On-Line Navigation in GPS-Route

Heiner Ackermann, Mohamed Bettahi, René Brüntrup, Maik Drozdzynski, Stefan Edelkamp, Vanessa Faber, Andreas Gaubatz, Thomas Härtel, Seung-Jun Hong, Shahid Jabbar, Miguel Liebe, Tilman Mehler, Anne Scheidler, Björn Schulz und Feng Wang

Fachbereich Informatik
Baroper Straße 301
Universität Dortmund

Zusammenfassung In der Projektgruppe *GPS-Route* wird ein On-Line Navigationssystem konzipiert und implementiert, das aus einer Menge von GPS-Spuren eine mit Fahrtzeiten und Fahrtmöglichkeiten annotierte Karte generiert, diese vorverarbeitet, speichert und darin möglichst effizient nach kürzesten Wegen sucht. Die eigentlichen Daten werden sowohl mit einem GPS Empfänger als auch in einer Simulationsumgebung erzeugt. Projektziel ist eine Client/Server Architektur, in der statische als auch dynamische GPS-Spurinformationen in Datenbanken verwaltet und zur effizienten und mehrfachen kürzesten- und schnellsten-Wege Suche von einem PDA aus genutzt werden. Dabei werden verschiedene Beschleunigungstechniken eingesetzt. Als Grundlage der Simulation und der Visualisierung dienen topographische Karten des Landesvermessungsamt NRW, deren Pixeldarstellung halbautomatisch in einen Straßengraphen konvertiert werden. Durch geeignete Clusteralgorithmen werden die GPS-Spuren inkrementell zu einer geeigneten Anfragestruktur kombiniert.

1 Problembeschreibung

Der Ansatz zur Routenplanung mit dynamisch erzeugten Spurdaten eignet sich besser zur Navigation als ein statischer Graph des Wegeplanes. Aktuelle Informationen können mit Hilfe von GPS-Aufnahmegeräten bereitgestellt werden. Dabei wird der personelle und technische Aufwand zur elektronischen Kartographierung minimiert. Zudem sind kommerziell vertriebene elektronische Karten ungenau und die gespeicherte Information für ein *on-board* Navigationssystem durch stete Realwelt-Änderungen oft veraltet.

Der Vorteil einer an der Universität angesiedelten Entwicklung ist der gezielter Einsatz von aktuellen Algorithmen und Datenstrukturen. Zur Datenreduktion empfehlen sich z.B. Verfahren des *geometrischen Rundens*. Für die statistische Datenverbesserung auf Basis zusätzlicher Informationsquellen sind geeignete *Filter* angemessen. Für die reduzierten und gefilterten Daten bietet sich die *automatische Erkennung von Kreuzungen* an. Innerhalb des entstehenden Graphen werden mit verschiedenen *Explorationsalgorithmen* längen- oder zeiteffiziente Routen berechnet und dem Anwender fahrtbegleitend zur Verfügung gestellt. Anfragen werden mit effizienten Punktlokalisationsverfahren an den bestehenden Graphen angepasst. Zur effizienten Suche kann der Graph zudem auf alle Schnitt- und Endpunkte *reduziert* werden.

Die Implementierung einer Benutzeroberfläche auf einen PDA (z.B. einem Pocket PC mit GPRS Internetverbindung) stellt hohe Anforderungen an die Effizienz der Kartenvisualisierung und an die Minimierung der Kommunikation zum Server.

Topographische Karten, z.B. aus den der Landesvermessungsämtern, sind sehr genau vermessen und lassen mit mit GPS Informationen verknüpfen. Zur Vektorisierung der Karten, d.h. zur Straßenextraktion und Graphgenerierung, benötigt man verschiedene graphische *Skelettierungs-* und *Trackingalgorithmen*. Auf diesen Graphen kann man dann eine Verkehrssimulation durchführen. Solche Simulationsumgebungen dienen desweiteren zur *Protokollierung des Verkehrsflusses* und des *Fahrzeugverhaltens*.

Kurzum, das Thema birgt viele algorithmische Fragestellungen und ist für ein Informatik-Hauptstudiumsprojekt (12 Personen und einem Jahr Entwicklungszeit) geeignet. Im Folgenden fassen wir den Zwischenbericht des Projekts zusammen. Dabei fangen wir bei den Vorarbeiten und der zu Verfügung gestellten Infrastruktur an. Dann rekapitulieren wir die Themen des Einstiegsseminars. Darauf folgend beschreiben wir die Aufteilung in Kleingruppen und berichten über erste Implementationserfolge. Zum Schluss verweisen wir auf den ausgearbeiteten Zwischenbericht der Projektteilnehmer und bieten einen Ausblick auf mögliche Ziele in naher Zukunft.

2 Vorarbeiten

Ein Prototyp für ein spurbasierten Routenplanungssystem wurde im Rahmen der Diplomarbeit von Shahid Jabbar entwickelt und der Projektgruppe zur Verfügung gestellt. Er ermöglicht das Einlesen von GPS-Spuren, die Berechnung des Schnittpunktgraphen, grundlegende Graphkompressionsverfahren, Punktlokalisation und kürzeste-Wege-Suche. Ansätze zur Effizienzsteigerung wurden entwickelt und dynamischen Veränderung der Karten vorgeschlagen. Aktuelle GPS Spurdaten wurden z.B. durch Rad- oder Taxi-Fahrten oder durch protokollierte Spaziergänge bereitgestellt. Die Darstellung der erstellten Information, sprich die Visualisierung einzelner GPS-Sequenzen, erwies sich als nicht weiter schwer. In der Arbeit wird entweder durch eigene oder durch referenzierte Resultate nachgewiesen, dass alle im Quelltext detailliert beschriebenen Algorithmen und Verfeinerungen optimale Pfade liefern. Dabei ist die Verwendung von Heuristiken in Form von unteren Schranken essentiell. Die Beweise zur Zulässigkeit und Konsistenz der Schätzungen in den vorgestellten insbesondere statischen Modellen begründen die Optimalität von A* als Variante des Algorithmus von Dijkstra und haben somit starken Einfluss auf die Laufzeitresultate.

Die genutzten Erkenntnisse aus dem Bereich der *Algorithmischen Geometrie* beinhalten auch neuere Ergebnisse zum Aufbau und zur Verwaltung effizienter Anfragestrukturen. Die durchgeführten Implementierungen reichen von der Eingabe über verschiedene Vorverarbeitungsschritte, der Anwendung von mitunter gerichteten Suchverfahren, bis hin zur Ausgabe der GPS-Spuren. Es werden die Grenzen existierender elektronischer Routenplanungssysteme diskutiert und die Besonderheit des entwickelten Systems hervorgehoben. Dabei werden Grundlagen zu den Themen der geodätischer Datenerhebung und Informationsverarbeitung angeboten.

Zur Datenreduktion empfiehlt die Arbeit als Verfahren des geometrischen Rundens den *Douglas-Peuker* Algorithmus, der den Linienzug einer Spur rekursiv vereinfacht. Auch wenn der Ansatz zu den einfachsten Verfahren gehört, ist er als Repräsentant dieser Klasse gut ausgewählt. Für die statistische Datenverbesserung auf Basis zusätzlicher inertialer Informationsquellen werden *Kalman* Filtermethoden vorgeschlagen. Auch wenn in der Praxis der Arbeit kein Inertialsystem verwendet wurde, ist die beschriebene Öffnung in die Richtung konsequent. Für die reduzierten und gefilterten

Daten nutzt der Ansatz zur Erkennung von Spurkreuzungen den *Algorithmus von Bentley und Ottmann*, der aus einer Menge von Sequenzen durch das *scan-line* Prinzip den assozierten gewichteten Graphen, inklusive der Schnittpunkte effizient berechnet. Zur Suche wird der Graph zudem auf alle Schnitt- und alle Endpunkte reduziert. Die gefundenen kürzesten Wege im kompakten Graphen werden dann wieder eindeutig dekomprimiert. Anfragen werden mit effizienten Punktlokalisationsverfahren basierend auf einer *Delaunay-Triangulation* (bzw. einem *Voronoi-Diagramm*) der GPS Punktmenge als Struktur nächster Nachbarn an den bestehenden Graphen angepasst. Innerhalb dieses Graphen werden mit verschiedenen Kürzeste-Wege Verfahren längen- oder zeiteffiziente Routen berechnen und dem Anwender fahrtbegleitend zur Verfügung gestellt werden. Die Implementatierung des Systems *GPS-Route* fußt auf ein unübliches, aber sehr flexibles und prägnantes Programmierparadigma: die *Mixin* Programmierung. Hier werden die Basisklassen als `c++`-Templates genutzt. Das Einlesen von GPS-Spuren und die Berechnung des Schnittpunktgraphen wurde mit Hilfe der LEDA-Bibiliothek verwirklicht. Das System *GPS-Route* kann über eine Internet-Schnittstelle aufgerufen und gestartet werden. Dabei wurden zwei Optionen voneinander getrennt: Die textuelle Verarbeitung von Daten über ein CGI/Perl Skript und die graphische Interpretation in dem System VEGA. Beide Ansätze sind noch beschränkt in der Größe der Datenmenge, die sie einlesen bzw. darstellen können. Das System verfügt noch nicht über die vorgeschlagene Netzwerk-Komponente, in der mehrere Anfragen und GPS-Aktualisierungen von verschiedenen Benutzern an ein und denselben Server gestellt werden.

3 Seminarphase

Es wurde ein Einstiegsseminar zu den folgenden Themen veranstaltet.

– *External Memory Point Localization*

  Zusammenfassung und ergänzende Erklärungen zum Artikel *I/O-efficient Point Location using Persistent B-Trees* von Lars Arge, Andrew Danner und Sha-Mayn Teh. Die Autoren beschreiben eine neue Variante persistenter B-Bäume, die im externen Speichermodell sowohl theoretisch als auch praktisch effizient ist. Sie setzt keine totale Ordnung auf den Elementen voraus, benötigt linearen Platz $O(N/B)$, und kann Updates in logarithmisch vielen I/Os ausführen. Unter Verwendung dieser Datenstrukturn kann das Punktlokationsproblem I/O-effizient gelöst werden.

– *Geometric Containers*

  Es wird angenommen, dass, wie bei Routenplanungssystemen üblich, ein statischer, gewichteter Graph vorliegt, auf dem eine ganze Reihe von Kürzeste-Wege-Anfragen gestellt werden. Es stehen also wenige Änderungen vielen Anfragen gegenüber, wobei die Idee naheliegend ist, ein Preprocessing durchzuführen, um die dann folgenden Anfragen schneller beantworten zu können. Eine Speicherung aller kürzesten Wege kommt aufgrund des quadratischen Speicherplatzbedarfs von im Allgemeinen nicht in Frage. Dann wird der Fall eines dynamischen Graphen betrachtet. Bei Änderungen der Gewichtungen im Graphen, sollen nicht alle Ergebnisse des Preprocessing verworfen, sondern möglichst effizient aktualisiert werden.

– *Dynamic Shortest Paths*

  Es wird die dynamische Version des Kürzesten-Wege-Problems mit einem Startknoten betrachtet. Das Problem besteht darin, die Informationen über die kürzesten Wege wieder herzustellen, nachdem sich der Graph verändert hat. Die Informationen über die kürzesten Wege sollen dabei nicht komplett neuberechnet werden, sondern möglichst von der älteren Version übernommen werden. Die häufigsten Änderungen an einem Graphen, die in diesem Zusammenhang vorgenommen werden, sind das Erhöhen und Verringern der Kantengewichte und das Hinzufügen und Löschen von Kanten. Wenn eine beliebige Folge dieser Operationen auf einem Graphen erlaubt sein soll, bezeichnet man das Problem als *dynamisch*.

– *Geometric Data Structures*

  Bei der orthogonale Bereichssuche geht es um Suchanfragen innerhalb von Datenbanken. Viele dieser Suchen können als geometrische Suchen interpretiert werden. Um diese Suche als geometrisches Problem darzustellen, stellen wir die Daten als Punkte im zweidimensionalen Raum dar. Wir stellen *KD-Bäume* und verschiedene Varianten von *Bereichsbäumen* vor. Beim *Windowing* stellen wir uns ein Navigationssystem im Auto vor, von welchem wir uns im näheren Umkreis unseres Aufenthaltsortes die Straßenkarte anzeigen lassen möchten. Befinden wir uns z.B. in Deutschland, möchten wir uns einen Ausschnitt aus der gesamten Strassenkarte von diesem Land anzeigen lassen. Das Navigationssystem muss also mithilfe einer Suchanfrage und unserem Standort diesen Kartenabschnitt ausgeben. Hierfür werden drei Datenstrukturen vorgestellt: *Intervallbäume*, *Prioritäts-Suchbäume* und *Segmentbäume*.

– *Geometric Filtering*

  Die Kombination von GPS mit INS (inertial navigation system) und dem Kalman-Filter ermöglicht eine wesentliche Verbesserung der Navigation. Der Kalman-Filter, der die statistischen Modelle beider Systeme berücksichtigt, kann die Vorteile beider nutzen, um den Einfluß ihrer nachteiligen Merkmale optimal zu minimieren. Der Kalman Filter löst das allgemeine Problem der Schätzung und Vorhersage des zukünftigen Verhaltens eines Prozesses, der als lineares, dynamisches System modelliert ist. Die Untersuchung und Vorhersage des Prozesses erfolgt diskret über die Zeit und rekursiv. Bei jedem Iterationsschritt wird der Zustand des Prozesses neu berechnet, wobei die Matrizen, die die Historie des Prozesses, die Varianzen und die Kovarianzen der Prozessvariablen beschreiben, aktualisiert werden. Man nimmt eine dem Prozessrauschen zugrundeliegende Gaussverteilung an.

– *Geometric Rounding*

  Beim geometrischen Runden wird die Komplexität geometrischer Objekte reduziert. Dies bedeutet, dass insbesondere die zur Darstellung der geometrischen Objekte benötigten Datenmengen verkleinert werden. Geometrische Objekte werden einerseits durch einen kombinatorischen und einen numerischen Datenanteil beschrieben. Als Beispiel sei ein Graph genannt, bei dem der kombinatorische Teil beschreibt in welcher Weise die Knoten des Graphen verknüpft sind und der numerische Teil die Koordinaten der Knoten in der Ebene darstellt. Als erster Rundungsalgorithmen werden die Algorithmen von *Green und Yao*, *Snap Rounding* und *Polygonzugvereinfachungen*, wie der *Douglas-Peuker Algorithmus* vorgestellt.

– *Map Generation*

  Die Aufgabe zur *Kartenerzeugung/Kartenverbesserung* kann in nacheinander folgenden Schritten aufgeteilt werden. *Sammeln* unbearbeiteter GPS Daten von Fahrzeugen, die entlang von Strassen fahren. *Filtern und Resamplen* von Spuren, um GPS-Störungen zu reduzieren. *Aufteilung der Spuren* in Sequenzen von Segmenten, indem die Spuren an eine Initialisierungskarte anpasst werden. Diese Initialisierungskarte könnte z.B. eine kommerziell erhältliche digitale Karte sein. Alternativ kann auch durch einen Algorithmus, eine Netzwerkstruktur allein durch Spuren erstellt werden. Für jedes Segment wird eine Straßenmittellinie generiert, die ungefähr die Form der Straße erfasst. Diese dient uns dann später auch als Referenzlinie für die einzelnen Spuren auf einer Straße. Die Anzahl der Spuren auf einer Straße wird bestimmt, indem ein Cluster-Algorithmus benutzt wird.

– *Electronic Maps and GPS Devices*

  Es wird erläutert, wie ein GPS-System aufgebaut ist. Dabei wird erklärt, wie man seine Position mit Hilfe von Referenzzeiten, die mittels elektromagnetischen Wellen ausgesendet werden, bestimmen kann. Der eindimensionalen Fall wird schrittweise bis zur dritten Dimension fortgeführt. Unterschieden wird anfänglich nach synchronen und asynchronen Uhren. Elektronische Karten lassen sich grob in zwei Kategorien einordnen. *Rasterkarten* bestehen aus einem Rasterbild, wie z.B. Bitmap, und einer Positionsreferenzdatei. Diese Datei enthält Informationen, um das Rasterbild in die Welt einzuordnen. Eine einfache Möglichkeit ist für die obere linke und untere rechte Ecke des Bildes die Koordinaten anzugegeben. Dafür muss das Bild rechtswinklig angeordnet sein. Es sind auch weitere, komplexere Referenzen möglich. *Vektorkarten* bestehen aus geometrischen Objekten. Hierbei werden nur die Positionen der Objekte gespeichert. Die Visualisierung kann diese nun geeignet darstellen und somit ist auch ein uneingeschränkter Zoom möglich.

– *Traffic Models*

  Hier werden Verkehrsmodelle, sowie der GPS-Simulator von Horst und Ralf Lichtenheld vorgestellt. Als erstes wird erklärt, was Verkehrsmodelle sind und welche Unterstützung hierbei die Mikrosimulation leistet. Danach wird der Blick in die Zukunft der Verkehrsmodellierung geworfen und untersucht, welche Fragestellungen relevant sind, um ein solches Modell auf spezielle Bedürfnisse auszurichten. Als nächstes werden repräsentativ sechs Modelle vorgestellt und charakterisiert. Der letzte Punkt befasst sich mit einem GPS-Simulator, der auf Möglichkeiten und Funktionen untersucht wird.

– *Algorithm Animation*

  VEGA steht für **V**isulisation **E**nvironment for **G**eometric **A**lgorithms und stellt ein verteiltes System dar, welches sich zur Aufgabe gemacht hat geometrische Algorithmen (Triangulation, Point Localization, Sweep Line Techniques etc.) zu Visualisieren und graphisch zu animieren. Mit anderen Worten: Die Arbeitsweise von geometrischen Algorithmen wird mit Vega auf dem Bildschirm sichtbar gemacht. Das Vega System besteht im Wesentlichen aus dem Vega-Server, dem Vega-Client und einer Algorithmenbibliothek. Es wird ebenfalls ermöglicht, die Bibliotheken LEDA und CGAL einzubinden und zu nutzen.

– *LEDA*

  LEDA ist die Abkürzung für *Library of Efficient Data Types and Algorithms*. LEDA wird seit 1988 am Max-Planck-Institut für Informatik in Saarbrücken entwickelt. Das Grundprogramm von LEDA wird in vier Pakete, *Basis*, *Graph*, *Geometrie* und *GUI*, unterteilt. Es handelt sich hierbei um eine C++ Klassenbibliothek, die eine beträchtliche Anzahl von effizienten Datentypen und Algorithmen beinhaltet: Keller, Schlangen, Listen, Mengen, Wörterbücher, gerichtete, ungerichtete und planare Graphen, Linien, Punkte und Ebenen; dazu kommen viele Algorithmen der Graphen- und Netzwerktheorie sowie der algorithmischen Geometrie. Zu dem Basispaket von LEDA existieren zahlreiche Erweiterungspakete, die sogenannten LEPs, die die Funktionalität von LEDA um die gewünschten Anwendungsbereiche erweitern. LEPs gibt es z.B. für die Bereiche *abstrakte Voronoi Diagramme*, *dynamische Graphenalgorithmen* und *erweiterte Geometrie*.

– *Multi-Level Shortest Path*

  Der Multi-Level Graph ist ein hierarchisches Konzept, bei dem der Originalgraph auf geeignete Weise in Schichten zerlegt wird. Aus dem Orginalgraphen konstruieren wir einen neuen Graphen $M(G, S_1, S_2, ..., S_l)$ durch Einfügung von zusätzlichen Kanten zwischen benachbarten Leveln des Graphen. Eine Schicht besteht aus einer Knotenmenge S_i, welche eine Untermenge der Knotenmenge des Originalgraphen ist. Für die Knotenmenge S_i gilt: $V \supset S_1 \supset S_2 \supset ... \supset S_l$. Das Gewicht einer neuen Kante zwischen zwei Schichten entspricht die Länge des kürzesten Weges zwischen den beiden Knoten auf dem Originalgraphen. Für jede Kundenanfrage wird anhand des Start- und Zielknotens aus dem Multi-Level Graphen ein Teilgraph aufgebaut und mit Hilfe des Algorithmus von Dijkstra der kürzesten Weg zwischen Start und Zielort berechnet.

Die Ausarbeitungen zu den Themen lieferten die Grundlage der folgenden Implementierungen und einen großen Bestandteil des Zwischenberichts[1]. Die folgenden Themen wurden von den Betreuern der PG präsentiert: *External Memory Graph Search*, *System GPS-Route* und *Mobile Programming*.

Als technisches Inventar wurde ein T-Mobile MDA mit GPS-Mouse, eine Telefonkarte und 512 MByte SD Speicherchip zur Verfügung gestellt. Desweiteren konnte ein GARMIN GPS Empfänger in Verbindung mit einem Laptop zur Aufzeichnung genutzt werden. Bei der Kartenervektorisierung und Simulation wurde der DTK-Kartensatz von Dortmund des Landesvermessungsamts NRW (Bonn) genutzt, von dem ca. 300 *Kacheln* der Größe 1 km^2 im TIFF Format zum Stückpreis von ca. 1 Euro pro Kachel eingekauft wurden. Die genauere Ausgangsbasis sind 18×16 Kartenelemente, die in der Auflösung von 2000 mal 2000 Pixeln vorliegen.

4 Kleingruppen

Bei der Aufteilung in Kleingruppen haben sich vier Gruppen gebildet:

[1] Ausgewählte Literaturangaben zu den Themen sind auf der Projektgruppeninternetseite `http://ls5-www.cs.uni-dortmund.de/~edelkamp/gpsroute` zu finden.

– Simulation: *Kartenvektorisierung* und Erstellen einer *Simulationumgebung* zur automatischen Erzeugung von GPS-Spuren
– Map Generation: *Dynamische Kartengenerierung* auf Basis von vorhandenen GPS-Spurdateien.
– Routenplanung: Datenstrukturen und Algorithmen zur *Punktlokalisation* und *Kürzeste-Wege Berechnung* auf dem Server, basierend auf den erstellten Karten.
– PDA: *PocketPC* Frontend zur *Karten- und Routendartellung, GPS Spuraufnahme* und *Schnittstelle zum Server* zur Verarbeitung von *Start/Ziel Anfragen*

Jede Kleingruppe hat eine Ausarbeitung über die ersten Implementationserfolge geschrieben. Diese Arbeiten schließen den Zwischenbericht ab und werden im folgenden zusammengefasst.

4.1 PDA Programmierung

Ziel der PDA-Gruppe ist es, ein Programm zu entwickeln, das die Nutzung des Navigationssystems ermöglicht. Die Anforderungen an das Programm unterteilen sich in vier verschiedene Bereiche. Zunächst wird ein *Rahmenprogramm* erstellt, das es erlaubt, die Straßenkarten, die errechneten Routen und den aktuellen Standort zu visualisieren. Um die errechneten Routen zu empfangen, ist eine *Netzwerkanbindung* erforderlich. Diese wird weiterhin benötigt, um die von der GPS-Maus empfangenen und gespeicherten Daten zu versenden. Hieraus ergibt sich der dritte Aufgabenbereich, die *Filterung und Speicherung* der empfangenen GPS-Daten. Der vierte und letzte Aufgabenbereich stellt die *Umrechnung* der GPS-Koordinaten in Gauss-Krüger-Koordinaten dar. Diese Konvertierung ist wichtig, um empfangene Daten, die im GPS-Format vorliegen, auf dem Kartenmaterial darzustellen.

Man kann sich in der Karte bewegen, indem man den Navigationsknopf des PDA betätigt. Durch mittiges Drücken auf diesen Knopf wird in vier verschiedenen Auflösungen zyklisch gezoomt, drückt man auf den rechten/linken/oberen/unteren Rand des Navigationsknopfs, kann man die Karte scrollen (vergl. Abbildung 1). Bei Auswahl des Menüpunktes „Neue Route“ öffnet sich ein Fenster, in welchem der Modus der Routenberechnung ausgewählt werden kann (schnellster/kürzester Weg) und die Start- und Zielangaben gesetzt werden können. Klickt man nun auf den Knopf *Suchen*, wird die Karte aufgerufen, auf welcher man den Start- und Zielpunkt mit Hilfe von Fähnchen setzen kann (vergl. Abbildung 2).

4.2 Kartenvektorisierung

In der Teilgruppe *Simulation* waren die Vektorisierung der Karteninformation und der Aufbau des Simulationsgraphen die zentralen Themen im ersten Semester. Die geeignete Verkehrssimulation selbst wird im zweiten Semester behandelt.

Der Startpunkt, das Kartenrohmaterial, besteht aus Pixeldateien. Jedem Pixel (Position i,j) ist ein Farbwert zugeordnet. Die Farbe selbst ist als Tripel von rotem, grünem und blauem Farbanteil gegeben (RGB-Farbskala). Zu unserem sich später herausstellenden Vorteil sind die gegebenen Karten nicht verrauscht, was bedeutet, dass sich die farbigen Flächen nicht aus einer Vielzahl unterschiedlicher RGB-Werte ergeben, sondern alle einen eindeutigen Farbwert haben. Aufgrund dieser Eigenschaft ist es einfach,

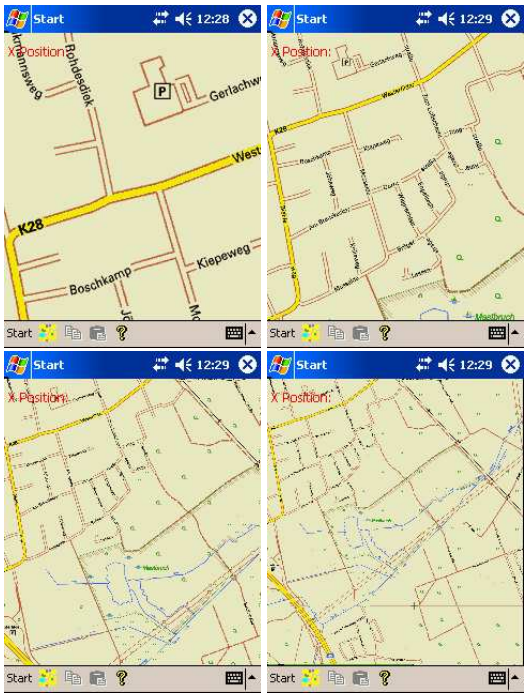

Abbildung 1. Skalieren der Karte.

die für die Straßengraphgenerierung interessanten Straßenflächen vom Rest zu trennen. Zu diesem Zwecke müssen vom Benutzer lediglich die Farbwerte der gewünschten Straßen angegeben werden. Ein einfacher Algorithmus setzt die Farbwerte derjenigen Pixel auf schwarz, welche die vorgegebenen Farben haben. Alle Pixel, die nicht eine der vordefinierten Straßenfarben haben, werden schließlich auf weiß gesetzt. Bei diesem einfachen sogenannten Filter treten allerdings Schwierigkeiten auf.

Zum einen gibt es schwarze Schriftzüge, welche Straßennamen, Ortsteile und andere wichtige Orte angeben und innerhalb der Straßen verlaufen oder Straßen schneiden. Darüberhinaus sind auch alle Straßenbahn- und Eisenbahnstrecken schwarz markiert.

Man könnte nun einerseits die Farbe schwarz als Straßenfarbe definieren. In diesem Fall wählt man allerdings auch gleichzeitig alle Bahnstrecken als Straßenfläche aus. Andererseits könnte man die Farbe schwarz nicht auswählen, würde somit aber weiße Lücken auf den Straßenflächen erhalten. Der größte Teil der Straßen ist in den TIFF-Dateien als weiße Fläche gegeben. Bei der Auswahl dieser Farbe werden dann allerdings auch Sportplatze und Parkplatzmarker und andere kleinere Dinge ausgewählt. Die Extraktion der Straßenflächen it also nicht ganz sauber und bedarf manueller und auch automatisierter Nachbearbeitungen. Die manuelle Nachbearbeitung besteht aus einfachen Mal- bzw Radierwerkzeugen, wie sie aus jedem Bildbearbeitungsprogramm bekannt sind. Zu den automatisierten Nachbearbeitungen zählen: Erosion, Dilatation,

Abbildung 2. Festlegen der Route.

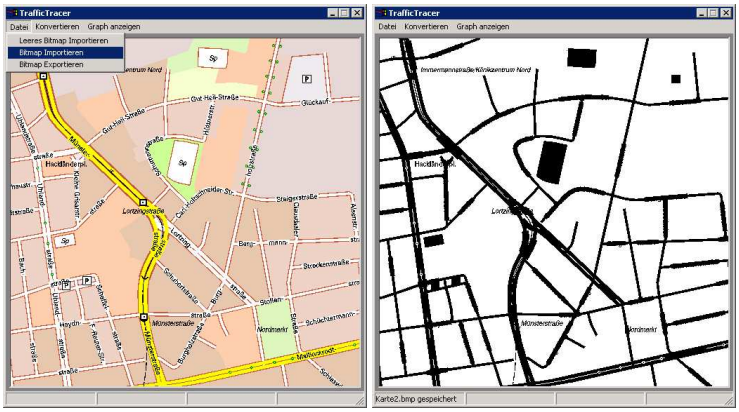

Abbildung 3. Extraktion der Straßenflächen.

Morphologisches Öffnen, Morphologisches Schließen, Lücken Schliessen und Fragmente Entfernen.

Im folgenden sollen diese sechs automatisierten Bildbearbeitungsalgorithmen anhand von Screenshots erklärt werden. Grundlage für die ersten *Säuberungen* der Straßenflächen durch Filterung sind morphologische Operationen. Die Basis der mathematischen Morphologie ist die Mengenlehre. Neben den geläufigen Mengenoperationen wie Vereinigung Durchschnitt, Differenz und Komplement spielen auch Reflektion ($\hat{A} = \{w \mid w = -a, a \in A\}$) und Translation ($(A)_z = \{c \mid c = a + z, a \in A\}$) eine wichtige Rolle.

Erosion und Dilatation Für zwei Mengen A und B in Z^2 ist die Erosion von A und B definiert als $A \ominus B = \{z \mid (B)_z \subseteq A\}$. Eine Erosion von A und B besteht also aus allen Punkten z, für die gilt, dass B um z verschoben in A enthalten ist. Die Menge B wird in diesem Zusammenhang als *strukturierendes Element* bezeichnet. Die Form und Größe des strukturierenden Elementes ist für das Erosionsergebnis entscheidend. Aufgrund der Aufgabenstellung ist die Wahl eines symmetrischen struk-

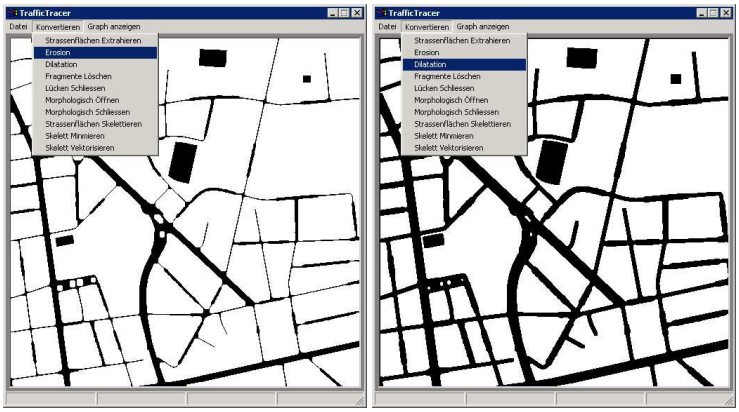

Abbildung 4. Erosion und Dilatation.

turierenden Elementes notwendig, da ansonsten die Straßenflächen *verzerrt* würden. Da durch die Erosion Pixel (vom Rand der schwarzen Straßenflächen aus) gelöscht werden, darf das strukturierende Element darüber hinaus nicht zu groß sein, da es sonst passieren kann, dass *dünne* Straßenflächen komplett gelöscht werden. Aus diesem Grunde wird ein 3x3-Pixel großes strukturierendes Element gewählt, wobei der Benutzer durch mehrfache Erosion die Straßenflächen stetig verdünnen kann. Auf diese Weise wird bereits ein Großteil der störenden Schriftzüge, sowie Straßen- und Eisenbahnschienen eliminiert:

Für zwei Mengen A und B in Z^2 ist die Dilatation definiert als $A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \oslash\}$. Somit sind Erosion und Dilatation komplementäre Operationen und es gilt die Gleichung: $(A \ominus B)^c = A^c \oplus \hat{B}$. Auch bei der Dilatation wird ein 3x3-Pixel großes strukturierendes Element gewählt. Bei der Benutzung des Dliatationsfilters muss der Anwender darauf achten, dass eng beieinander liegende Straßenflächen nicht zu einer großen Straßenfläche verschmolzen werden. Allerdings können kleine Lücken innerhalb der Straßenflächen verkleinert oder geschlossen werden.

Morphologisches Öffnen und morphologisches Schliessen Das morphologische Öffnen einer Menge A durch das strukturierende Element B (beide aus Z^2) ist definiert als $A \circ B = (A \ominus B) \oplus B$. Es ist also nichts anderes, als eine Erosion von A und B, unmittelbar gefolgt von einer Dilatation des Erosionsergebnisses. Durch diese Operation werden schmale Verbindungen zwischen schwarzen Flächen aufgebrochen, die Konturen werden abgerundet und kleine Vorsprünge eliminiert.

Das morphologische Schließen einer Menge A durch das strukturierende Element B (beide aus Z^2) ist definiert als $A \bullet B = (A \oplus B) \ominus B$. Durch diese Operation werden ebenfalls Konturen geglättet, allerdings werden im Gegensatz zum morphologischen Öffnen schmale Verbindungen zwischen schwarzen Flächen verstärkt, und kleine Löcher und Einbuchtungen gefüllt:

Lücken Schliessen Ein Problem welches nicht allein durch Dilatationen gelöst werden kann ist die Eliminierung von ´Lücken´ innerhalb breiter Straßenflächen, z.B. bei Autobahnen. Zwar könnten durch mehrfache Dilatationen solche Lücken geschlossen

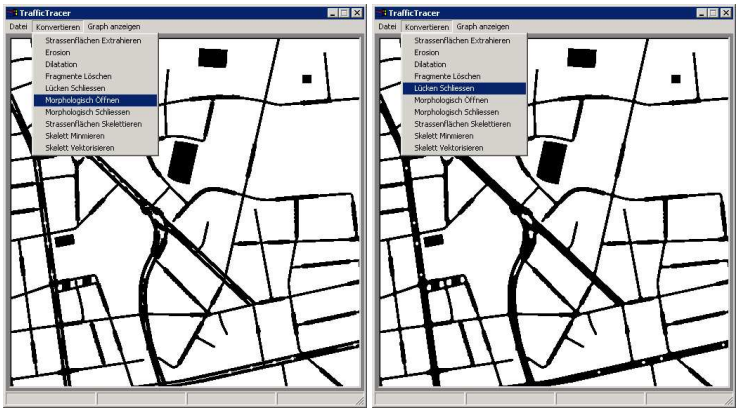

Abbildung 5. Morphologisches Öffnen und Schließen.

werden, wobei dann allerdings viele Straßenflächen verschmolzen würden (s.o.). Abhilfe schafft in diesem Zusammenhang der Algorithmus *Lücken Schließen*, welcher lediglich für alle weißen Pixel überprüft, ob dieses mindestens $n \in \{1, 2, ..., 8\}$ schwarze Nachbarpixel hat. Die Anzahl n soll vom Benutzer festgelegt werden können, wobei sich ein Wert von $n = 5$ als praktikabel erwiesen hat.

Fragmente Entfernen Durch die Erodierung werden die Straßenbahnschienen leider nicht ganz eliminiert. Außerdem befinden sich unter anderem Sportplätze und verschiedene Kartensymbole auf dem 0/1 - Bitmap. Ziel dieses Algorithmus ist es, diese schwarzen Stellen zu finden und weiß zu färben.

Es wird jedes Pixel des Bitmaps untersucht. Findet sich ein schwarzes Pixel *x* , wird in weiteren Schritten die Umgebung von *x* analysiert. Es wird die Eigenschaft ausgenutzt, dass Straßenzüge stets aufeinanderfolgende schwarze Pixel besitzen. Sind nun alle umgebenden Pixel um *x* weiß, kann man davon ausgehen, dass *x* isoliert von anderen schwarzen Pixeln ist und es weiß gefärbt werden kann.

In der Methode wird ein ungerader Ganzzahlwert *i* größer 3 übergeben, der die maximale Größe eines Quadrates angibt, welches um *x* gezogen wird. In einer Schleife wird iterativ das Quadrat von Kantenlänge 3 bis *i* vergrößert. Bestehen die Kanten eines Quadrates zu einem Zeitpunkt nur aus weißen Pixeln, so wird der Flächeninhalt weiß gefärbt und die Iteration abgebrochen. Ist dies nicht der Fall wird das Quadrat entsprechend der Iteration (bis zur maximalen Kantenlänge *i*) vergrößert. Randpixel des Bitmaps müssen in Sonderfällen betrachtet werden.

Die Skelettierung einer Pixeldatei basiert auf der Verwendung morphologischer Operatoren. Nachdem die Straßenflächen durch die vorgestellten Filter sowie durch manuelle Nachbearbeitung sauber extrahiert wurden führt ein Skelettierungsalgorithmus zu einer Darstellung aus welcher dann im nächsten Schritt ein Straßengraph entwickelt werden kann. Das Skelett einer s/w-Pixeldatei ist, vereinfacht und anschaulich gesagt, eine Menge von dünnen Linien, welche die *Mittelachsen* der ursprünglichen schwarzen Flächen darstellen. Auf die Straßenflächen bezogen bedeutet dies, dass das Straßenskelett die Mittellinien der ursprünglichen Straßenflächen darstellen soll. In der

mathematischen Morphologie ist das Skelett von A definiert als $S(A) = \bigcup_{k=0}^{k \leq K} S_k(A)$ mit $S_k(A) = (A \ominus B) \circ B$, wobei B das strukturierende Element ist und $(A \ominus kB) = (\ldots(A \ominus B) \ominus B) \ominus B \ominus \ldots) \ominus B$ gilt. K ist der letzte iterative Schritt bevor A zu einer leeren Menge erodiert, also $K = \max\{k \mid (A \ominus kB)\} \neq \oslash$. Eine erste Implementierung dieser klassischen Definition eines Skelettes hat nicht das gewünschte Ergebnis gebracht. Durch obige Definition wird nicht der Zusammenhang des Skeletts garantiert. Dies hatte zur Folge, dass gerade an Straßenkreuzungen (und anderen Stellen) das Skelett zerbricht und eine Grapherstellung aus dem Skelett sehr schwierig oder unmöglich wird. Eine Lösung bietet ein heuristischer Ansatz, der von Blum 1967 vorgestellt wurde [Digital Image Processing, Rafael C. Gonzales, Second Edition, Kap. 11.1.5, S.650]. Er basiert auf der so genannten Mittelaxentransformation (MAT). Die MAT einer Region R mit Grenze G ist folgendermaßen definiert. Für jeden Punkt p in R finde man alle nächsten Nachbarn auf der Grenze G. Falls p mehr als einen nächsten Nachbarn hat gehört er zu der Mittelachse, also dem Skelett. Eine noch anschaulichere Definition einer MAT kann über ein Steppenfeuer gemacht werden. Man stelle sich eine zusammenhängende Fläche (homogenen und) trockenen Steppengrases vor, welches gleichzeitig an seinen Grenzen entzündet wird. Die MAT dieser Grasregion besteht aus allen Punkten, welche von mindestens zwei Brandfronten gleichzeitig erreicht wird. Eine direkte Umsetzung dieser Idee führt zu ineffizienten Algorithmen, da die Distanzen von jedem inneren Punkt zu allen Randpunkten berechnet werden müssten. Der in der Implementierung benutzte Algorithmus geht folgendermaßen vor: Die zu skelettierende s/w-Grafik wird in 2 Basisschritten bearbeitet, welche dann sukzessive wiederholt werden, bis keine Veränderungen mehr gemacht werden.

Eine direkte Generierung des Graphen aus dem Skelett, welches mit obigem Algorithmus generiert wurde, ist immer noch problematisch, da das Skelett nicht maximal verdünnt ist. Ein maximal verdünntes Skelett sei definiert als ein Skelett, welche zerfallen würde wenn man einen weiteren Pixel mit mindestens 2 Nachbarn löscht. Diese Bedingung ist Voraussetzung dafür, dass durch die lokale Untersuchung der Umgebung eines Pixel auf dem Skelett, dieser eindeutig als Kreuzungspixel, als Sackgaßenendpixel oder als ein Pixel im Straßenverlauf identifiziert werden kann. Der Algorithmus zur Skelettverdünnung muss mit 2^8 = 256 Möglichkeiten für die Nachbarschaftskonstellation umgehen können und die richtige Entscheidung über das Löschen oder Nichtlöschen eines schwarzen Pixels treffen. Auch dieses Problem wurde *heuristisch* gelöst:

Nachdem das Skelett auf geeignete Weise vorbereitet wurde kommt nun ein Trackingalgorithmus zum Einsatz, welcher einen zusammenhängenden, und gerichteten Graphen erzeugt. Der Algorithmus basiert auf der Sweepline-Technik. Die Pixel werden spaltenweise (Sweepline) betrachtet. Sobald ein Pixel gefunden wird, welches zum Skelett gehört, wird eine Teilroutine aufgerufen, welche die Zusammenhangskomponente bearbeitet, zu der der gefundenen Pixel gehört.

Dabei ist zu beachten, dass nicht alle in den Listen gespeicherten Nachfolger eines Kreuzungspixels aufgerufen werden müssen. Es kann vorkommen, dass man durch die Traversierung des Skelettes auf diese Weise, zu Kreuzungsknoten gelangt, die bereits zuvor besucht wurden, und deren Nachbarliste noch nicht vollständig abgearbeitet wurde. Eine spezielle Markierung der noch nicht abgearbeiteten Nachfolgerpixel eines Kreuzungspixels löst dieses Problem. Erst wenn auf der durch den Sweeplinealgorithmus gefundenen Zusammenhangskomponente der Graph erzeugt wurde, traversiert die Sweepline weiter das Bild und sucht die nächste Komponente. Auf diese Weise können

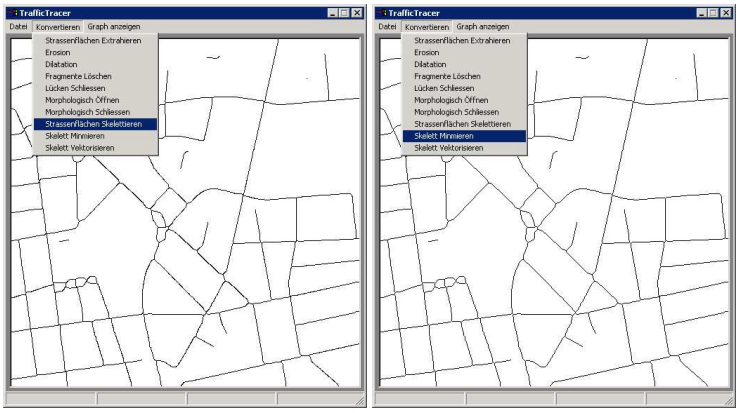

Abbildung 6. Skelettierung und Minimierung.

auch isolierte schwarze Flächen (wie sie in den Karten u.a. bei Sportplätzen vorkommen) eliminiert werden.

4.3 Kartengenerierung

Die *Map Generation* Gruppe hat sich im ersten Semester der Projektgruppe mit der Erstellung eines flexiblen, leicht erweiterbaren Prototypen zur Generierung von Wegekarten (z.B. Straßenkarten) beschäftigt. Im folgenden wird ein kurzer Überblick über dessen Funktionsweise geliefert und im Weiteren näher spezifiziert. Der Prototyp verlangt als Eingabe GPS-Spuren im NMEA-Format. Diese werden über das Internetprotokoll TCP/IP entgegengenommen. Die Daten werden in ein für dieses Programm lesbares Format gebracht und zwischengespeichert. Bei entsprechenden Voraussetzungen - genug Systemressourcen - wird diese gespeicherte Spur weiterverarbeitet. Dazu müssen bereits verarbeitete Spuren geladen und mit der aktuellen Spur zusammengeführt werden. Hierfür wird ein Clusteralgorithmus verwendet. Die Verwaltung der bereits verarbeiteten Spuren ist durch eine Datenbank mit vorgeschaltetem Cache realisiert. Die Datenbank enthält somit immer die gerade aktuelle Version der Karte und kann nach evtl. Konvertierung zur Routenplanung weiterbenutzt werden.

Die Wegekarte ist ein Graph mit zeitannotierten Kanten. Dieser Graph wird jedoch nicht in klassischer Art gespeichert. Es wurde hier eine Zwei-Komponenten-Struktur implementiert: Die Knoten, die annähernd den GPS-Punkten entsprechen, werden in konstant großen Kacheln gespeichert, die durch entsprechende Zugriffsmethoden eine schnelle Nachbarsuche in einem zu spezifizierenden Umfeld ermöglicht. Momentan ist die Nachbarsuche nur durch den Brute-Force-Ansatz realisiert, d.h. alle Punkte auf einer Kachel werden getestet. Die dazugehörigen Kanten werden jeweils in einer listenähnlichen Struktur als Folge von Knoten-IDs gespeichert, wobei eine Kante nur aus Knoten mit Grad zwei bestehen darf, ausgenommen Start- und Endknoten. Somit stellt eine Kante ein Wegestück zwischen zwei Kreuzungen dar. Das zugrunde liegende Koordinatensystem ist das der geographischen Länge, Breite und Höhe.

Abbildung 7. Vektorisierung und Grapherzeugung.

Die Verwaltung der Daten auf der Festplatte übernimmt eine Datenbank, die sowohl unter Windows, als auch Linux bzw. Unix angesteuert werden kann. Dazu wird unter Windows das native *ODBC-Interface* benutzt, während unter Linux/Unix der Zugriff über *unixODBC* abgewickelt wird. Die Struktur der Datenbank ist sehr einfach gehalten. Es ist ein Server realisiert worden, der standardmäßig auf Port 9000 lauscht und bei entsprechenden Anfragen reagiert. Er nimmt die GPS-Spur, die laut NMEA-0183 Standard im ASCII-Format vorliegen, entgegen und beendet nach Übermittlung die Verbindung. Der empfangene String wird in einzelne GPS-Punkte umgewandelt und in zeitlich korrekter Reihenfolge zu einer linearen Liste verknüpft.

Für die Verwaltung und das Anstoßen der Weiterverarbeitung der Traces ist der *TileManager* verantwortlich. Liegt mindestens ein neuer Trace vor, so wird er aktiv: Ein Trace wird nach vorgegeben Kriterien (z.B. neuester Trace) ausgewählt, die von diesem Trace beeinflussten Kacheln werden gesperrt und ein neuer *TraceProcessor* wird für diese Spur angelegt und gestartet. Der *TraceProcessor* kombiniert im weiteren den neuen Trace mit der bestehenden Wegekarte, indem er entweder bestehende Kanten verschiebt oder neue Kanten anlegt.

4.4 Routing

Die Hauptaufgabe der Routinggruppe bestand in der Entwicklung und Implementierung einer Schnittstelle, über die Benutzer Routinganfragen stellen können. Die Schnittstelle sollte Geräteunabhängig (Handheld, Laptop, ...) sein. Außerdem sollten geeignete Datenstrukturen und Speed-Up Techniken erweitert und implementiert werden, so dass Routinganfragen schnell beantwortet werden können.

Die Schnittstelle wird über eine Client/Server-Architektur basierend auf dem TCP/IP-Protokoll realisiert. Als Grundlage für die Datenstrukturen wird die LEDA-Bibliothek in Version 3.1.9 verwendet, sowie verschiedene *Template*-Klassen, die jeweils unterschiedliche Aspekte und Speed-Up Techniken bezüglich des Algorithmus von Dijkstra

zur Berechnung kürzester Weg implementieren. Diese *Template*-Klassen wurden an der Universität Konstanz entwickelt.

Der *Routinggraph* ist ein gerichteter Graph, der alle zum Routen benötigten Informationen enthält. Darunter verstehen wir Kreuzungen, Fahrzeiten zu verschiedenen Tageszeiten und Straßenlängen. Eine Kreuzung wird dabei nicht notwendigerweise durch einen einzelnen Knoten repräsentiert. Der Routinggraph abstrahiert vom konkreten Straßenverlauf. In einer Datenbank existiert deshalb zu jeder Kante des Routinggraphen eine Folge von GPS-Koordinaten, die den genauen Straßenverlauf beschreiben. Der genaue Straßenverlauf wird beim Berechnen eines kürzesten Weges nicht benötigt, so dass eine getrennte Datenhaltung geeignet erscheint. Der Routinggraph wird von der Map-Generation-Gruppe in Abhängigkiet von GPS-Traces erzeugt und in einer Datei abgelegt. Diese Datei wird in regelmäßigen Abständen (z.B. alle 24 Stunden) gelesen und einen Graph für Routingfunktionen erzeugt. Die Graph-Information wird als ASCII-Datei gespeichert. Routinganfragen, die unterstützt werden sind:

1. Berechnung eines *kürzesten* (S, T)*-Weges*. Um diese Berechnungen ausführen zu können, werden lediglich die Straßenlängen benötigt.
2. Berechnung eines *schnellsten* (S, T)*-Weges*.
   (a) Berechnung eines schnellsten (S, T)-Weges zu einer *vorgegebenen Abfahrtszeit* t *am Knoten* S.
   (b) Berechnung eines schnellsten (S, T)-Weges zu einer *vorgegebenen Ankunftszeitzeit* t *am Knoten* T.

   Um diese Berechnungen ausführen zu können, werden die Fahrzeitinformationen benötigt. Leicht einzusehen ist, dass die Varianten 2a und 2b symmetrisch sind. Variante 2b entspricht Variante 2a im Graphen G', indem alle Kanten umgedreht worden sind.

Eine Routinganfrage setzt sich nun wie folgt zusammen:

1. Start- und Zielkoordianten (S, T).
2. Die Angabe der Bewertungsfunktion *Strecke* oder *Zeit*. Im Falle der Zeitfunktion muss zusätzlich zwischen Variante 2a und 2b gewählt werden, wobei auch ein Abfahrts- bzw. Ankunkftszeitpunkt t angegeben werden muss.

Alle Routinganfragetypen haben gemeinsam, dass zunächst zwei Knoten (S', T') im Routinggraphen bestimmt werden müssen, von denen aus ein kürzester/schnellster Weg berechnet wird, da man nicht davon ausgehen kann, dass die Knoten (S, T) im Routinggraphen enthalten sind. Eine naheliegende Lösung für diese Problem ist es, diejenigen Punkte (S', T') zu bestimmen, die minimalen Abstand zu (S, T) haben. Gegebenenfalls kann es auch sinnvoll sein, alle von den Knoten (S', T') ausgehenden Kanten aus der Datenbank zu laden und zu untersuchen, an welchen Kanten die Knoten (S, T) am nächsten liegen.

Die drei von uns unterstützten Routingvarianten erfordern alle die Lösung eines kürzesten Wege Problemes mit verschieden Bewertungsfunktionen. Variante 1 entspricht dem allgemein bekannten *Single-Source-Single-Target Shortest-Path* Problem und kann mit dem Algorithmus von Dijkstra gelöst werden. Die Varianten 2a und 2b sind ebenfalls *Single-Source-Single-Target Shortest-Path* Probleme. Dabei ist zu berücksichtigen, dass die Bewertungsfunktion zeitabhängig ist. Mit einer entsprechenden Anpassung können auch diese Varianten mit dem Algorithmus von Dijksta gelöst werden.

Um die Berechung eines kürzsten Weges zwischen einem Knotenpaar (s, t) zu beschleunigen, stehen zwei Vorverarbeitungsmöglichkeiten zur Auswahl. *Lösung des All-Pair-Shorstest-Path Problem*: Dieser Ansatz ermöglicht, dass Routinganfragen prinzipiell sehr schnell beantwortet werden können. Ein Nachteil dieses Ansatz ist allerdings, dass er viel Speicherplatz benötigt. *Verwendung von Containern:* Die zentrale Idee dieses Ansatz ist es, zu jeder Kante (s, v) eine Datenstruktur zur Verfügung zu stellen, in der alle Knoten t gespeichert werden, so dass ein kürzester Weg von s nach t über die Kante (s, v) führt. Speed-Up Techniken zur Berechnung schnellster Wege sind zum jetzigen Zeitpunkt noch nicht realisiert. Erste Experimente für das kürzeste Wege Problem haben gezeigt, dass die Vorverarbeitung zur Berechnung der Container auf sehr grossen Graphen extrem zeitaufwendig ist.

Um aktuelle Verkehrsinformationen aufzunehmen und dadurch Kantengewichte im Routinggraphen anzupaßen, soll ein Web-Interface entwickelt werden. Im Web-Interface sollen Staus, sowie Straßensperrungen durch einen prozentualen Faktor zwischen mehreren Knoten im Graphen angegeben werden. Die Einträge zur temporären Kantengewichtsänderung können für unbestimmte Zeit, als auch für eine festgelegte Zeitperiode eingetragen werden. Die Eintragungen erfolgen in einer Datenbank, die vom Routingsystem in bestimmten Zeitabständen durchsucht wird, um den Routinggraphen zu aktualisieren. Einträge, die sich über Ihre Existenzzeit hinaus in der Datenbank befinden, werden vom Routingsystem automatisch gelöscht.

5 Ausblick

Der Zwischenbericht der Projektgruppe dokumentiert den Stand nach einem von zwei Semestern. Die Projektgruppe ist ihrem Mindestziel bedeutend näher gekommen. Die Grundbausteine des Systems wurden bereits verwirklicht. So ist eine Vektorisierung und Straßengraphgenerierung der Karten genauso möglich, wie die Darstellung selbiger in verschiedenen Vergrößerungen auf dem PDA. Die Kartengenerierungsgruppe ist dabei, die Überlagerung von per TCP/IP eingefügten GPS Spuren zu verarbeiten; die Datenbankanbindungen stehen. Kürzeste Wege können in kleinen Datensätzen gefunden werden und auch die Punklokalisation funktioniert.

Allerdings sind viele Implementierungen noch nicht stabil und für den Praxiseinsatz geeignet. Desweiteren sind die Schnittstellen zwischen den Gruppen noch nicht ausreichend getestet.

Abschliessend einige, aber längst nicht alle möglichen Erweiterungsmöglichkeiten:

- Ein Ziel ist es, die *praktische Akzeptanz* deutlich zu erhöhen und Firmenkontakte aufzubauen. Ein großeres Manko für die *Autonavigation* ist derzeit allerdings der Mangel an Informationen über die Straßentyp und Ausstattung (Autobahn, Landstraße, Einbahnstraße und Ampel). Außerdem ist hier die industrielle Konkurenz recht groß.
- Unser *universitäres Interesse* gilt der Entwicklung, Ausarbeitung und Verwirklichung von von effizienten, insbesonderer geometrischen Algorithmen. So soll die Einbindung dynamischer Routeninformation z.B. über nicht (mehr) passierbares Gelände und stockenden Verkehrsfluss ermöglicht werden. Desweitern ist die Integration weiterer Messdaten, die Implementierung von Externspeicherplatzsuchverfahren, sowie die effektive Zusammenfassung und Bereinigung von GPS-Daten von Interesse.

Danksagung Die Arbeit wird von der *Deutsche Forschungsgemeinschaft* in den Projekten *Heuristische Suche* (Ed 74/3) und *Gerichtete Modellprüfung* (Ed 74/2) unterstützt.

Literatur

1. H. Ackermann, M. Bettahi, R. Brüntrup, M. Drodzynski, V. Faber, A. Gaubatz, A. Härtel, S.-J. Hong, M. Liebe, A. Scheidler, B. Schulz, F. Wang, S. Edelkamp, S. Jabbar, and T. Mehler. Zwischenbericht PG 452: GPS-Route. Technical report, Universität Dortmund, 2004.
2. R. K. Ahuja, K. Mehlhorn, J. B. Orbin, and R. E. Tarjan. Faster algorithms for the shortest path problem. *Journal of the ACM*, pages 213–223, 1990.
3. J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. *Transactions on Computing*, 28:643–647, 1979.
4. D. H. Douglas and T. K. Peuker. Algorithms for the reduction of the number of points. *The Canadian Cartographer*, 10:112–122, 1973.
5. S. Edelkamp, S. Jabbar, and T. Willhalm. Accelerating heuristic search in spatial domains. In *Workshop on Planning and Configuration (PUK)*, pages 1–20, 2003.
6. S. Edelkamp, S. Jabbar, and T. Willhalm. Geometric travel planning. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2003.
7. S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In *Computer Science in Perspective*, Lecture Notes in Computer Science, pages 128–151. Springer, 2003.
8. S. J. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, pages 153–174, 1987.
9. S. Gutmann. Markov-kalman localization for mobile robots. In *In International Conference on Pattern Recognition (ICPR)*, 2002.
10. C. A. Hipke and S. Schuierer. Vega—a user-centered approach to the distributed visualization of geometric algorithms. In *Computer Graphics, Visualization and Interactive Digital Media (WSCG)*, pages 110–117, 1998.
11. S. Jabbar. GPS-based navigation in static and dynamic environments. Master's thesis, Universität Freiburg, 2003.
12. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
13. Navigation Technologies, Sunnyvale, CA. *Software Developer's Toolkit*, 5.7.4 Solaris edition, December 1996.
14. S. Schroedl, S. O. Rogers, and C. K. H. Wilson. Map refinement from GPS traces. Technical Report RTC Report Number 6/2000, DaimlerChrysler Research and Technology North America, Palo Alto, CA, November 2000.
15. D. Wagner and T. Willhalm. Geometric speed-up techniques for finding shortest paths in large sparse graphs. Technical report, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, 2003.
16. D. Wagner, T. Willhalm, and C. Zaroliagis. Dynamic shortest path containers. Technical report, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, Computer Technology Institute, and Department of Computer Engineering & Informatics University of Patras, 2003.
17. J. Wang, S. Rogers, C. Wilson, and S. Schroedl. Evaluation of a blended DGPS/DR system for precision map refinement. In *Proceeedings of the ION Technical Meeting 2001*, Institute of Navigation, Long Beach, CA, 2001.