. Die Auswertung der Daten erfolgte für den deutschen Raum, da hier die meisten Daten gesammelt werden konnten.

7.1 Datenerfassung

Bei der Erfassung der Daten kam es zu einigen Schwierigkeiten, welche die Menge der Daten verringerte. Als ein großes Problem stellte sich die Verwendung von SQLite für die Datenbank heraus. Dieser Datenbanktyp wurde gewählt, da Android damit standardmäßig Datenbanken erstellt und dieser Typ gut in Android integriert ist. In dieser Umgebung funktioniert die Datenspeicherung hervorragend und ohne Probleme. Auf Seite des Servers ist SQLite jedoch nicht zu empfehlen, da es bei gleichzeitigem Zugriff auf die Datenbank von mehreren Nutzern zu Problemen kommen kann. Die während dieser Arbeit verwendete Datenbank wurde bei gleichzeitigem Schreib- und Lesezugriff oft blockiert (gelocked) und konnte nur durch hin und her kopieren der gesamten Datenbank in eine temporäre Datei wieder entsperrt werden. Hier sollte darauf geachtet werden auf der Serverseite einen stabileren Datenbanktyp zu verwenden, wie z. B. PostgreSQL, welches durch Multiversion Concurrency Control (MVCC) das Blockieren der Datenbank verhindert. Zudem fiel der Server wegen eines Hardwaredefekts knapp zwei Wochen aus, in denen keine Daten gesammelt werden konnten.

Trotz allem konnten eine Menge an Daten gewonnen werden, welche Rückschlüsse auf die Gründe für die Bewertungen der Nutzer zulassen. Diese Erkenntnisse werden im folgenden Abschnitt 7.2 diskutiert.

7.2 Korrelation zwischen Stresswerten und Daten

Neben den Stresswerten wurden Daten zu den Beschleunigungssensoren und gefahrenen Geschwindigkeiten der einzelnen Orte gesammelt. Besonders die Daten der Z-Achse sind von Interesse, da über diese Rückschlüsse auf das Beschleunigungs- und Bremsverhalten gezogen werden können, sofern das Smartphone in einer Halterung befestigt ist mit dem

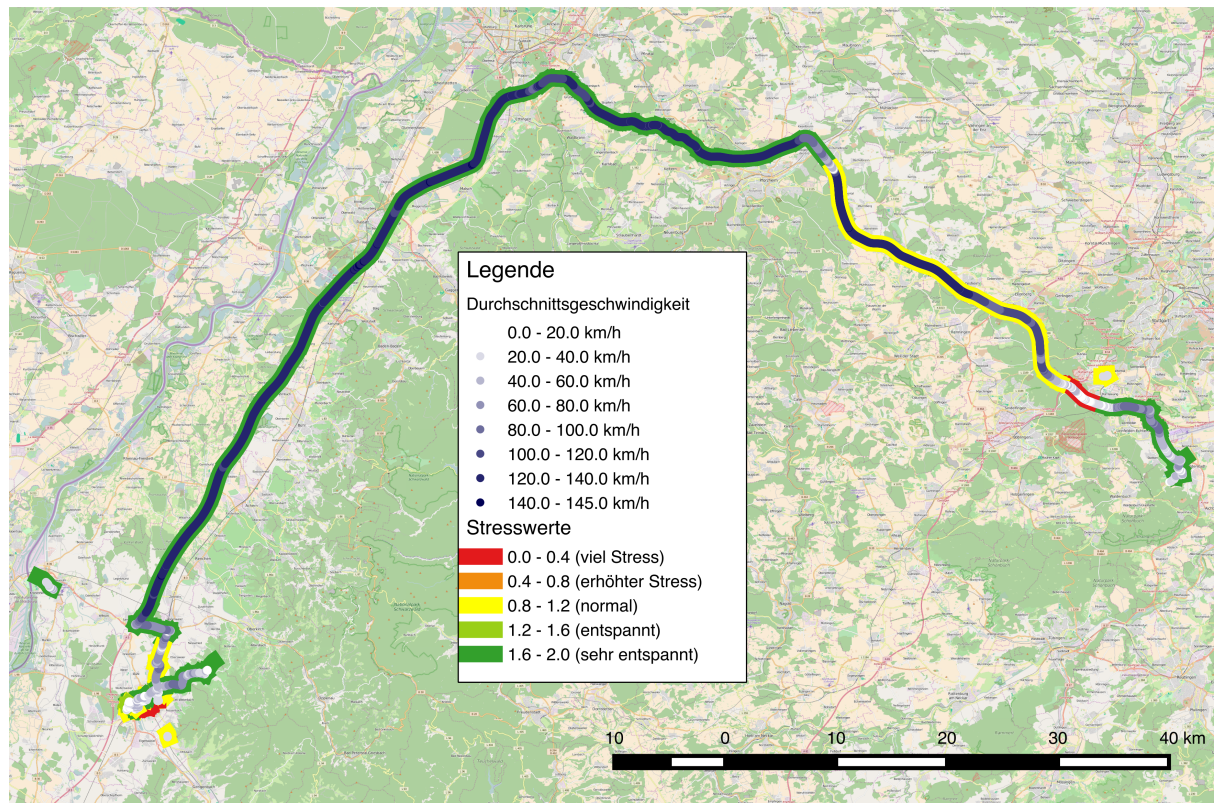


Abbildung 7.1: Beschleunigungswerte und Bewertungen der Fahrt 1.

Display zum Fahrer gewandt. Auch über die Geschwindigkeit lassen sich Rückschlüsse ziehen, beispielsweise bei starken Schwankungen oder Geschwindigkeiten weit unter der zulässigen Höchstgeschwindigkeit. Anhand dieser Werte kann untersucht werden, ob es einen Zusammenhang zwischen Fahrverhalten und den Bewertungen der Streckensegmente gibt.

In Abbildung 7.1 ist eine Fahrt über die Autobahnen A5 und A8 zu sehen. Die Beschleunigungswerte der Z-Achse weisen nur selten starke Beschleunigungs- und Bremsvorgänge auf und es kann angenommen werden, dass die Fahrt relativ gleichmäßig verlaufen ist. Dies wird durch die in Abbildung 7.2 abzulesenden Geschwindigkeitswerte auch belegt, welche nur geringe Schwankungen der Geschwindigkeit während der gesamten Fahrt aufweisen. Die Abschnitte wurden mit normal und gut bewertet, was den Erwartungen anhand der Daten entspricht.

Hier wird allerdings ein Problem im Zusammenhang mit der Bewertung von Autobahn- und Bundesstraßenabschnitten sichtbar. Auf diesen erfolgen Bewertungen nur sehr selten, beziehen sich dann aber auf sehr viele Abschnitte auf einmal. Eine Bewertung erfolgt vermutlich häufig nur dann, wenn der Fahrer in einen Stau gerät und zum Stehen kommt. Dies könnte sich negativ auf zurückliegende Streckenabschnitte auswirken, welche jedoch normal oder sogar gut zu

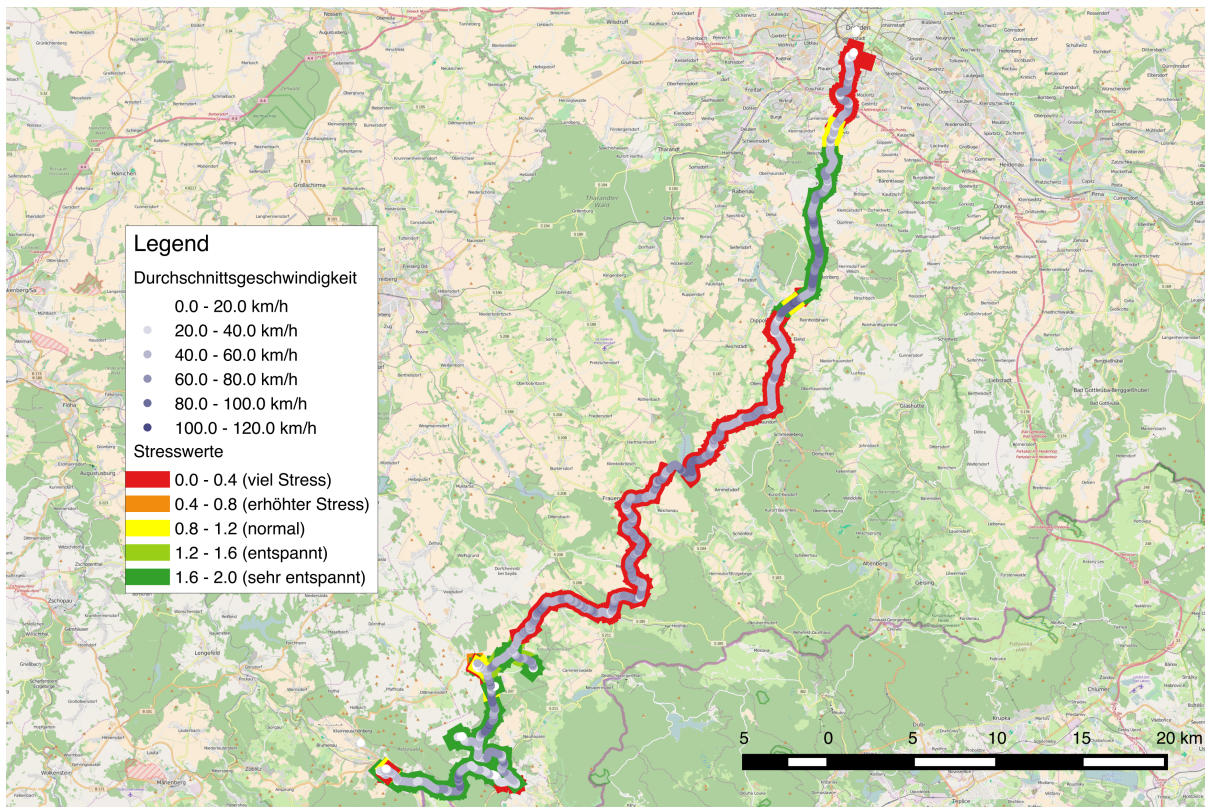


Abbildung 7.2: Geschwindigkeiten und Bewertungen der Fahrt 1.

bewerten gewesen wären. Um die einzelnen Abschnitte differenzierter bewerten zu können, müsste die Durchführung der Evaluierung häufiger erfolgen. Hierfür müssten die Bewertungen allerdings während der Fahrt vorgenommen werden. Dies ist bei manueller Ausführung aus sicherheitstechnischer Sicht nicht empfehlenswert, da durch visuell-manuelle Nebentätigkeiten während der Fahrt die Reaktionsgeschwindigkeit abnimmt und ein erhöhtes Risiko für Unfälle besteht [KAP11]. Zudem ist die gesetzliche Lage zur Bedienung von mobilen Geräten während der Fahrt nicht eindeutig. In §23 Absatz 1a der deutschen Straßenverkehrs-Ordnung¹ wird die Nutzung von Mobiltelefonen untersagt, wenn dafür das Gerät aufgenommen oder gehalten werden muss. Das Tippen auf das Smartphone in einer Halterung ist aber nicht ausdrücklich untersagt und die Bedienung des Evaluierungsdialogs somit rein rechtlich gesehen möglich. Prinzipiell sollte aber auf die händische Bedienung mobiler Geräte während der Fahrt verzichtet und eine andere Methode zur Interaktion gewählt werden.

¹Straßenverkehrs-Ordnung vom 6. März 2013 (BGBl. I S. 367), die durch Artikel 2 der Verordnung vom 15. September 2015 (BGBl. I S. 1573) geändert worden ist.

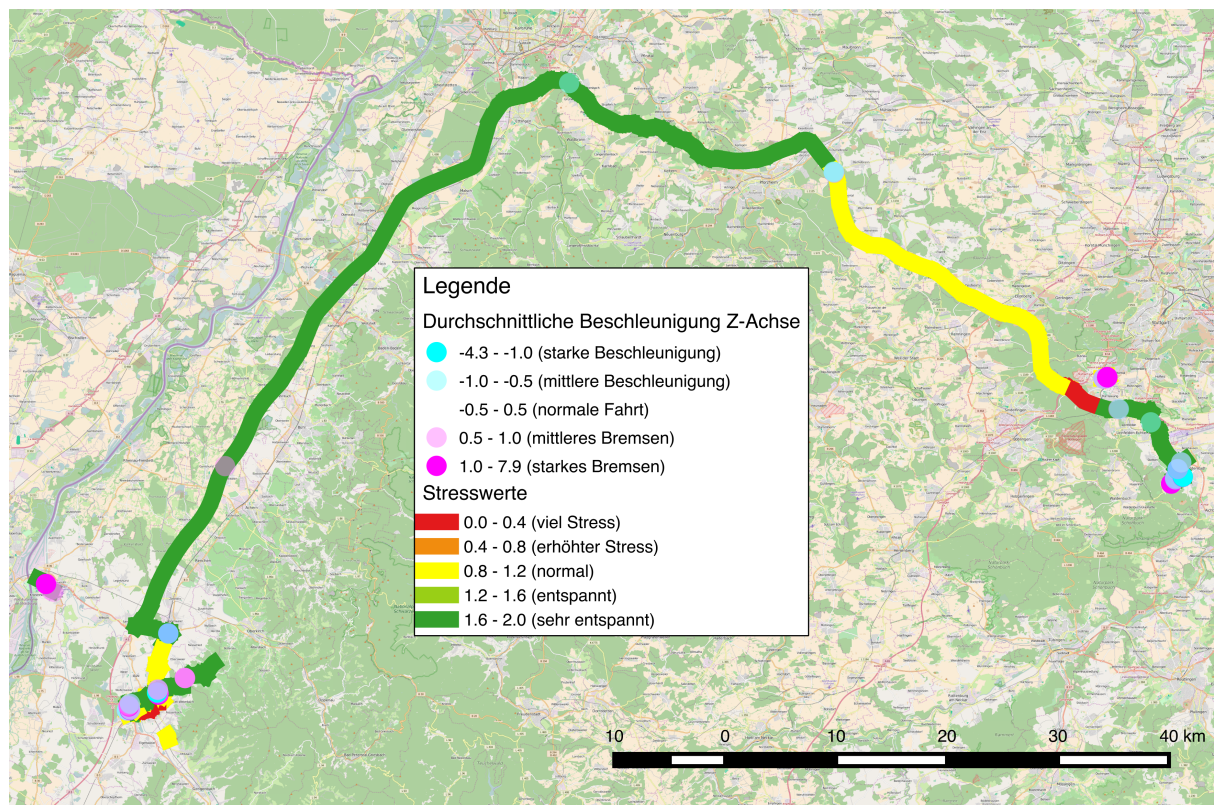


Abbildung 7.3: Beschleunigungswerte und Bewertungen der Fahrt 2.

Auf einer zweiten Fahrt in Abbildung 7.3 ist das genaue Gegenteil zur ersten Fahrt zu sehen. Hier ist der Fahrer auf den Bundesstraßen B170 und B171 unterwegs und seine Fahrt weist sehr häufiges Beschleunigen und Bremsen auf. Dementsprechend schlecht werden die Streckenabschnitte danach auch bewertet. Ein Blick auf Abbildung 7.4 zeigt, dass auch die Geschwindigkeit verhältnismäßig niedrig ist und einer hohen Fluktuation unterliegt.

7.3 Automatische Stresswert-Bestimmung

Anhand der erfassten Sensordaten kann keine hinreichende Aussage darüber getroffen werden, ob alleine anhand der Beschleunigungswerte und Geschwindigkeiten der Stressfaktor eines Streckenabschnitts bestimmt werden kann. Die zur Verfügung stehenden und untersuchten Ergebnisse deuten aber darauf hin, dass dies möglich sein könnte. Um dies mit Gewissheit festlegen zu können, ist es notwendig, mehr Daten über einen längeren Zeitraum zu sammeln und auszuwerten. Auch das von Vhaduri et al. [VAS+14] beschriebene GStress Modell könnte

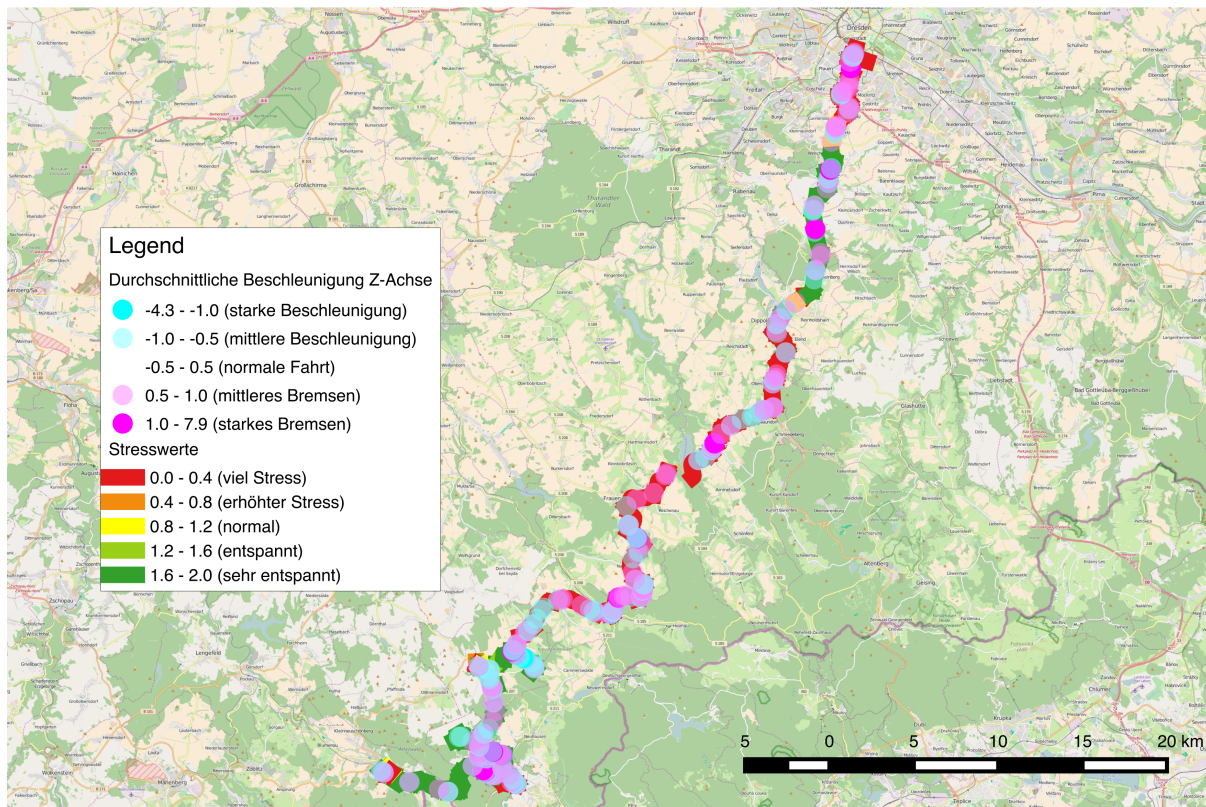


Abbildung 7.4: Geschwindigkeiten und Bewertungen der Fahrt 2.

für die automatische Bestimmung der Stresswerte herangezogen oder als zusätzliche Validierungsmethode eingesetzt werden. Mit den von ihnen vorgeschlagen Methoden zur Auswertung von GPS-Daten und der Bestimmung des Stressgehalts von verschiedenen Elementen der Strecke, können zusätzliche Einblicke gewonnen werden. Zudem sind für ihre Methode keine zusätzlichen Werkzeuge erforderlich, da hierfür die GPS-Daten des Smartphones ausreichen.

7.4 Stressreduzierte Routen

Bis zum Zeitpunkt der Auswertung waren lediglich vier stressreduzierte Routenplanungen durchgeführt worden, von welchen drei knapp einen Kilometer vor dem Ziel abgebrochen wurden. Der Grund für die geringe Anzahl der bisher gefundenen stressreduzierten Routen liegt sehr wahrscheinlich an der noch zu geringen Anzahl bewerteter Straßensegmente. Mit steigender Anzahl an Bewertungen sollten hier jedoch vermehrt solche Routen gefunden werden. Auf den verfügbaren Daten lassen sich somit keine konkreten Aussagen zu stressreduzierten Routen treffen.

8 Zusammenfassung und Ausblick

In diesem Kapitel werden wichtigsten Aspekte und Ergebnisse der vorliegenden Arbeit zusammengefasst und ein Ausblick auf mögliche weiterführende Arbeiten und Themen gegeben.

8.1 Zusammenfassung

Ziel dieser Diplomarbeit war die Untersuchung einer neuen Routenoption, welche den Fahrer entspannter und mit weniger Stress an sein Ziel bringt. Hierfür wurden in der Literaturrecherche verschiedene Algorithmen zur Routenberechnung in Graphen und Konzepte zur Stressbestimmung und Erlebniserfassung von Autofahrern betrachtet. Aufgrund der weiten Verbreitung des Betriebssystems Android bei mobilen Geräten, wurde die Entwicklung einer Navigation-App für Android umgesetzt. Als Grundlage für diese App wurde OsmAnd gewählt, zum einen weil der Quellcode frei verfügbar ist, zum anderen weil es von allen betrachteten quelloffenen Navigation-Apps die umfangreichste, fortschrittlichste und am weitesten verbreitete Anwendung ist. Ausgehend von der Basisversion wurde ein Plugin entwickelt, über welches der Benutzer Eingaben zu Stresswerten für befahrene Streckenabschnitte vornehmen kann. Die Eingabe konnte hierbei manuell oder per Sprache getätigt werden. Des Weiteren wurden verschiedene sensorische Daten der Geräte gespeichert, um einen besseren Überblick über die äußeren Umstände während der Fahrt zu gewinnen. Alle auf diese Weise generierten Daten werden an einen zentralen Server gesendet und dort in einer Datenbank gespeichert. Aus dieser wird in regelmäßigen Abständen eine separate Datenbank generiert, welche die durchschnittlichen Bewertungen aller erfassten Straßensegmente enthält. Diese wird wiederum von den mobilen Geräten heruntergeladen und die in der Datenbank gespeicherten Werte in die Berechnung der Route miteinbezogen. Der Benutzer kann somit aktiv an der Verbesserung der Suche nach stressfreien und entspannten Routen teilnehmen. Zudem können andere Teilnehmer von den bisherigen Bewertungen profitieren und gleichzeitig selbst eventuelle alternative Routen bewerten.

Die Ergebnisse zeigen eine steigende Tendenz bei der Nutzung der neuen Routenoption. Mithilfe der sensorischen Daten lässt sich erkennen, dass es einen möglichen Zusammenhang zwischen den Beschleunigungswerten und der Geschwindigkeit im Bezug auf die Bewertung der Streckensegmente gibt. Bei häufigen Fluktuationen in diesen wurden die Streckensegmente tendenziell schlechter bewertet als bei ruhiger und gleichmäßiger Fahrt. Um diese

Tendenzen bestätigen zu können müssen mehr Daten von den Benutzern und Geräten gesammelt und ausgewertet werden. Damit könnte es in Zukunft möglich sein, vollautomatisch bestimmen zu können, wie viel Stress eine Route beim Fahrer auslöst und die Routenführung dementsprechend anzupassen.

8.2 Weiterführende Arbeiten und Themen

Für weitere Arbeiten in diesem Themengebiet gibt es noch einige Möglichkeiten. Ein großes Problem stellt die Laufzeit für das Berechnen der verschiedenen Routenoptionen dar. Hier könnte mithilfe von Parallelisierung die Berechnungszeit drastisch verkürzt werden. Durch kürzere Berechnungszeiten wäre es dann auch möglich, mehrere stressreduzierte Routen zu berechnen um dem Benutzer eine größere Auswahl an Routen zu präsentieren. Auch bei der grundlegenden Berechnung der Route gibt es noch Optimierungspotenzial. Zum einen werden durch den Standardalgorithmus von OsmAnd teilweise immer noch nicht die optimalen Routen gefunden, zum anderen könnten auch für die stressreduzierten Routen diverse Schranken für den maximalen Umweg oder den maximalen Zeitunterschied zwischen der normalen und der stressreduzierten Route implementiert werden. So kann dem Benutzer mehr Entscheidungsgewalt bei der Berechnung und Wahl der Route eingeräumt und die Routenberechnung individuell an den jeweiligen Benutzer angepasst werden.

Durch mehr Daten könnte es auch möglich werden die Stresswerte auf die Tageszeiten anzuwenden um herauszufinden, ob die Bewertung für ein Streckensegment am Morgen eine andere ist als am Mittag oder Abend. Hierdurch wäre es möglich die Stressreduzierung weiter zu verbessern und zu verfeinern. Als unterstützendes Element könnten zudem die Streckensegmente entsprechend ihrer aktuellen Bewertung eingefärbt werden, um dem Nutzer auch visuell zu verdeutlichen, warum eine andere Route vorgeschlagen wird als die normalerweise kalkulierte Route. Zudem könnten noch weitere Sensoren für die Auswertung hinzugenommen werden. Mögliche Sensoren wären hierbei der Helligkeitssensor und das Mikrofon des Smartphones oder im Auto verbaute Sensoren, welche über eine OBD Schnittstelle ausgelesen werden können.

In Happy Navi (OsmAnd) wurde mit PocketSphinx ein erster Versuch unternommen eine Offline-Spracherkennung mit in die App aufzunehmen. Auf diesem Gebiet gibt es noch sehr viel Verbesserungspotential. Android unterstützt die Offline-Spracherkennung nicht auf allen Geräten und Version. Zudem müssen die Sprachpakete auf kompatiblen Geräten manuell heruntergeladen und installiert werden, was zusätzliche Arbeit und Wissen des Nutzers voraussetzt. Daher wäre eine direkt in Happy Navi (OsmAnd) integrierte Lösung wünschenswert, zumal aus Gründen der Verkehrssicherheit die Bedienung per Sprache eine geeignete Alternative zur manuellen Eingabe durch Berührung des Displays darstellt.

Verbesserungspotenzial gibt es auch bei der Handhabung von Bewertungen für Autobahn- und Bundesstraßenabschnitte. Insbesondere sollte hier die Frequenz der Stresswerteingaben

erhöht werden, um die gleichzeitige Bewertung von sehr vielen Streckensegmenten auf einmal zu verhindern. Eine Möglichkeit wäre hierfür die Spracheingabe zu verbessern und in regelmäßigen Abständen eine Bewertung zu forcieren. Eine andere Möglichkeit könnte sein, die durch Beschleunigungs- und GPS-Sensoren erfassten Werte für eine automatische Bewertung heranzuziehen. Mit der zunehmenden Sensorik in modernen Automobilen könnten sich hier in Zukunft zudem noch mehr Werte gewinnen und auswerten lassen.

Der letzte zu nennende Punkt sind die Umstände durch das gleichzeitige Bereitstellen von zwei verschiedenen Version, einer freien für den Google Play Store und einer Vollversion, welche über einen anderen Weg verteilt wird. Während Geräte mit der freien Version automatisch ein Update der App durchführen können, müssen die Nutzer der Vollversion diese bisher manuell herunterladen und installieren. Hier wäre eine einfachere Ein-Klick-Lösung wünschenswert, welche es auch dem unerfahrenen Benutzer ermöglicht die App einfach auf dem aktuellen Stand zu halten.

Literaturverzeichnis

- [AGB13] M. Akamatsu, P. Green, K. Bengler. „Automotive technology and human factors research: Past, present, and future“. In: *International journal of vehicular technology* 2013 (2013) (zitiert auf S. 15).
- [And15] Android. *Compatibility Test Suite*. Dez. 2015. URL: <https://source.android.com/compatibility/cts/index.html> (besucht am 15. 12. 2015) (zitiert auf S. 32).
- [Atz13] U. Atz. „Evaluating Experience Sampling of Stress in a Single-subject Research Design“. In: *Personal Ubiquitous Comput.* 17.4 (Apr. 2013), S. 639–652 (zitiert auf S. 22).
- [Bah11] D. Bahrtdt. „Effiziente Textsuche in OpenStreetMap-Daten auf mobilen Geräten“. Studienarbeit. Universität Stuttgart, Nov. 2011 (zitiert auf S. 29).
- [BBZV12] K. Boriboonsomsin, M. Barth, W. Zhu, A. Vu. „Eco-Routing Navigation System Based on Multisource Historical and Real-Time Traffic Information“. In: *Intelligent Transportation Systems, IEEE Transactions on* 13.4 (Dez. 2012), S. 1694–1704 (zitiert auf S. 21).
- [BDH+14] M. Baum, J. Dibbelt, L. Hübschle-Schneider, T. Pajor, D. Wagner. „Speed-Consumption Tradeoff for Electric Vehicle Route Planning“. In: *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*. 2014, S. 138 (zitiert auf S. 21).
- [BM04] J. E. Bell, P. R. McMullen. „Ant colony optimization techniques for the vehicle routing problem“. In: *Advanced Engineering Informatics* 18.1 (2004), S. 41–48. URL: <http://www.sciencedirect.com/science/article/pii/S1474034604000060> (zitiert auf S. 18).
- [Büh13] S. Bühler. „Onboard Routenplanung auf dem Smartphone“. Diplomarbeit. Universität Stuttgart, Aug. 2013 (zitiert auf S. 16).
- [Dij59] E. W. Dijkstra. „A note on two problems in connexion with graphs“. In: *Numerische mathematik* 1.1 (1959), S. 269–271 (zitiert auf S. 16).
- [DMC91] M. Dorigo, V. Maniezzo, A. Coloni. *The ant system: An autocatalytic optimizing process*. Techn. Ber. Technical report, 1991 (zitiert auf S. 18).
- [GMG+89] E. Gulian, G. Matthews, A. I. Glendon, D. R. Davies, L. M. Debney. „Dimensions of driver stress“. In: *Ergonomics* 32.6 (1989), S. 585–602. eprint: <http://dx.doi.org/10.1080/00140138908966134> (zitiert auf S. 14, 23).

- [Hei12] F. Heinrich. „Vorhersage der Fahrerbelastung während der Fahrt“. Diplomarbeit. Universität Stuttgart, Sep. 2012 (zitiert auf S. 37).
- [HNR68] P. Hart, N. Nilsson, B. Raphael. „A Formal Basis for the Heuristic Determination of Minimum Cost Paths“. In: *Systems Science and Cybernetics, IEEE Transactions on* 4.2 (Juli 1968), S. 100–107 (zitiert auf S. 17).
- [HP05] J. Healey, R. Picard. „Detecting stress during real-world driving tasks using physiological sensors“. In: *Intelligent Transportation Systems, IEEE Transactions on* 6.2 (Juni 2005), S. 156–166 (zitiert auf S. 37).
- [HW99] D. A. Hennessy, D. L. Wiesenhal. „Traffic congestion, driver stress, and driver aggression“. In: *Aggressive Behavior* 25.6 (1999), S. 409–423. URL: [http://dx.doi.org/10.1002/\(SICI\)1098-2337\(1999\)25:6%3C409::AID-AB2%3E3.0.CO;2-0](http://dx.doi.org/10.1002/(SICI)1098-2337(1999)25:6%3C409::AID-AB2%3E3.0.CO;2-0) (zitiert auf S. 13, 23).
- [KAP11] K. Kircher, C. Ahlström, C. Patten. „Mobile telephones and other communication devices and their impact on traffic safety“. In: *VTI Report A 729* (2011) (zitiert auf S. 61).
- [MHB12] S. Montealegre, R. Haase, R. Boufelliga. *Vehicle navigation system with route determination based on a measure of sportiness*. US Patent 8,285,485. Okt. 2012. URL: <http://www.google.com/patents/US8285485> (zitiert auf S. 20).
- [MR14] M. Manseer, A. Riener. „Evaluation of Driver Stress While Transiting Road Tunnels“. In: *Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI ’14. Seattle, WA, USA: ACM, 2014, S. 1–6 (zitiert auf S. 14, 23).
- [MTD+13] A. Meschtscherjakov, S. Trösterer, C. Döttlinger, D. Wilfinger, M. Tscheligi. „Computerized Experience Sampling in the Car: Issues and Challenges“. In: *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI ’13. Eindhoven, Netherlands: ACM, 2013, S. 220–223 (zitiert auf S. 22).
- [POP+05] C. Puri, L. Olson, I. Pavlidis, J. Levine, J. Starren. „StressCam: Non-contact Measurement of Users’ Emotional States Through Thermal Imaging“. In: *CHI ’05 Extended Abstracts on Human Factors in Computing Systems*. CHI EA ’05. Portland, OR, USA: ACM, 2005, S. 1725–1728 (zitiert auf S. 37).
- [PSMT14] B. Pfleging, S. Schneegass, A. Meschtscherjakov, M. Tscheligi. „Experience Maps: Experience-Enhanced Routes for Car Navigation“. In: *Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI ’14. Seattle, WA, USA: ACM, 2014, S. 1–6 (zitiert auf S. 15, 21, 23).

- [SKNI13] W. Sha, D. Kwak, B. Nath, L. Iftode. „Social Vehicle Navigation: Integrating Shared Driving Experience into Vehicle Navigation“. In: *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*. HotMobile '13. Jekyll Island, Georgia: ACM, 2013, 16:1–16:6. URL: <http://doi.acm.org/10.1145/2444776.2444798> (zitiert auf S. 38).
- [SPB+13] S. Schneegass, B. Pfleging, N. Broy, F. Heinrich, A. Schmidt. „A Data Set of Real World Driving to Assess Driver Workload“. In: *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI '13. Eindhoven, Netherlands: ACM, 2013, S. 150–157 (zitiert auf S. 23).
- [TKG15] P. Tejaswin, R. Kumar, S. Gupta. „Tweeting Traffic: Analyzing Twitter for Generating Real-time City Traffic Insights and Predictions“. In: *Proceedings of the 2Nd IKDD Conference on Data Sciences*. CODS-IKDD '15. Bangalore, India: ACM, 2015, 9:1–9:4. URL: <http://doi.acm.org/10.1145/2778865.2778874> (zitiert auf S. 38).
- [Tom10] TomTom. „White paper: How TomTom's HD Traffic and IQ Routes data provides the very best routing – Travel Time Measurements using GSM and GPS Probe Data“. In: (2010) (zitiert auf S. 21).
- [VAS+14] S. Vhaduri, A. Ali, M. Sharmin, K. Hovsepian, S. Kumar. „Estimating Drivers' Stress from GPS Traces“. In: *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI '14. Seattle, WA, USA: ACM, 2014, 20:1–20:8 (zitiert auf S. 37, 62).
- [WGS10] C. Wagner, I. S. Groß, I. S. Steinbrecher. „Ein Bewertungssystem für OpenStreet-Map“. Diplomarbeit. Technische Universität Dresden, Dez. 2010 (zitiert auf S. 26).
- [WMB+13] D. Wilfinger, M. Murer, A. Baumgartner, C. Döttlinger, A. Meschtscherjakov, M. Tscheligi. „The Car Data Toolkit: Smartphone Supported Automotive HCI Research“. In: *Proceedings of the 5th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*. AutomotiveUI '13. Eindhoven, Netherlands: ACM, 2013, S. 168–175 (zitiert auf S. 22, 23).
- [YCC15] J. Yang, L. Chou, Y. Chang. „Electric Vehicle Navigation System Based on Power Consumption“. In: *Vehicular Technology, IEEE Transactions on* PP.99 (2015), S. 1–1 (zitiert auf S. 21).

Alle URLs wurden zuletzt am 04. 02. 2016 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift