Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projekt 2

Christian Wagner

Annotationsmöglichkeiten bei Städetouren

Fakultät Technik und Informatik
Studiendepartment Informatik

Inhaltsverzeichnis

1 Einleitung

1.1 Vision

Die Vision für dieses Projekt wird in dem folgenden Szenario erläutert.

Szenario
Ein Tourist besucht zum ersten Mal die HafenCity in Hamburg. Beim Rundgang durch die Straßen bleibt er vor einem großen Neubau stehen. Ihn interessiert für welchen Verwendungszweck dieser Gebäudekomplex gebaut wird.

Er holt sein Mobiltelefon heraus und wählt die Anwendung „Tourand“ aus. Diese zeigt ihm eine Karte mit seinem Standort und allen im Umkreis befindlich interessanten Punkten (POIs). Er wählt auf der Karte den Punkt vor sich aus, wodurch die Anwendung ihm zeigt, dass es sich um das zukünftige Bürogebäude des Spiegel-Verlages handelt und fragt, ob noch weitere Informationen gewünscht sind. Dieses bestätigt der Benutzer, woraufhin ihm Texte zum Lesen über Spiegel angezeigt werden. Nach kurzem Durchstöbern der Texte ist ihm das Lesen auf dem kleinen Display zu aufwendig. Durch Bestätigen einer Taste werden die eben angezeigten Texte nun als Sprachausgabe vorgelesen und einige Bilder dazu angezeigt.

Der Tourist ist von den Artikeln so begeistert, dass er beschließt etwas Zeit in die Besichtigung der HafenCity zu investieren. Dafür gibt er in die Anwendung nur den Zielpunkt und die Dauer an. Es besteht außerdem die Möglichkeit noch besondere Interessen (z.B. Architektur, Geschichte etc.) für die Tour einzugeben. Zusätzlich zu den bereits bestehenden Interessen des Benutzerprofile werden diese Informationen für die Planung verwendet werden.

Nach kurzer Berechnung kann der Tourist die Tour starten. Entlang des Kaiserkai werden ihm automatisch die verschiedenen, teilweise noch entstehenden Gebäude erklärt. Am Ende der Straße angekommen werden ihm automatisch Informationen zur Elbphilharmonie vorgelesen. Dieses Gebäude beeindruckt den Touristen; durch den längeren Standortaufenthalt werden noch zusätzlich Informationen angeboten. Der Tourist beschließt sich eine kleine Videosequence anzusehen. Die Benutzerinteressen sind besonders auf historische Architektur ausgelegt. Daher werden ihm Videos zum Kaispeicher A angezeigt, da dieser bis zum 2.Weltkrieg an dem Standort der Elbphilharmonie stand. Durch den verlängerten Aufenhalt kann die Zeitdauer mit der planten Tour nicht eingehalten werden. Daraufhin berechnet Tourand den Tourverlauf neu und entfernt einige nicht so interessante POIs.
Während des Rundgangs beschließt der Tourist noch einen Zwischenstop in einem Restaurant einzulegen. Dafür gibt er in die Anwendung die Essenskategorie „Italenisch“ und die geplante Uhrzeit ein. Daraufhin berechnet Tourand die Tour erneut, um den Zwischenstop

für die Mahlzeit miteinzuplanen.
Pünktlich mit einem Zwischenstop in einem Italenischen Restaurant erreicht der Tourist am Ende seiner Tour den Zielpunkt.

1.2 Ziel des Projektes

Das Ziel des Projektes ist die Analyse der Umsetzungsmöglichkeiten für die vorgestellte Vision (s. Kap. 1.1). Dabei stehen die geeigneten Darstellungsmöglichkeiten und Benutzeraktionen im Vordergrund. Mögliche Ansätze sollen in einer Anwendung prototypisch implementiert und getestet werden.

Dabei geht es hauptsächlich um die Kartendarstellung mit den dazugehörigen POIs auf einem mobilen Client. Zusätzlich sollen die verschiedenen Benutzerinteraktionen bei der Karten- und POI-Bedienung evaluiert werden.

Desweiteren geht es um die automatische Ankunftserkennung an beliebigen Orten, um an diesen ggf. automatisch Informationen anzuzeigen.

Die Tourplanung soll für die Entwicklung der Anwendung mit vorgesehen werden. Eine komplette Umsetzung wird aber nicht realisiert.

2 Architektur und Entwicklungsumgebung

Die Architektur und der Aufbau der Entwicklungsumgebung wurde bereits im Projekt I-Bericht [[12]] diskutiert. Auch der Infrakstrukturaufbau mit den verschiedenen Komponenten wurde in diesem Bericht beschrieben.

2.1 Google Android

Für den mobilen Client wurde die Android-Plattform von Google als Entwicklungsumgebung gewählt. Viele Hersteller entwickeln bereits Mobiltelefone, die auf dem Android-Betriebssystem basieren.

Durch die für Android-Devices vorgegeben Touch-Bedingung ist eine benutzerfreundliche Steuerung der Anwendung auf den in der Regel kleinen Displays gut realisierbar.
Zusätzlich bietet Android eine intergrierte GoogleMaps API an. Dadurch ist eine individuelle Kartendarstellung leicht umsetztbar.

2.2 GoogleMaps API

Die intergrierte GoogleMaps API in Android ermöglicht die einfache Anbindung einer Anwendung an die GoogleMaps-Funktionen. So können Kartendarstellungen, Zoomfunktionen, Wechsel der Kartendarstellung zwischen Karteansicht(Stadtplan) und Satelittenansicht und die Berechnung zwischen den Display- zu den Geo-Koordinaten einfach verwendet werden.

2.3 Server

Dienste, deren Ausführung auf den Android-Devices nicht möglich sind, werden von einem zentralen Server implementiert werden. Der Austausch der Informationen zwischen dem Client und dem Server wird über Webservices realisert. Dafür wurde ein Tomcat-Webserver zur Implementierung der Webservices für den Server ausgewählt. Bisher wurden auf dem Server folgende Dienste umgesetzt:

- **Text-To-Speech-Umwandlung:** Für die Umwandlung von Texten in Audio-Datei (s. Kap. 3.4) wurde die Anwendung auf dem Server installiert. So können verschiedene Texte an den Server übertragen werden, der dann dem Client eine Audio-Datei zur Verfügung stellt, indem der übertragende Text in Sprache umgewandelt worden ist.
- **Routenberechnung:** Für die Darstellung der Tour übernimmt der Server die Aufgabe über GoogleMaps die Routen zwischen den einzelnen Tourpunkten zu errechnen. Dafür ermittelt der Server die benötigten Punkte zur Darstellung der Tour und überträgt dem Client diese Linienpunkte (s. Kap. 3.3), der sie als Tourlinie darstellt.
- **Tourplanung:** Die Tourplanung wurde bisher nur testweise umgesetzt. (s. Kap. 5. In der Fortsetzung des Projektes ist vorgesehen, dass der Server die komplette Tourplanung übernimmt und dem Client die optimale Tour für den Benutzer mitteilt.

2.4 Komponenten

Das folgende Komponenten-Diagramm (s. Abb. 1) zeigt einen geplanten Aufbau für eine komplette Realisierung der Vision (s. Kap. 1.1).

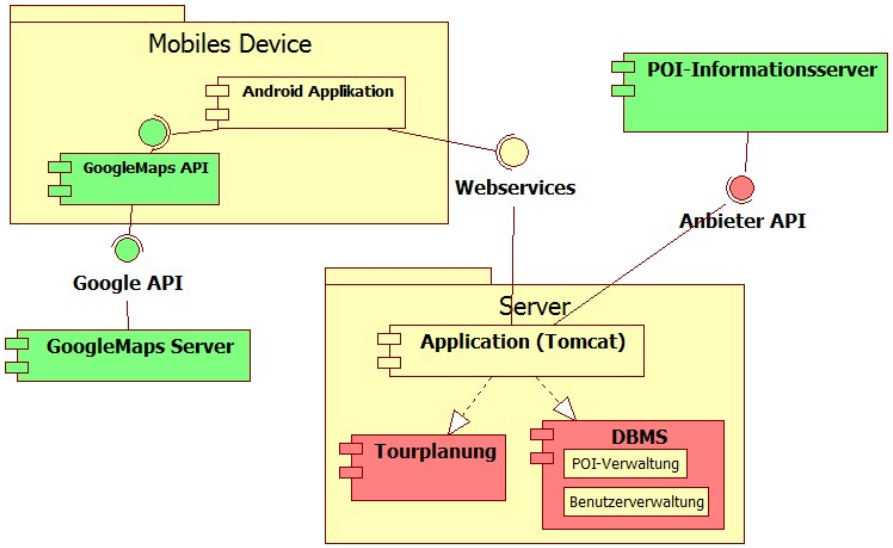

Abbildung 1: Komponentendiagramm

Die rotmarkierten Komponenten wurden während des Projekts nur für zwischenzeitliche Tests teilweise implementiert. Externe verwendete Komponenten sind in der Grafik grün markiert.

3 Darstellungen

3.1 POI-Darstellung

Für die Darstellung der verschiedenen POIs wird die Klasse „Overlay“ der Android-GoogleMaps-Bibliothek verwendet werden. Diese werden in „ItemizedOverlay“-Klassen zusammengefasst und auf der Karte dargestellt. Bei den Overlays können Geo-Koordinaten angegeben werden, wodurch eine Berechnung mit Display-Koordinaten und Zoomfaktor nicht notwendig ist. Auch bei Veränderung des Zooms oder Verschiebung der Karte werden die Overlay-Objekte automatisch mitangepasst.
Zur besseren Darstellung der POIs wird allerdings die Größe der Overlays bei Änderung des Zooms verkleinert bzw. vergrößert. Dafür können die ItemizedOverlay verwendet werden.

Hier reicht es der übergeordneten ItemizedOverlay-Klasse eine Grafik zuzuordnen und alle dazugehörigen Overlays werden mit dieser Grafik angezeigt (s. Abb. 2).

Abbildung 2: Overlay als POI-Anzeige

3.2 Darstellung der POI-Information

Nach Erreichen oder Auswählen eines POIs wird ein Context-Menü angezeigt, bei dem der Benutzer verschiedene Informationsmöglichkeiten bzw. Darstellungstypen auswählen kann. Für das Projekt wurden die Möglichkeiten des lesbaren Textes, des vorlesbaren Textes, einer SlideShow mit oder ohne Audio-File und das Abspielen eines Videos vorgesehen. (s.Abb. 3)

Abbildung 3: Wahl der Informationsanzeige

Die für die Information verwendeten Daten liegen nicht lokal auf dem mobilen Device, sondern nur die Datei-Quellen werden als Link bei den POI-Information zusätzlich gespeichert. Bei Abfrage der Informationen werden die von den verschiedenen Server heruntergeladen und angezeigt.

Bei Audio oder Video-Dateien wird eine Streaming-Klasse verwendet, die es ermöglicht die Dateien bereits vor dem kompletten herunterladen abzuspielen.

Bei einer Slideshow werden die Bilder einzeln geladen. Es wird jeweils das Bild angezeigt, das als letztes komplett heruntergeladen wurde. Der entstehende Zeitverzug beim Laden der Bilder dient als Pause der SlideShow zwischen den Bildern. Wenn alle angeforderten Bilder geladen sind, kann der Benutzer zwischen den verschiedenen Bildern wechseln (s. Abb. 3). Bei der Wahl einer Slideshow mit Audiodatei wird zusätzlich die Streaming-Klasse verwendet, um die hinterlegte Audiodatei zu den Bildern wiederzugeben.

3.3 Darstellung der Tour

Für die Darstellung der Tour wurden ebenfalls die Overlay- und OverlayItem-Klassen verwendet. Zur Darstellung einer Tour werden die einzelnen Routenpunkte benötigt, die beim hintereinander Verbinden durch Linien die Tour ergeben. Diese Punkte können durch verschiedene Routenplaner ermittelt werden, die in der Regel GPX- oder KML-Dateien verwenden. Beide Formate enthalten die benötigten Informationen zur Darstellung und können somit verwendet werden. Die Ermittlung der Route zwischen den Tourpunkten übernimmt der Server, da dieser in der Regel über eine schneller Internetverbindung verfügt. Der Server extrahiert aus den Dateien die Punkte zur Darstellung der Tour und übertragt sie dem Client. Die einzelnen Linien bestehend aus zwei Punkten werden in einem Overlay-Objekt „RouteLine“ gespeichert. Die Anzeige aller RouteLine-Objekte ergeben dann die Tour. (s. Abb. 4)

Abbildung 4: Darstellung der Tour

Die Tourlinie wird zusätzlich mit einem Farbverlauf angezeigt. Das hilft insbesondere bei Überschneidung der Linie, den Richtungsverlauf der Tour nachvollziehen zu können.

3.4 Text-To-Speech

Um die lesbaren Informationen in hörbare Audio-Datei umzuwandeln, benötigt man eine Text-To-Speech-Applikation. Die bereits im vorherigen Projekt [12] getestete OpenMary-Applikation wurde auf einem Server umgesetzt. An diesen Server können Texte übertragen werden und eine zur Sprache übersetzte Audio-Datei oder als Audio-Stream heruntergeladen werden. Über Sprachpakete können verschiedenen Sprachen sowie verschiedenen Sprachsynthesen verwendet werden.
Die Qualität des Ergebnisses ist nicht überragend aber akzeptierbar.

4 Ankunft am POI

4.1 Automatische Ankunftserkennung

POIs sind in der Regel nur mit einer bestimmten Koordinate in den verschiedenen Systemen hinterlegt. Um hier eine automatische Ankunftserkennung implementieren zu können, müsste der Benutzer die Koordinate exakt erreichen. Das Problem ist nicht nur, dass es nicht praktikabel ist, sondern dass in vielen Fällen die Koordinate direkt auf Gebäuden plaziert ist. In den Gebäuden funktioniert allerdings kein GPS-Signal, wodurch es in diesen Fällen sogar unmöglich ist diese Koordinate genau zu erreichen.

Zur Lösung dieses Problems wurde als erste Möglichkeit ein Entfernungsradius implementiert. Mit Hilfe dieses Entfernungsradius kann ein Umkreis um das POI bestimmt werden (s. Abb. 5). Befindet sich die Benutzerkoordinate im Umkreis des POIs, werden die Informationen automatisch angezeigt.
Das Problem an diesem Ansatz ist, dass es POIs gibt, die beispielsweise nur von einer bestimmten Seite als besucht angesehen werden können wie z.B. das St.Pauli Theater. Erreicht man diesen Punkt von der anderen Seite, ist schwer ersichtlich, dass es sich um das gewünschte Besuchsziel handelt und hier sollten die Informationen noch nicht angezeigt werden. Zusätzlich gibt es auch POIs wie die Hamburger Alster, welche eine große Erreichbarkeitsfläche besitzen. Hier wäre der Koordinatenpunkt und der dazugehörige Umkreis schwer zu bestimmen.

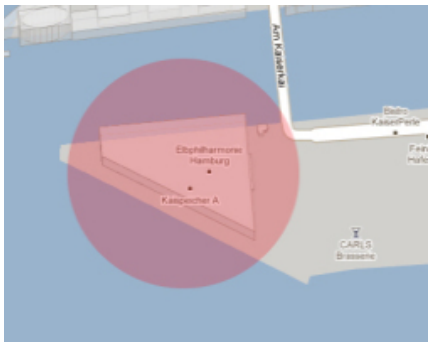

Abbildung 5: Umkreisbestimmung

Abbildung 6: Figuren und freie Flächen

Eine weitere Möglichkeit wäre die Implementierung verschiedener geometrischer Figuren oder freien Flächen. Dadurch könnte der Benutzer genau die Flächen bestimmen, die als Ankunft an einem POI verwendet werden sollen (s. Abb. 6). Dafür ist allerdings die Eingabe der Flächen und die Berechnung, ob sich der Benutzer in dieser Fläche aufhält, aufwendiger. Für die Applikation wurde dafür folgendes Interface implementiert:

```
public interface IPoiArea {
    public boolean isInArea(GeoPoint gp);
    public double distanceToArea(GeoPoint gp);
}
```

Dieses Interface wird für die Standort-Eingabe von POIs verwendet. So können die möglichen Flächen um beliebig PoiArea-Klassen erweitert werden. Diese erweiterten Klassen müssen nur zwei Methoden, die Entfernung einer Koordinate (distanceToArea()) und ob sich eine Koordinate in der Fläche befindet (isInArea()), implementieren. Die Anwendung kann somit die POIs befragen, ob sich der Benutzer mit seiner Koordinate schon im Bereich des POIs aufhält und wie weit er noch davon entfernt ist.

Alle Schwierigkeiten können mit dieser Methoden nicht gelöst werden, da auch der Unterschied zwischen dem Erreichen und dem Erkennen eines Punktes berücksichtigt werden müssen. Beispielsweise ist der Hamburger Michel in vielen Richtungen aus einer weiten Entfernung bereits erkennbar. Sollte dem Benutzer eine weitere Entfernung ausreichen, um ihn als besucht zu akzeptieren, wäre die Hinterlegung aller Flächen, aus denen man ein POI einsehen könnte, notwendig. Der Aufwand für diese Eingaben wäre aber sehr schwer realisierbar. Daher sollte dem Anwender zusätzlich die Möglichkeit gegeben werden auch manuell Punkte auszuwählen und Informationen hierzu abzufragen.
Zusätzlich sollte vermieden verwerden, dass sich POI-Flächen überschneiden, da die Anwendung sonst bestimmen müsste, welche Informationen angezeigt werden. Andere Informationen könnten dadurch verloren gehen.

4.2 Manuelle Auswahl durch Touch-Event

Die verfügbaren POIs werden durch Overlays auf der Karte angezeigt. Das Auswählen eines POIs erfolgt durch einfaches Tippen auf den jeweiligen Punkt. Leider erfolgt kein einzelnes Event für den ausgewählten Punkt. Somit muss die onTouchEvent-Methode der übergeordneten Map-Activity den ausgewählten Punkt ermitteln. Dafür wird die GeoKoordinate des Touch-Punktes bestimmt und der nächstliegende POI dazu gestimmt.
Um unterscheiden zu können, ob wirklich ein POI gemeint wurden ist, muss eine maximal erlaubte Entfernung von Touch-Punkt zum POI bestimmt werden (s. Abb. 7). Bei der maximalen Entfernungsbestimmung muss der aktuelle Zoom beachtet werden, da bei höherem Zoomfaktor die maximale Entfernung abnehmen sollte. Die Methode zur Berechnung des nächsten POIs ist im Folgenden beschrieben:

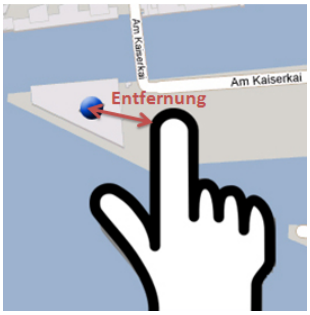

Abbildung 7: Touch-Punkt-Entfernung

Java-Code zur Berechnung des am nächstliegendem Overlay

```
// p - Geo-Koordinaten des TouchEvents
double entf; // Entfernung
for (Overlay o : overlayList) {
   if (distanceOf(o.getPoint(), p) < entf) {
      entf = distanceOf(oi.getPoint(), p);
      if (isDistInFactor(getZoom(), entf)) {
         nearestTabPoint = o.getPoint();
      }
   }
}
```

5 Tourplanung

Die Problematik der Tourenplanung wird in der Informatik in den Bereichen „Travelling Salesman Problem“, kurz TSP und „Orienteering Problem“, kurz OP diskutiert.

5.1 Travelling Salesman Problem & Orienteering Problem

Bei der Lösung eines TSP geht ins besonders darum eine bestimmte Anzahl von Punkten in kürzester Reihenfolge zu besuchen. Der Hauptunterschied zwischen einem TSP und einem OP ist, dass bei einem OP ein Faktor, z.B. die Tourdauer, die Planung einschränkt und dadurch nicht zwingend alle Punkte besucht werden.
Das Hauptproblem beim Errechnen einer Tour ist, dass Verfahren zur exakten Berechnung mindestens eine Komplexität von $O(2^n)$ besitzen. Das führt bereits bei wenigen POIs zu einem hohen Rechenaufwand. Um den Rechenaufwand zu verringern werden heuristische Verfahren verwendet. Diese liefern zwar nicht unbedingt die exakte Lösung, aber können

durch die geringer Komplexität auch bei einer hohen Anzahl von POIs in einer annehmbaren Zeit Lösungen liefern.

Eine ausführliche Diskussion zur Tourplanung befindet sich in der Ausarbeitung Master Seminar WS09/10 „Ortsabhängige Annotation von Städtetouren am Beispiel Hafen City“ [[11]]. In der Ausarbeitung wird für eine Tourplanung die parallele Ausführung verschiedenen Lösungvefahren diskutiert, welches die Qualtität der Lösung optimieren soll.

5.2 Framework zum Verfahrentest von TSP & OP

Für diese Ausarbeitung wurde im Rahmen dieses Projekt ein Java-Framework zum Testen solcher Verfahren entwickelt. Hierbei können verschiedenen Verfahren implementiert und mit vorgegebenen Tourbedingungen ausgeführt werden.
Nach Ausführung können diese mit den bereits vorhandenen Verfahren verglichen und analysiert werden. Für die Analyse wurde zusätzlich eine grafische Darstellung implementiert, die es vereinfacht einen Überblick der Lösung zu erhalten (s. Abb. 8).

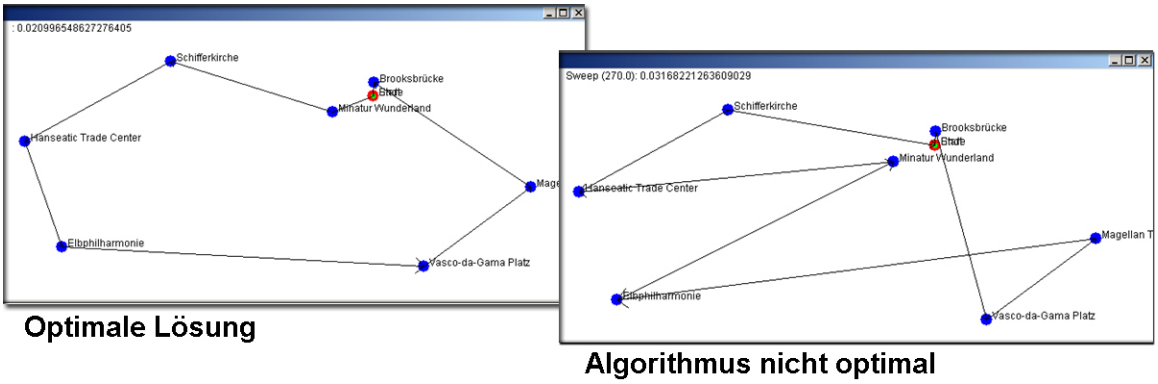

Abbildung 8: Framework - Verfahrensvergleich

6 Fazit

Das Android-Umgebung ist für die Darstellung der Karten und POI-Information gut geeignet. Durch die vorhandene GoogleMaps-API können zahlreiche Maps-Funktionen leicht und schnell implementiert werden. Die in Android enthalten Klassen wie Overlays ermöglichen eine einfache Darstellung der POI-Informationen. Die Touch-Bedienung des Android-System ermöglicht zusätzlich eine innovative Benutzerinteraktion.
Ein ausschließlich automatische Ankunftserkennung an einem POI kann nicht einfach realisert werden. Besonders die mögliche ausreichende Erkennbarkeit eines POIs aus weiter

Entfernung führten dazu, dass auch das manuelle Auswählen eines POI im Projekt umgesetzt wurde. Ein automatische Erkennung wurde für frei definierte Fläche realisiert.

Die Text-To-Speech-Applikation OpenMary eignet sich bestens zur Umwandlung der Texte in Sprache. Somit kann dem Benutzer überlassen werden, ob er die Texte lesen möchte oder ob diese als Audio-Datei vorgelesen werden. Die Qualität der Sprachergebnisse der Audio-Files ist für die englische Sprache gut; für die deutsche Sprache nicht optimal aber akzeptierbar.

Das komplexe Thema der Tourplanung wurde noch nicht vollständig umgesetzt, sondern nur die Umsetzungsmöglichkeiten durch einfache Tests ausprobiert. Bei der weiteren Umsetzung dient das Framework für Verfahrenstests als gut geeignetes Werkzeug für die Analyse der verschiedenen Tourplanungsverfahren.

6.1 Ausblick - Masterarbeit

In der anstehenden Masterarbeit wird die im Projekt erstellte Client-Anwendung „Tourand“ übernommen. Abgesehen von einigen wenigen Funktionen implementiert sie schon die meisten Anforderungen an den geplanten Client des Systems. Durch kleine Anpassungen kann die Anwendung in der Masterarbeit als Client-Anwendung verwendet werden.

Besonders das Framework zum Testen von TSP- und OP-Verfahren soll zum Einsatz kommen, da ein großer Teil der Masterarbeit sich mit der Umsetzung der Tourplanung beschäftigt. Hier sollen verschiedene Verfahren implementiert werden und parallel ein möglichst optimales Ergebnis errechnen. Zum Analysieren dieser Verfahren wird das erstellte Framework verwendet.

Weitere Informationen zu den Themenbereichen der Masterarbeit ist in der Seminar Ausarbeitung [11] zu finden.

Literatur

[1] Android Development Community - anddev.org
. http://www.anddev.org.

[2] Qype. http://www.qype.com.

[3] Tutorial - Audio Streaming. http://blog.pocketjourney.com/..
/android-streaming-mediaplayer-tutorial-updated-to-v1-5-cupcake/.

[4] Apache <Web Services /> Project. Web Services - Axis
. http://ws.apache.org/axis/.

[5] Thorsten Berka. Routen- und Tourenplanung
. http://www.ikg.uni-bonn.de.

[6] Ed Burnette. *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf, 2008.

[7] Google Inc. GoogleMaps
. http://maps.google.de.

[8] Google Inc. Offizielle Android-Developer Website
. http://developer.android.com.

[9] Projekt - OpenMary. OpenSource Text-To-Speech-Applikation
. http://mary.dfki.de/.

[10] The Apache Software Foundation. offizielle Apache Tomcat Website
. http://tomcat.apache.org/.

[11] Christian Wagner. Ortsabhängige Annotation von Städtetouren am Beispiel Hafen City
. http://users.informatik.haw-hamburg.de/~ubicomp.

[12] Christian Wagner. Projekt I
. http://users.informatik.haw-hamburg.de/~ubicomp.

[13] Wikipedia. Wikipedia - Travelling salesman problem
. http://en.wikipedia.org/wiki/Travelling_salesman_problem.