Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 2794

SmartGPS – Lokationsmodell für PerFlows

Eduard Huber

Studiengang:	Informatik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. Matthias Wieland Dipl.-Inf. M.Sc. (USA) Stephan Urbanski
begonnen am:	23. Juni 2008
beendet am:	23. Dezember 2008
CR-Klassifikation:	C.1.4 E.1 F.2.0 G.3 H.4.2 I.4.8

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1. Motivation

Mobile elektronische Geräte sind im heutigen Alltag quantitativ bereits weit verbreitet [Como8, ABCo8]. Auch ihre qualitative Integration in alltägliche Prozesse schreitet immer weiter voran. Dennoch beschränkt sich der Einsatz derartiger Geräte im Wesentlichen auf separate nicht miteinander verknüpfte Einzelapplikationen wie Navigation im Straßenverkehr, Surfen im Internet, Kalender- und Telefon-Funktionalität. Den Benutzer über mehrere Einzelschritte hinweg unterstützende Applikationen sind hingegen kaum präsent. Bereits 1991 stellte Mark Weiser in einer zukunftsweisenden Arbeit das Szenario einer weitreichenden Integration elektronischer Geräte in menschliche Alltagsprozesse vor [Wei91]. Hierin beschreibt Weiser die ubiquitäre Unterstützung des Benutzers durch zahlreiche, miteinander interagierende, in vielen Fällen transparente, elektronische Geräte und Applikationen.

Modellierung persönlicher kontextsensitiver Prozesse [UBRo6, UHSRo6], im Folgenden als *PerFlows* bezeichnet, ist eine erfolgsversprechende Möglichkeit zur Umsetzung Weisers Vision. Die Idee der PerFlows besteht darin alltägliche Prozesse wie Einkaufen, Fahrt zur Arbeit oder Abhalten eines Seminars zu modellieren [Dudo6], diese anschließend in einem *PerFlow-Manager* ausführen zu lassen. Ein PerFlow-Manager ist eine Art persönlicher, ubiquitärer Assistent, welcher situationsabhängig dem Benutzer die jeweils als nächste auszuführenden Schritte vorschlägt, respektive diese bereits im Voraus initiiert. Anders als die umfangreich untersuchte Modellierung von Geschäftsprozessen [LRoo] stellen PerFlows dynamische, im hohen Maß situations- und lokationsabhängige Prozesse dar. Das Hochfahren des Arbeitsrechners am Arbeitsplatz, Vorwärmen der Wohnung vor dem jeweiligen Eintreffen des Benutzers oder das Vorschlagen eines Einkaufes und Generieren entsprechender Einkaufsliste sind einige der möglichen Beispiele, in denen das PerFlow-Management den Benutzer in seinen Alltagsprozessen unterstützen kann. Die Entscheidung des PerFlow-Managers bezüglich des jeweils als nächsten auszuführenden Schrittes hängt indessen primär von der aktuellen Situation des Benutzers, speziell dessen momentanen Lokation, ab. Zuverlässige Lokationserfassung des Benutzers und weiterer hinsichtlich der PerFlows relevanter Personen und Gegenstände bildet auf Grund dessen eine wichtige Voraussetzung für die Funktionalität des PerFlow-Managements.

1.2. Ziele der Arbeit

Das zentrale Ziel dieser Arbeit besteht in der Erarbeitung eines Lokationsmodells, welches primär eine umfassende Möglichkeit zur persönlichen Lokationserfassung und -modellierung bietet. Das Modell soll hierbei in der Lage sein prinzipiell alle zur Verfügung stehenden Informationenquellen zu nutzen. Diese können von einem GPS-Empfänger über Infrarotbake, bis hin zu einem Terminkalender reichen. Des Weiteren analysieren wir die über eine persönliche Lokationserfassung hinausgehenden funktionalen Anforderungen des PerFlow-Managements an ein Lokationsmodell und zeigen auf auf welche Weise das in dieser Arbeit vorgestellte Lokationsmodell diese erfüllen kann. Neben den funktionalen Anforderungen analysieren wir das erarbeitete Lokationsmodell ebenso hinsichtlich dessen Effizienz.

1.3. Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in sieben Kapitel. Im Kapitel 2 gehen wir zunächst auf das der Arbeit zugrunde gelegte Systemmodell ein und zeigen hierbei die Rolle des Lokationsmanagements auf. Im Kapitel 3 analysieren wir die seitens des PerFlow-Managements an Lokationsmodelle gestellten Anforderungen. Dabei gehen wir sowohl auf funktionale als auch auf Effizienz bezogene Anforderungen ein. Mit bestehenden Lokationsmodellen beschäftigt sich Kapitel 4. Hier stellen wir einige der bereits vorhandenen Lokationsmodelle vor, beschreiben ihre Eigenschaften und analysieren diese in Hinsicht auf die im Kapitel 3 gestellten Forderungen. Kapitel 5 bildet den Kern der Arbeit, in dem das vom Autor entwickelte Lokationsmodel *SmartGPS* vorgestellt wird. Zunächst wird die Lokationsmodellierung statischer (Abschnitt 5.2) sowie dynamischer (Abschnitt 5.3) Objekte getrennt betrachtet, im Anschluss eine Möglichkeit des gemeinsamen Einsatzes aufgezeigt (Abschnitt 5.4). Kapitel 6 befasst sich kurz mit der im Zuge der Arbeit entwickelten Implementierung der im Abschnitt 5.3 vorgestellten Lokationsmodellierung und -erfassung dynamischer Objekte. Zuletzt fasst das Kapitel 7 die Arbeit abschließend zusammen und wagt einen Ausblick.

Kapitel 2

Systemmodell

Das dieser Arbeit zugrundegelegte Systemmodell besteht aus vier Komponenten. Diese lassen sich schichtartig übereinander legend vorstellen und werden im Folgenden einzeln vorgestellt (vgl. Abbildung 2.1).

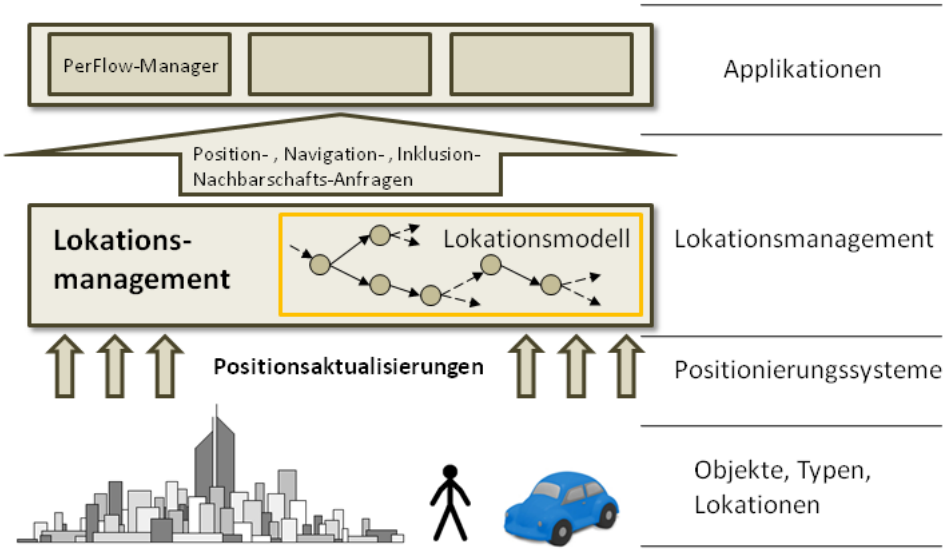


Abbildung 2.1.: Systemmodell

Objekte

Objekte bilden die unterste Ebene des Systemmodells. Diese repräsentieren alle materiellen Gegenstände der realen Welt: Personen, Gebäude, Automobile, usw. Den Objekten werden im Modell zwei Attribute zugeordnet:

- Zum einen besitzt jedes Objekt einen *Typ*, welcher aus einer Menge an möglichen Typen so gewählt ist, als dass er das entsprechende Objekt auf eine möglichst umfassende Art charakterisiert.
- Des Weiteren besitzt jedes Objekt eine *Lokation*. Lokation eines Objektes bezeichnet einen Punkt oder Ausdehnung im Raum, an dem sich das Objekt befindet, respektive von diesem eingenommen wird.

Für die weitere Verwendung der besprochenen Objekte, Objekttypen und Lokationen in dieser Arbeit werden diese formal als Mengen definiert:

- $Obj :=$ Menge aller Objekte.
- $Type :=$ Menge möglicher Objekttypen.
- $Loc :=$ Menge aller Lokationen, wobei zunächst nicht explizit zwischen ausgedehnten Lokationen und Punktlokationen unterschieden wird.

Des Weiteren definieren wir die bereits besprochene Typisierung und Lokalisierung der Objekte als Funktionen. Es gilt:

$$loc : Obj \rightarrow Loc, \text{ mit} \tag{2.1}$$
$$loc(o_i) = l_i, \text{ wobei } l_i \text{ die vom Objekt } o_i \text{ eingenommene Lokation ist, und} \tag{2.2}$$
$$type : Obj \rightarrow Type, \text{ mit} \tag{2.3}$$
$$type(o_i) = t_i, \text{ wobei } t_i \text{ Typ des Objektes } o_i \text{ ist} \tag{2.4}$$

Positionierungssysteme

Sensoren, welche Lokationen der eben beschriebenen Objekte aufzeichnen und verfolgen, gehören zu den Positionierungssystemen des Systemmodells. Sensoren können sich je nach Zweck der Aufgabe in ihrer Art stark unterscheiden. Einmalige Vermessung eines Geländes betrachten wir ebenso als Sensor-Tätigkeit, wie die eines GPS-Empfängers, welcher aktuelle Koordinaten eines fahrenden Fahrzeuges erzeugt. Zu einem Positionierungssystem des Modells gehören somit alle Geräte und Aktivitäten, deren Intention in der Zuordnung von Lokationen zu Objekten, der Berechnung der Funktion *loc* also, besteht.

Lokationsmanagement

Lokationsmanagement bildet nun die Komponente, deren Zweck in der Aufnahme, Verarbeitung und Abspeicherung der durch Positionierungssysteme generierten Daten, sowie in der Anfragebeantwortung an die Applikationsschicht besteht. Zur Abspeicherung der Lokationsdaten dient ein Lokationsmodell, dessen Art sich in erster Linie an die Funktionalität der das Lokationsmanagement nutzenden Applikationen, insbesondere an die von ihnen gestellten Anfragen (vgl. Kapitel 3) richtet. Nach [Leo98, Dom01] lassen sich allgemein zwei Lokationsmodelltypen unterschieden: Symbolische [Sat05] sowie geometrische Lokationsmodelle. Die ersteren verwenden symbolische Lokationsangaben wie

Straßennamen, Haus- und Raumnummern, Postleitzahlen oder IP-Adressen. Abspeicherung geometrischer Lokationsangaben hingegen erfordert zunächst eine Definition des Koordinatensystems, auf welches sich die Lokationsangaben (Koordinaten) schließlich beziehen. Lokationsmodelle und Lokationsmanagement bilden den zentralen Schwerpunkt dieser Arbeit.

Applikationen

Die das Lokationsmanagement nutzenden Applikationen bilden die oberste Schicht des Systemmodells. Beispiele sind in [CK00, Maa98] aufgeführt. In dieser Arbeit richten wir unsere primäre Aufmerksamkeit den PerFlows zu und betrachten im folgenden Kapitel welche der aus dem Bereich der PerFlows resultierenden Anfragen seitens des Lokationsmanagements zu unterstützen sind.

Kapitel 3

Anforderungsanalyse

Lokationsmodelle lassen sich, wie im letzten Kapitell kurz angesprochen, grob in die Kategorie der geometrischen und die der symbolischen einteilen. Um diese gegeneinander vergleichen, bzw. ihre Vor- und Nachteile diskutieren zu können, wird eine gewisse Metrik benötigt. Ziel dieses Kapitels besteht in der Präsentation der in erster Linie aus den Bedürfnissen des PerFlow-Managements resultierenden Anforderungen an Lokationsmodelle. Die Erfüllung, bzw. Nichterfüllung der behandelten Anforderungen gibt uns die Möglichkeit Lokationsmodelle gegeneinander zu vergleichen und somit über deren praktischen Nutzwert im Bereich der PerFlows zu diskutieren. Die hier vorgestellten Anforderungen werden nach [LL07] in zwei Klassen eingeteilt:

1. *Effektivitätsanforderungen*. Bietet ein Lokationsmanagement die geforderte Funktionalität und erfüllt somit seinen Zweck, ist es effektiv. Effektivität eines Lokationsmanagements ist primär von dem verwendeten Lokationsmodell und den darauf eingesetzten Algorithmen abhängig.
2. *Effizienzanforderungen*. Effizienz beschreibt das Verhältnis des gewonnenen Nutzens zum benötigtem Aufwand (an Speicher, Laufzeit). Im Fall des Lokationsmanagements betrachten wir dessen Effizienz aus Sicht der zur Erfüllung der Funktionalität eingesetzten Algorithmen. Ein besonderes Augenmerk richten wir hierbei auf die jeweilige Laufzeitkomplexität.

Im Folgenden diskutieren wir einzelne Anforderungen entsprechend der vorgestellten Klassifikation.

3.1. Effektivitätsanforderungen

Die meisten Anforderungen an Lokationsmodelle im Bereich der Effektivität lassen sich aus den seitens des Benutzers bzw. Applikationen gestellten Anfragen an das Lokationsmanagement ableiten. In unserem Fall ist die Applikation ein PerFlow-Manager, welcher aus den vom Benutzer modellierten PerFlows abgeleitete Anfragen an das Lokationsmanagement stellt. Die in diesem Kapitel behandelten Anforderungen werden, wenn möglich,

formal in Form einer zu berechnenden Funktion (in der Reihenfolge ihrer Relevanz) dargestellt. Hierbei orientieren wir uns partiell an den in [BD05] diskutierten Anforderungen, erweitern die Liste jedoch um weitere, unter Anderem in [Dro03] erwähnte, Punkte.

- *Lokation (position queries).* Bestimmung der Lokation eines Objektes ist das primäre und wichtigste Ziel eines Lokationsmanagements. Hierbei soll die Lokationsermittlung sowohl für statische (Gebäude, Straßen, Haltestellen), als auch für dynamische (Menschen, Fahrzeuge) Objekte möglich sein. Lokationserfassung ist auch insofern ein wichtiger Punkt, da weitere in diesem Kapitel diskutierte Funktionalitäten auf diesem aufbauen. Formal soll ein Lokationsmanagement die – im Idealfall total definierte – Funktion

  $$loc : Obj \rightarrow Loc, \text{ mit} \tag{3.1}$$
  $$loc(o_i) = l_j, \text{ wobei } l_j \text{ die Lokation von Objekt } o_i \text{ ist,} \tag{3.2}$$

  berechnen können.

  Im Fall von geometrischen Koordinaten besteht die Möglichkeit innerhalb des globalen Koordinatensystems lokale, eventuell dynamische (ineinander verschachtelte) Koordinatensysteme zu definieren. Insbesondere für größere, dynamische Objekte (Zug, Schiff) erweist sich die Möglichkeit Lokationsangaben in einem relativen Koordinatensystem anzugeben, als vorteilhaft.

- *Distanz.* Die Distanzfunktion, im Folgenden als *dist* bezeichnet, ist eine auf der Menge der Lokationen *Loc* definierte Metrik. Für diese muss gelten:

  $$dist : (Loc \times Loc) \rightarrow \mathbb{R} \tag{3.3}$$
  $$dist(l_i, l_i) = 0 \tag{3.4}$$
  $$dist(l_i, l_j) = 0 \Rightarrow l_i = l_j \tag{3.5}$$
  $$dist(l_i, l_j) = dist(l_j, l_i) \tag{3.6}$$
  $$dist(l_i, l_k) \leq dist(l_i, l_j) + dist(l_j, l_k) \tag{3.7}$$

  Im Fall von geometrischen Koordinaten ist die Distanzfunktion zwischen zwei Punkten meist klar definiert. Handelt es sich um ein kartesisches Koordinatensystem, lässt sich Satz von Pythagoras anwenden. Distanz zwischen ausgedehnten Lokationen, wie in Abbildung 3.1 gezeigt, kann hingegen mehrdeutig interpretiert werden:

  - $dist_{min}$: Abstand zwischen zwei sich am nächsten noch im jeweiligen Raum befindenden Punkte. Es gilt: $dist_{min}(A, D) > dist_{min}(A, B) = dist_{min}(A, C) = dist_{min}(B, C) = dist_{min}(B, D) = dist_{min}(C, D)$
  - $dist_{max}$: Abstand zwischen zwei sich am weitesten noch im jeweiligen Raum befindenden Punkte. Es gilt: $dist_{max}(A, D) = dist_{max}(B, C) > dist_{max}(A, B) = dist_{max}(C, D) > dist_{max}(B, D) = dist_{max}(C, A)$
  - $dist_{avg}$: Abstand zwischen den Schwerpunkten jeweiliger Räume. Es gilt: $dist_{avg}(A, D) > dist_{avg}(B, C) > dist_{avg}(A, C) = dist_{avg}(C, D) > dist_{avg}(A, B) = dist_{avg}(B, D)$

Aus Sicht des PerFlow-Managements spielen die genannten Definitionen für den praktischen Nutzen eine eher untergeordnete Rolle, da in keiner die Länge einer tatsächlichen *begehbaren* Strecke zwischen zwei Räumen berücksichtigt wird. Die in der Abbildung dargestellte gestrichelte Linie bildet den wohl typischen zurückzulegenden Pfad zum Erreichen des Raumes D aus Raum A heraus.

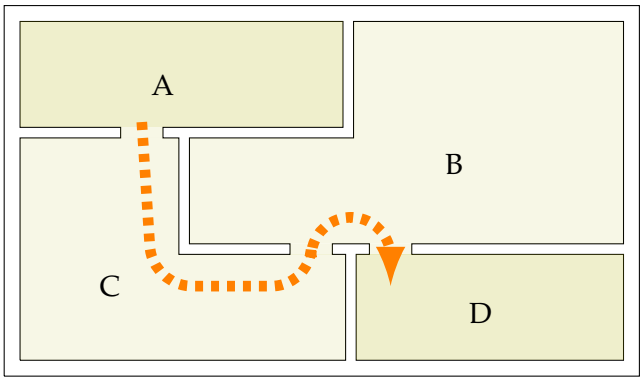


Abbildung 3.1.: Mehrdeutige Abstandsberechnung

Das Beispiel verdeutlicht drei im Bereich der Distanzfunktion bestehende Problemfelder:

1. Wird Funktion *dist* auf ausgedehnte Lokationen angewandt, so ist Ergebnis mehrdeutig interpretierbar. Um Willkürlichkeiten zu vermeiden, verlangen wir eine (lediglich) auf Lokationspunkte angewandte Distanzberechnung.

2. Bei der Distanzberechnung zwischen zwei Lokationen, möchte sie im Bereich der PerFlows praktischen Nutzen besitzen, ist eher die Länge einer begehbaren Verbindung als die einer Luftlinie zu berücksichtigen. Interpretation der *Begehbarkeit* kann indessen höchst Nutzer spezifisch (Autofahrer / Fußgänger / Rollstuhlfahrer) ausgelegt werden.

3. Distanz zweier Lokationen ist nicht notwendigerweise als deren räumliche Entfernung zu begreifen. Andere Kriterien wie der für einen Autofahrer relevante Spritverbrauch oder Steigungswinkel für einen Fußgänger besitzen eine eventuell substanziellere Bedeutung. Im Allgemeinen muss ein Lokationsmodell arbiträre Nutzer abhängige Gewichtungen der Verbindungen zwischen Lokationen erlauben. Auch unendlich hohe Gewichtungen für nicht begehbare Verbindungen müssen erlaubt werden können.

- *Nächster Nachbar (nearest neighbour queries).* Die Problematik der Ermittlung des nächsten Nachbarn ist mit der der Distanzberechnung eng verbunden. Auch hier muss die Definition des Begriffes *nächster* anfangs geklärt werden. Bei der Berechnung besteht die Aufgabe anschließend darin zu einer gegebenen Lokation $l_i \in Loc$ ein Objekt vom bestimmten vorgegebenen Typ zu finden, dessen Distanz zu l_i minimal ist. Praktische Anwendung für die Berechnung des nächsten Nachbarn

besteht beispielsweise im Auffinden des nächstgelegenen Restaurants oder Werkstätte. Anders als beispielsweise in [DR03, BD05] verlangen wir bei der Ermittlung des nächsten Nachbarn dessen Typ explizit mit angeben zu können (Restaurant, Werkstätte). Formal lässt sich die Funktion wie folgt definieren:

$$nearest : Loc \times Type \rightarrow Obj, \text{ mit} \tag{3.8}$$
$$nearest(l_i, t_j) = o_k, \text{ wobei} \tag{3.9}$$
$$type(o_k) = t_j \text{ und} \tag{3.10}$$
$$\forall o_l \in Obj : (type(o_l) = t_j \wedge o_l \neq o_k) \Rightarrow$$
$$(dist(loc(o_l), l_i) \geq dist(loc(o_k), l_i)) \tag{3.11}$$

- *Inklusion (range queries).* Im Fall der Inklusionsanfrage werden für eine (ausgedehnte) Lokation alle sich in dieser befindenden Objekte zurückgegeben. Beispiel für eine Anwendung dieser Anfrage bietet GeoCast [DR03, NI97a]. Hier sollen alle sich in einer bestimmten Lokation befindenden Personen (beispielsweise im Fall eines Feueralarms) benachrichtigt werden. In der Domäne der PerFlows liegt die Relevanz etwa im Auffinden bestimmter Geschäfte in einer für den Benutzer unbekannten Stadt. Auch hier soll der Typ der aufzufindenden Objekte explizit als Parameter der Funktion mit angeben werden können. Die zu berechnende Funktion definieren wir wie folgt:

$$incl : Loc \times Type \rightarrow 2^{Obj}, \text{ mit} \tag{3.12}$$
$$incl(l_i, t_j) = \{o_0, o_1, \ldots o_{n-1}\}, \text{ wobei} \tag{3.13}$$
$$\forall k \in \{0, 1, \ldots n-1\} : (loc(o_k) \subseteq l_i \wedge type(o_k) = t_j) \tag{3.14}$$

  Bedingung $loc(o_k) \subseteq l_i$ in Gleichung 3.14 besagt, dass Objekt o_k sich innerhalb der Lokation l_i auffindet (vgl. Punkt *Mengenoperationen* weiter unten).

- *Inverse Inklusion.* Bei dieser Anfrage ist eine Art Umkehrung der im letzten Punkt besprochenen Inklusion gefordert. Für eine, im Allgemeinen kleine, Lokation ist ein diese umschließendes Objekt eines bestimmten Typs gesucht. Für eine Person suchen wir also beispielsweise den Raum, das Gebäude, die Stadt in der sich diese aufhält. Formal lautet die Funktion wie folgt:

$$enclosedBy : Loc \times Type \rightarrow Obj \text{ mit} \tag{3.15}$$
$$enclosedBy(l_i, t_j) = o_k, \text{ wobei} \tag{3.16}$$
$$type(o_k) = t_j \text{ und} \tag{3.17}$$
$$l_i \subseteq loc(o_k) \tag{3.18}$$

  Der letzte Punkt 3.18 sagt aus, dass Lokation l_i in der Lokation $loc(o_k)$ enthalten ist (vgl. Punkt *Mengenoperationen*).

- *Navigation.* Navigationssysteme sind heute insbesondere im Bereich des Straßenverkehrs bereits weit verbreitet. Dessen ungeachtet fokussieren sich die vorhandenen Techniken fast ausschließlich auf Verkehrswege. Navigation auf Gebäude- oder

Raumebene wird hingegen häufig nicht unterstützt. Im Bereich der PerFlows spielt jedoch gerade diese Art der Navigation eine wichtige Rolle.

Die Aufgabe der Navigation besteht allgemein darin für zwei gegebene Punkte einen Pfad (Liste von Lokationspunkten) zu bestimmen, der die beiden gegebenen Punkte miteinander verbindet. Der berechnete Pfad soll unter der gegebenen Gewichtung minimale Kosten aufweisen, was zugleich dessen Begehbarkeit impliziert.

$$nav : Loc \times Loc \rightarrow \{(M, \pi) | M \subseteq Loc, \pi : M \rightarrow M, \pi \text{ bijektiv}\}, \text{ mit} \tag{3.19}$$

$$nav(l_i, l_j) = (l_0, l_1, \ldots l_n), \text{ wobei} \tag{3.20}$$

$$l_0 = l_i \wedge l_n = l_j \text{ und} \tag{3.21}$$

$$\sum_{k=0}^{n-1} dist(l_k, l_{k+1}) \rightarrow min \tag{3.22}$$

- *Mengenoperationen.* Lokationen können als (im Allgemeinen überabzählbare) Mengen von (Lokations-)Punkten betrachtet werden. Folglich erscheint es als sinnvoll auf diesen die üblichen Mengenoperationen wie Vereinigung ($\cup$) oder Schnitt ($\cap$) sowie Ordnungsrelationen $\subset$, $\subseteq$ zu erlauben. Mittels Mengenoperation ließen sich Lokationen beliebig komponieren und als Eingangsparameter der in diesem Kapitel beschriebenen Funktionen einsetzen. In der Domäne der PerFlows kann es beispielsweise von Interesse sein alle sich innerhalb gewisser Gebäude A, B und C befindenden Konferenzräume zu bestimmen. Dies kann, sofern $Konferenzraum$ eine gültige Objekttypbezeichnung ist, durch die Berechnung von $incl((loc(A) \cup loc(B) \cup loc(C)), Konferenzraum)$ ermittelt werden.
- *Metalokationen.* Anlegen einer Metalokation soll dem Benutzer eines Lokationsmodells die Möglichkeit geben bestimmte, einzig für ihn relevante und unter Umständen durch Komposition aus anderen Lokationen und Metalokationen entstandene (fiktive) Lokationen der Art *meine Arbeit*, *mein Zuhause*, *Freizeit* frei zu definieren und anzulegen. Eine Metalokation ist folglich eine Art Referenz, bzw. Menge von Referenzen, auf andere Lokationen. Ein Vorteil der Verwendung von Metalokationen ist in der (allgemein) dynamischen Eigenschaft dieser begründet. Eine Lokation wie beispielsweise *mein Zuhause* ändert sich bei jedem Umzug, die zugrunde liegende Räume und Gebäude bestehen jedoch weiterhin. Die Möglichkeit einer schnellen Änderung einer Metalokation ist folglich vorteilhaft.
- *Visualisierung.* Visualisierung der Lokationen auf einer Karte ist eine naheliegende Applikation für ein Lokationsmanagement. Ergebnisse der in bisherigen Punkten besprochenen Anfragen können dem Benutzer auf diese Weise in visueller Form zurückgegeben werden. Lokationsmodelle mit Geometriemodellierung erweisen sich hierbei als vorteilhaft.
- *Orientierung.* Sofern keine exakte dreidimensionale Lokationsmodellierung eines Objektes erfolgt, bedarf es häufig der Angabe eines zusätzlichen Vektors. Im Bereich der PerFlows ist häufig die Geh- und Blickrichtung des Benutzers von Bedeutung. Diese Angaben können jedoch aus der reinen Lokationsbeschreibung nicht bestimmt werden, wodurch Entscheidung bezüglich der relativen Position bestimmter Objekte,

wie *vorne* oder *hinten*, nicht getroffen werden kann. Insbesondere im Bereich der Navigation ist die Definition von Richtungsangaben von hohem Stellenwert. Lokationsmodelle, welche Orientierungsangaben ermöglichen, erachten wir demzufolge als vorteilhaft.

3.2. Effizienzanforderungen

Die meisten Anforderungen in Bezug auf Effizienz von Algorithmen lassen sich mit dem Begriff der *Skalierbarkeit* beschreiben. Skalierbarkeit ist eines der wichtigsten Kriterien zur Charakterisierung von Algorithmen. Als (zeitlich gut) skalierbar werden im Allgemeinen Algorithmen bezeichnet, welche bei einer Eingabe der Länge n $O(n \log n)$ Schritte benötigen. Algorithmen mit höherer Anzahl an Schritten werden hingegen als schlecht skalierbar bezeichnet. Ähnliches gilt für Speicherintensität.

Im Fall eines Lokationsmanagements bildet Skalierbarkeit der eingesetzten Algorithmen ein essentielles Kriterium für dessen praktische Einsetzbarkeit. Folglich werden wir die in dieser Arbeit vorgestellten Lokationsmodelle (siehe Kapitel 4) neben der gebotenen Funktionalität stets mit Hinblick auf die Skalierbarkeit der Funktionsberechnung analysieren, wobei ein besonderes Augenmerk auf Laufzeit gerichtet ist. Im Folgenden verwenden wir zur Beschreibung der Laufzeitkomplexität eines Algorithmus stets die O-Notation [Sch01]. Diese mag im praktischen Einsatz eine untergeordnete Rolle spielen, für konzeptuellen Vergleich verschiedener Lokationsmodelle ist sie jedoch unumgänglich. Tabelle 3.1 fasst die Anforderungen zusammen, welche wir im Bereich der Laufzeitkomplexität an ein Lokationsmodell stellen.

Funktion	Laufzeit	Ergänzung
$loc(o_i)$	$O(1)$	
$dist(l_i, l_j)$	$O(1)$	
$nearest(l_i, t_j)$	$O(n \log n)$	n: Anzahl der Objekte, welche eine geringere Distanz zu l_i besitzen als das naheste Objekt vom Typ t_j
$incl(l_i, t_j)$	$O(m \log n)$	n: Anzahl der in l_i vorhandenen Objekte m: Anzahl der in l_i vorhandenen Objekte von Typ t_j
$enclosedBy(l_i, t_j)$	$O(\log n)$	n: Anzahl der modellierten Objekte
$nav(l_i, l_j)$	$O(n \log n)$	n: Anzahl der Objekte, welche eine geringere Distanz zu l_i besitzen als die Lokation l_j
$\cap, \cup, \subset, \subseteq$	$O(1)$	

Tabelle 3.1.: Laufzeitanforderungen an Lokationsmodelle

Die hierbei aufgeführten Laufzeiten sind in erster Linie durch die für Suche und Pfadberechnung bekannten Standardalgorithmen [AHU83] motiviert. Wir erlauben Algorithmen, welche eine Suche implementieren (hier: *enclosedBy* und *incl*), logarithmische Zeit in Abhängigkeit der Anzahl der modellierten Objekte zu beanspruchen. Faktor m in der

Laufzeit der Funktion *incl* begründet sich mit der m-maligen Suche nach einem Objekt von Typ t_j. Laufzeiten jener Algorithmen, welche eine Art Pfad berechnen (hier: *nearest* und *nav*), orientieren sich am Dijkstra-Algorithmus [Dij59], dessen Laufzeit $O(n \log n)$ bekannt ist [1]. Für die restlichen Funktionen (*loc*, *dist*, $\cap$, $\cup$, $\subset$, $\subseteq$) ist die konstante Laufzeit damit begründet, dass diese unabhängig von anderen im Lokationsmodell vorhandenen Daten berechnet werden können.

Für alle Arten der Aktualisierungen eines Lokationsmodells (Einfügen eines neuen Objektes, Positionsänderung eines bereits bestehenden Objektes), welche bisher nicht Gegenstand der besprochenen Funktionalität waren, fordern wir ohne weiteres Eingehen konstante Laufzeiten. Die Forderung ist notwendig, um Erstellen eines Lokationsmodells in linearer Zeit zu ermöglichen. Längere Laufzeiten sind dagegen nicht vertretbar.

[1] Die in dieser Arbeit aufgeführte Laufzeit des Dijkstra-Algorithmus bezieht sich auf die Annahme eines linearen Zusammenhangs zwischen der Anzahl der Kanten und Knoten in einem Graph.

Kapitel 4

Bestehende Lokationsmodelle

Lokationsmodelle lassen sich in die Klasse der geometrischen sowie der symbolischen einteilen. Auf allgemeine Eigenschaften dieser wurde im Kapitel 2 kurz eingegangen. In diesem Kapitel werden einige der bestehenden Lokationsmodelle vorgestellt. Die meisten der behandelten Modelle sind in ihrer Art sehr einfach – ihre praktische Einsetzbarkeit infolgedessen beschränkt. Nichtsdestotrotz enthalten diese wichtige Elemente, welche in komplexeren, für den praktischen Einsatz relevanten Lokationsmodellen ihre Verwendung finden. Der Präsentation der Modelle folgt eine Analyse in Bezug auf die im letzten Kapitel besprochenen Anforderungen.

4.1. Geometrische Lokationsmodelle

Geometrische Lokationsmodellierung erfordert eine Definition des Koordinatensystems, in dessen Bezug geometrische Koordinaten interpretiert werden. Globale Systeme wie das in der Praxis verwendete WGS84 [wgs04] (vgl. Anhang A.2) erlauben mittels einer *Referenzellipsoid*-Definition weltweite Lokationsbestimmungen durch Angabe von Längen- und Breitengraden. Lokale Koordinatensysteme hingegen, vorgegeben beispielsweise durch Verlauf der Wände eines Raumes, erlauben eine räumlich nur beschränkte, stattdessen eine eventuell genauere Lokationserfassung [BHC+05].

Die konkrete Modellierung der Lokation eines Objektes kann im einfachsten Fall durch Angabe eines Punktes erfolgen. Für genügend kleine Objekte mag diese Art der Modellierung bereits ausreichen. Objekte größerer Ausdehnung können dagegen je nach Detaillierungsgrad mittels eines zweidimensionalen Polygonzuges, Polygonzuges mit Höhenangabe (2,5-dimensionale Objekte [DR03]) oder auch komplexer dreidimensionalen Modellierung [Rol95] dargestellt werden.

Lokationsmodellierung der Objekte durch Angabe geometrischer Koordinaten reicht zum Erbringen der geforderten Funktionalität häufig nicht aus. Der Grund hierfür liegt unter anderem in der fehlenden Darstellung der Konnektivität einzelner Objekte, bzw. ihrer Lokationen untereinander. Des Weiteren sagt Geometriemodellierung zunächst nichts über mögliche Anordnung der modellierten Objekte innerhalb des Modells aus. Diese kann

für Berechnung einzelner Funktionen jedoch von hoher Bedeutung sein (vgl. *Inklusion*). Geometriemodellierung kann infolgedessen immer nur als Zusatz einer, wie im nächsten Abschnitt 4.2 beschriebenen, umfassenden Datenstruktur des Lokationsmodells betrachtet werden.

4.2. Symbolische Lokationsmodelle

4.2.1. Mengen-basiertes Lokationsmodell

Ein allgemeines und einfaches Lokationsmodell bildet das Mengen-basierte Lokationsmodell [BD05]. Die Basis des Modells bildet ein aus atomaren, nicht notwendigerweise disjunkten, Lokationen bestehendes Universum $U = \{l_1, l_2, \ldots, l_n\}$. Jede Teilmenge $L_i \subseteq U$ mit $L_i = \{l_{i_1}, l_{i_2}, \ldots l_{i_k}\}$ repräsentiert hierbei eine bestimmte Lokation. Ein Mengen-basiertes Lokationsmodell ist eine Menge $M \subseteq 2^U$, wobei $\forall l_i \in U : \{l_i\} \in M$ gilt. Diese Art der Modellierung ist äußerst einfach, bietet dennoch bereits einige, im vorigen Kapitel diskutierte, Funktionalität: Inklusionsbeziehungen, inverse Inklusion, Mengenoperationen sowie Definition von Metalokationen werden vom Modell inhärent unterstützt.

Des Weiteren gestatten Mengen-basierte Lokationsmodelle einfache qualitative Distanzberechnungen. Hierzu wird zunächst Menge $U_{connect} \subset 2^U$ definiert, welche alle *Nachbarschaftsmengen* $N_i \subseteq U$ enthält, in denen jeweils sich angrenzende Lokationen l_j befinden. Für ein beliebiges Lokationspaar l_i, l_j und eine Nachbarschaftsmenge N_k muss

$$l_i, l_j \in N_k \Rightarrow \exists(i_1, i_2, \ldots, i_m) : (l_i | l_{i_1} \wedge l_{i_1} | l_{i_2} \cdots \wedge l_{i_m} | l_j) \tag{4.1}$$

gelten, wobei das Symbol | direkte Angrenzung bzw. Überschneidung zweier Lokationen symbolisiert. Anschließend lässt sich bezüglich der Distanz zweier Lokationen folgende Aussage treffen:

$$\begin{aligned} &dist(l_i, l_j) < dist(l_i, l_k) \Leftrightarrow \\ &\exists N_{i_1} \forall N_{i_2} \in U_{connect} : ((l_i \in N_{i_1} \wedge l_j \in N_{i_1} \wedge l_i \in N_{i_2} \wedge l_k \in N_{i_2}) \Rightarrow |N_{i_1}| < |N_{i_2}|) \end{aligned} \tag{4.2}$$

Die Größe der kleinsten l_i und l_j enthaltende Nachbarschaftsmenge, definiert somit den qualitativen Abstand beider Lokationen.

Trotz Unterstützung vieler Operationen besteht die essentielle Schwäche eines Mengenbasierten Lokationsmodells in dessen Ineffizienz hinsichtlich der Laufzeit und des Modellierungsaufwandes. Praktische Einsetzbarkeit bei höherer Objektanzahl erscheint insofern als unmöglich. Genauere Analyse des Modells erfolgt im Abschnitt 4.4.

4.2.2. Hierarchisches Baummodell

Eine Ursache für Nachteile Mengen-basierter Lokationsmodelle besteht in deren flachen Datenstruktur. Hierarchische Anordnung einzelner Lokationen in einer Baumstruktur löst das Problem. Ein Baummodell [BD05] M ist gegeben durch ein Zweiertupel (V, E), wobei V eine Menge der Lokationen und E eine auf diesem definierte Kantenmenge ($E \subseteq V \times V$) ist. Es gilt:

$$\begin{aligned} &\forall l_i, l_j \in V : (l_i \subset l_j) \Leftrightarrow \\ &\exists l_0, l_1, \ldots l_{n-1} : (l_j, l_0) \in E \wedge (l_{n-1}, l_i) \in E \wedge \forall k \in \{0, \ldots n-1\} : (l_k, l_{k+1}) \in E \end{aligned} \tag{4.3}$$

$$\forall l_i \in V : |\{(l_j, l_i) | l_j \in V \wedge (l_j, l_i) \in E\}| = 1 \tag{4.4}$$

Gleichung 4.3 besagt, dass Kanten Inklusionsbeziehungen zwischen einzelnen Lokationen repräsentieren. Bedingung 4.4 drückt aus, dass allgemeine Überlappungen mehrerer Lokationen, wie diese in Mengen-basierten Modellen gestattet waren, hier nicht möglich sind. Folglich können in einem Baummodell lediglich zwei Beziehungen zwischen Lokationen abgebildet werden: Disjunktheit und Inklusion. Abbildung 4.1 verdeutlicht in einem Beispiel die Abbildung verschiedener Objekte in ein Baummodell.

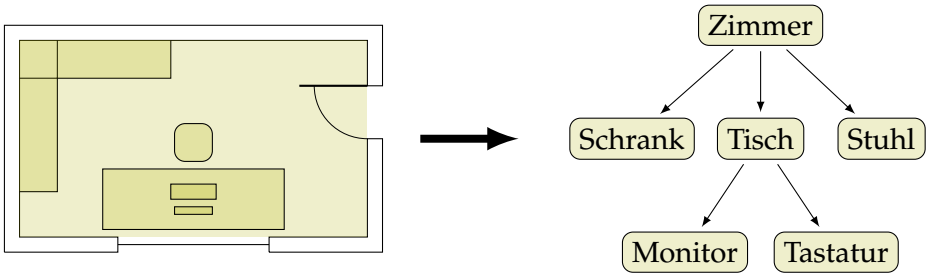
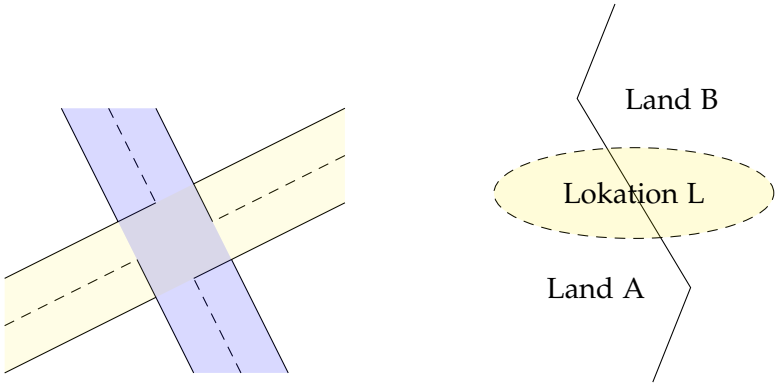


Abbildung 4.1.: Beispiel einer Baummodellierung

Ein Baummodell ermöglicht auf natürliche und effiziente Weise die Berechnung der Inklusion, sowie der inversen Inklusion. Nachbarschaftsbeziehungen, Mengenoperationen $\cap$ und $\cup$ oder Berechnung der Distanz zwischen zwei Lokationen werden hingegen nicht unterstützt. Ein essentieller Nachteil des Baummodells besteht in dessen starren Beziehungen zwischen einzelnen Lokationen. Die somit erzeugte Einschränkung der Modellierungsfreiheit führt zu einer nur bedingten Fähigkeit des Lokationsmodells reale Welt zu reflektieren. Abbildung 4.2 verdeutlicht den Sachverhalt: Für die abgebildete Straßenkreuzung existiert im Baummodell keine Möglichkeit diese als Schnitt zweier Straßen zu modellieren (a). Auch kann eine über mehrere Lokationen (A, B) erstreckende Lokation (L) nicht modelliert werden. Als Ausweg bieten sich mehrere Möglichkeiten an, welche jedoch mit Redundanz oder Informationsverlust einhergehen.



(a) Schnitt zweier Objekte

(b) Überlappung über mehrere Lokationen hinweg

Abbildung 4.2.: Schnitt, Überlappung: Im Baummodell nicht möglich

4.2.3. Hierarchisches Gittermodell

Das Gittermodell [BD05] ist eine Erweiterung des Baummodells, welches die Modellierung von den sich partiell überlappenden Lokationen erlaubt und damit die zentrale Schwäche des Baummodells ausgleicht. Ein Gittermodel M ist weiterhin ein aus einer Menge von Knoten und Kanten bestehendes Zweiertupel (V, E), welches die Bedingung

$$\begin{aligned}&\forall l_i, l_j \in V : (l_i \subset l_j) \Leftrightarrow \\ &\exists l_0, l_1, \ldots l_{n-1} : (l_j, l_0) \in E \wedge (l_{n-1}, l_i) \in E \wedge \forall k \in \{0, \ldots n-1\} : (l_k, l_{k+1}) \in E\end{aligned} \tag{4.5}$$

erfüllt. Diese drückt aus, dass jede Kante, gleich dem Baummodell, eine Inklusionsbeziehung zweier Lokationen darstellt. Hierdurch impliziert ist zugleich die Zyklusfreiheit der Kantenmenge E. Anders als im Baummodell kann ein Knoten hier mehr als nur eine Eingangskante besitzen, wodurch sich teilweise überlappende Lokationen modelliert werden können. Abbildung 4.3 verdeutlicht dies an einem Beispiel.

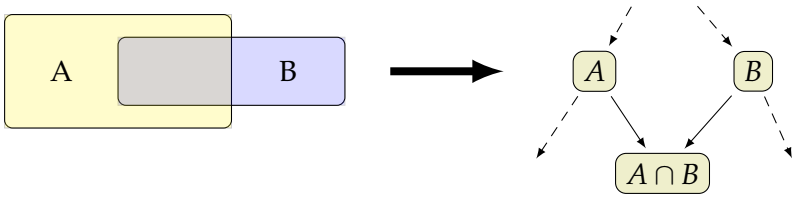


Abbildung 4.3.: Modellierung eines Schnittes im Gittermodell

Das Gittermodell ermöglicht ähnlich dem Baummodell eine effiziente Berechnung der Inklusionsfunktion. Berechnung der inversen Inklusion ist nun aufgrund der Erweiterung des Modells ineffizienter, da auf dem Weg von einem Knoten zur Wurzel eventuell

mehrere Pfade traversiert werden müssen. Weiterhin unterstützt das Modell keine Nachbarschaftsbeziehungen, Vereinigung von Lokationen sowie Berechnung der Distanzfunktion. Abbildung 4.4 stellt den damit verbundenen Nachteil des Modells dar. Für drei abgebildete Lokationen ist es die Feststellung, ob Lokation C sich vollständig in A und B befindet, oder auch Bereiche außerhalb dieser überdeckt nicht möglich.

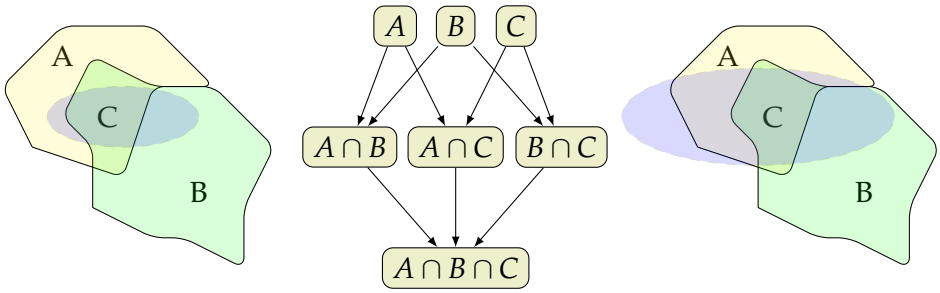


Abbildung 4.4.: Gleiches Gittermodell für unterschiedliche Lokationen

4.2.4. Graphen-basiertes Lokationsmodell

Anders als hierarchische Modelle (Baum- und Gittermodell) setzen Graphen-basierte [Dro03] Lokationsmodelle ihren Schwerpunkt auf Modellierung von Verbindungen und Entfernungen zwischen einzelnen Lokationen. Wie auch in beiden vorigen Modellen besteht ein Graphenmodell M aus einem Zweiertupel (V, E), wobei Knotenmenge V (disjunkte) Lokationen darstellt. Kantenmenge E repräsentiert anders als in hierarchischen Modellen keine Inklusionsbeziehungen, stattdessen Verbindungen zwischen Lokationen. Diese können von einer direkten Angrenzung (Tür zwischen zwei Räumen) bis zu einer nicht weiter modellierten Lokation größerer Ausdehnung reichen (Straßenverbindung zwischen zwei Städten). Im Allgemeinen sind Verbindungen zwischen zwei Lokationen symmetrisch, dies ist jedoch nicht zwingend. In einem solchen Fall ist die Kantenmenge E asymmetrisch und der Gesamtgraph (V, E) somit gerichtet. Durch verschiedene Arten der Kanten- und Knoten-Gewichtung kann zwischen Modellierungsaufwand und Modellgenauigkeit nach Bedarf abgewogen werden. Im Folgenden werden vier Möglichkeiten der Gewichtung und die für das Modell hieraus resultierenden Konsequenzen diskutiert:

1. *Uniforme Kantengewichtung.* Alle Kanten des Graphen erhalten gleiche Gewichtung, wodurch der Modellierungsaufwand minimiert wird. Die Berechnung der Distanz kann in linearer Zeit erfolgen [AMOT90], eignet sich jedoch nur bedingt zur praktischen Interpretation, wie Abbildung 4.5 dies veranschaulicht. Entfernung zwischen Raum A und D ist hiernach über Flur E geringer als durch Räume B und C.
2. *Allgemeine Kantengewichtung.* Allgemeine Kantengewichtung erlaubt beliebige (positive) Gewichte für Verbindungen zwischen zwei Lokationen zu vergeben. Mit der Gewichtung der Kanten können neben der geometrischen Entfernung zwischen zwei Lokationen andere bezüglich der für Überwindung der Entfernung aufzuwendenden Kosten relevante Faktoren berücksichtigt werden. Für die Praxis bedeutend kann

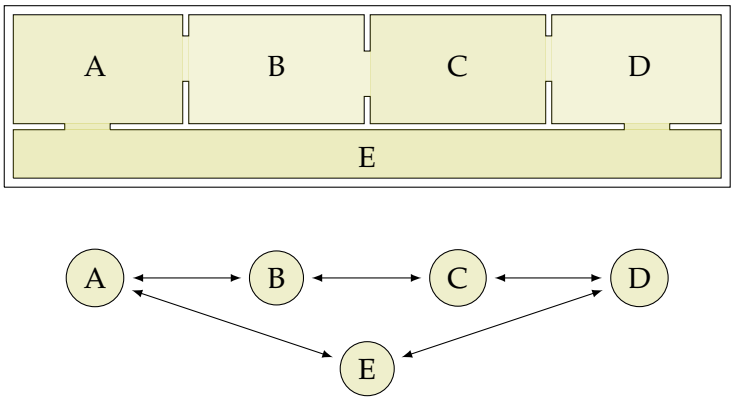


Abbildung 4.5.: Uniforme Kantengewichtung

beispielsweise die benötigte Zeit sein. Des Weiteren kann die Kantengewichtung zeitlich variieren, sowie für unüberwindbare Hindernisse (Schließung einer Straße) auf unendlich gesetzt werden. Mit Hilfe von Dijkstra kann die Entfernung in $O(n \log n)$ Zeit berechnet werden.

3. *Einfache Knotengewichtung.* Die Problematik der besprochenen Kantengewichtung ist die Nichtberücksichtigung der Ausdehnung der durch Knoten des Graphen modellierten Lokationen. Diese kann jedoch entscheidend sein. Gewichtung der Knoten geht diesem Problem nach. Jeder Knoten erhält ein Gewicht, welches seine, wiederum nach verschiedenen Kriterien berechenbare, räumliche Ausdehnung repräsentiert. In Kombination mit Kantengewichten kann die Berechnung der Entfernung nun auf präzisere Weise erfolgen.

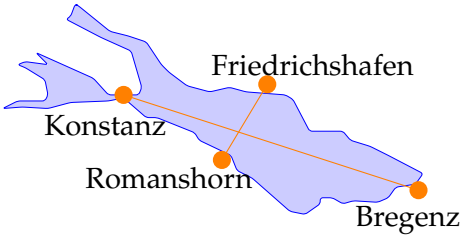


Abbildung 4.6.: Unterschiedlicher Aufwand zum Überqueren des Bodensee

Wesentlicher Nachteil dieser Methode besteht in der allgemeinen Ungenauigkeit der Aussage bezüglich des tatsächlichen Aufwandes, welcher zum Passieren der Lokation von einem Grenzpunkt zum anderen, betrieben werden muss. Abbildung 4.6 veranschaulicht die angesprochene Problematik. Das Überqueren des Bodensee von Konstanz nach Bregenz impliziert einen enorm höheren Aufwand als dessen Überquerung beispielsweise von Friedrichshafen nach Romanshorn. Einfache Knotengewichtung spiegelt diesen Sachverhalt indessen nicht wider.

4. *Mehrfache Knotengewichtung.* Mehrfache Knotengewichtung löst das im letzten Punkt angesprochene Problem. Für jeden Knoten wird pro Paar bestehend aus eingehender

und ausgehender Kante ein spezifisches Gewicht definiert. Entfernungen zwischen Lokationen können nun auf eine sehr genaue Weise berechnet werden. Der Modellierungsaufwand erhöht sich hierdurch enorm. Sei $l_i \in V$ ein Knoten mit Eingangsgrad e_i und Ausgangsgrad a_i. Die Anzahl der für Knoten l_i zu definierenden Gewichte beträgt $e_i(e_i - 1)/2$ im symmetrischen und $e_i \cdot a_i$ im asymmetrischen Fall.

Der Ausgangspunkt der bisher betrachteten Graphenmodelle ist die Exklusivität der durch Knoten modellierten Lokationen. Sich überschneidende oder enthaltenden Lokationen wurden nicht betrachtet. Prinzipiell können nicht disjunkte Lokationen in Graphenmodellen zugelassen werden, dies jedoch ist im Hinblick auf sinnvolle Definition der Kantengewichte problematisch. Im Gegensatz zu hierarchischen Modellen unterstützen Graphenmodelle die Berechnung der Funktionen *nearest* und *nav*. Auch eine einfache, qualitative, Visualisierung ist aufgrund von Gewichtung der Kanten sowie Knoten möglich. Inklusion, inverse Inklusion sowie Mengenoperationen werden hingegen (auch im Fall der nicht exklusiven Lokationen) nicht unterstützt. Im nächsten Abschnitt soll die Möglichkeit untersucht werden durch Kombination des Graphen- und des Gittermodells die Vorteile beider zu vereinen.

4.2.5. Kombination des Graphen-basierten und des Gittermodells

Wie im letzten Abschnitt gezeigt, liegt der zentrale Fokus Graphen-basierter Modelle auf der Modellierung von Entfernungen so wie auch der Lokationsgrößen. Hierarchische Lokationsmodellierung (Abschnitt 4.2.2 und 4.2.3) hingegen erlaubt eine effiziente Berechnung der Inklusion. Es liegt somit auf der Hand beide Modelle zu vereinen, um sowohl Entfernungen als auch Hierarchien zwischen einzelnen Lokationen modellieren zu können. Das aus Graphen- und Gittermodell kombiniertes Lokationsmodell enthält zwei Arten von Kanten. Die eine räumliche Inklusion implizieren *Hierarchiekanten* sowie *Verbindungskanten*, welche (gewichtete) Verbindungen zwischen Lokationen darstellen. Abbildung 4.7 zeigt beispielhaft wie eine Modellierung zweier Gebäude bis hin auf Raumebene mittels des kombinierten Modells erreicht werden kann. Es ist zu berücksichtigen, dass zwei Lokationen immer nur durch eine der beiden Kantenarten miteinander verbunden sein können. Des Weiteren können Lokationen nicht nur innerhalb einer Hierarchieebene, sondern über diese beliebig hinweg durch Verbindungskanten miteinander verknüpft werden können (z.B. Verbindung zweier Gebäude durch einen Gang.)

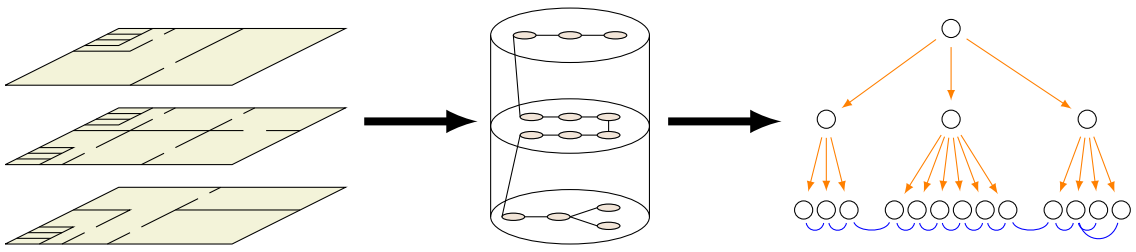


Abbildung 4.7.: Kombination des Gitter- und Graphenmodells

Hierarchischer Bestandteil des Lokationsmodells (Inklusionskanten) kann weiterhin zur Schnittmodellierung, Berechnung der Inklusion sowie der inversen Inklusion verwendet werden. Hingegen werden Verbindungskanten zu Zwecken der Navigation sowie Bestimmung des nächsten Nachbarn verwendet.

Durch zusätzliche Geometriemodellierung in den Knoten des Modells könnte die Abstandfunktion günstig berechnet sowie angemessene Visualisierung der Lokationen erfolgen. Diese, *hybride*, Art der Modellierung ist Thema des nächsten Abschnittes 4.3.

4.3. Hybride Lokationsmodelle

Wesentlicher Nachteil einfacher geometrischer Modelle, wie im Abschnitt 4.1 beschrieben, besteht in fehlenden Beziehungen zwischen Lokationen. Anders als im Fall der vorgestellten symbolischen Lokationsmodelle können viele Beziehungen wie Inklusion, Überschneidung oder Nachbarschaft erst durch teure geometrische Berechnungen ermittelt werden. Symbolische Lokationsmodelle eignen sich hingegen kaum zur Berechnung quantitativer Werte wie Distanz oder Größe. In diesem Abschnitt stellen wir ein hybrides Lokationsmodell aus [DR03] vor, welches Eigenschaften sowohl geometrischer als auch symbolischer Lokationsmodelle in sich vereint.

4.3.1. Location Model for Fine Grained Geocast

Das in [DR03] vorgestellte Lokationsmodell, im Folgenden als *LMFGG* abgekürzt, dient in erster Linie zur Unterstützung von GeoCasting [NI97b, KV03]. Eine der wesentlichen Funktionalitäten des Lokationsmodells besteht in der Berechnung der Wahrscheinlichkeit $p(t, c) \rightarrow [0, 1]$, mit der sich ein Subjekt c in der Ziellokation t einer zu versendenden Nachricht befindent. Das Modell in seinem Aufbau ist dem Gittermodells ähnlich (vgl. Abbildung 4.8). Die Knoten erhalten je eine Bezeichnung, wodurch eine Lokation über die Konkatenation der Bezeichnungen entlang eines Pfades global eindeutig referenziert werden kann. Ein Knoten repräsentiert, so wie in den bisher besprochenen Modellen, eine Lokation, wobei diese in LMFGG neben der Bezeichnung zusätzlich geometrisch modelliert werden kann. Geometrische Modellierung erfolgt mittels der Angabe eines die Lokation repräsentierenden Polygonzuges (2-dimensional) mit einer optionalen Höhenangabe (2.5-dimensional). Für das verwendete Koordinatensystem exisiteren grundsätzlich zwei Möglichkeiten:

- Verwendung des bereits in Elternknoten verwendeten Koordinatensystems, wobei die obersten geometrisch modellierten Knoten (Städte, Länder) ein globales Referenzsystem wie das WGS84 verwenden.
- Definition eines neuen lokalen Koordinatensystems, welches sich auf die Kinderknoten vererbt.

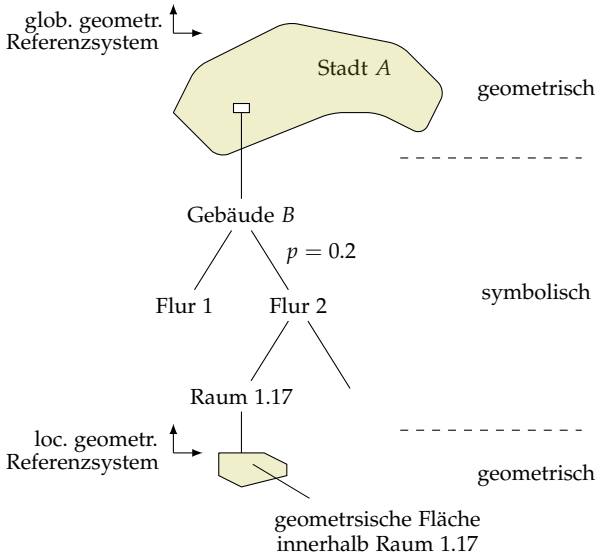


Abbildung 4.8.: Hybride Lokationsmodellierung im LMFGG

Abbildug 4.8 veranschaulicht das Modell an einem Beispiel. Zu erkennen ist die teilweise geometrische Modellierung der Lokationen, wobei auf der oberen Ebene ein globales (z.B. WGS84) auf den unteren Ebenen ein lokales Koordinatensystem verwendet wird. Den Kanten des Modells wird je eine Wahrscheinlichkeit zugeordnet, welche zur Berechnung der Funktion $p(t, c)$ verwendet wird. In unserem Beispiel drückt diese aus, dass eine Person welche sich im Gebäude B aufhält, sich zugleich mit einer Wahrscheinlichkeit von 0.2 im Flur 2 befindet. Im Fall, wenn beide Lokationen (einer Kante) geometrisch modelliert sind, kann die Wahrscheinlichkeit auch explizit berechnet werden.

Hybride Lokationsmodellierung im LMFGG dient in erster Linie der Vereinfachung der Modellerstellung, da nicht alle Lokationen geometrisch modellieren werden müssen. Auf der anderen Seite können mit Hilfe des Modells bei einer an eine geometrsiche Lokation adressierte Nachricht auch die jenigen Empfänger ermittelt werden, bei denen lediglich die symbolische Lokation bekannt ist. So wie anderere, bereits vorgestellte Baum-basierte Lokationsmodelle unterstützt LMFGG die Berechnung der Funktionen *incl* und *enclosedBy*. Andere, nicht auf Inklusionskanten eines Baumes basierende Funktionen wie *dist* oder *nearest* werden hingegen nicht unterstützt. Ausführlicherer Vergleich der Modelle erfolgt im nächsten Abschnitt 4.4.

4.4. Analyse

Funktionalität der vorgestellten Lokationsmodelle wurde in jeweiligen Abschnitten dieses Kapitels bereits diskutiert. Hier stellen wir zusammenfassend die Erfüllung der im Kapitell 3 diskutierten Anforderungen bzgl. der Funktionalität und Effizienz seitens der Lokationsmodelle dar.

Modell	$loc(o_i)$	$dist(l_i, l_j)$	$nearest(l_i, t_j)$	Ergänzung
gefordert	$O(1)$	$O(1)$	$O(n \log n)$	n: # Objekte
Mengen-basiert	($O(1)$)	($O(2^n)$)	$O(2^n)$	n: # Objekte
Baummodell	($O(1)$)	–	–	
Gittermodell	($O(1)$)	–	–	
Graph	($O(1)$)	–	$O(n \log n)$	n: # Objekte
Gitter + Graph	($O(1)$)	–	$O(n \log n)$	n: # Objekte
LMFGG	($O(1)$)	–	–	

Modell	$incl(l_i, t_j)$	Ergänzung	$nav(l_i, l_j)$	Ergänzung
gefordert	$O(m \log n)$	n: # Objekte in l_i	$O(n \log n)$	
Mengen-basiert	$O(2^n)$	n: # Objekte	–	
Baummodell	$O(n)$	n: # Objekte in l_i	–	
Gittermodell	$O(n)$	n: # Objekte in l_i	–	
Graph	–		$O(n \log n)$	n: # Objekte
Gitter + Graph	$O(n)$	n: # Objekte in l_i	$O(n \log n)$	n: # Objekte
LMFGG	$O(n)$	n: # Objekte in l_i	–	

Modell	$enclosedBy(l_i, t_j)$	Ergänzung	$\cap$ / $\cup$	Ergänzung
gefordert	$O(\log n)$	n: # Objekte	$O(1)$	
Mengen-basiert	$O(2^n)$	n: # Objekte	$O(n^2)$	n: # Elementarlokationen
Baummodell	$O(\log n)$	n: # Objekte	–	
Gittermodell	$O(\log n)$	n: # Objekte	–	
Graph	–		–	
Gitter + Graph	$O(\log n)$	n: # Objekte	–	
LMFGG	$O(\log n)$		–	

Modell	$\subset, \subseteq$	Ergänzung	Metalokationen Visualisierung Orientierung
gefordert	$O(1)$		✓✓✓
Mengen-basiert	$O(n^2)$	n: # Objekte in Mengen	✓– –
Baummodell	$O(\log n)$	n: # Objekte	– (✓) –
Gittermodell	$O(\log n)$	n: # Objekte	– (✓) –
Graph	–		– ✓–
Baum + Graph	$O(\log n)$	n: # Objekte	– ✓–
LMFGG	$O(\log n)$	n: # Objekte	– (✓) –

Tabelle 4.1.: Laufzeiten zur Berechnung geforderter Funktionen

Hervorzuheben ist, dass keines der in diesem Kapitel vorgestellten Lokationsmodelle das in dieser Arbeit zugrunde gelegte Systemmodell (vgl. Kapitel 2) unterstützt. So differenziert keines der Modelle zwischen Objekten und Lokationen, wodurch die Funktion *loc* überflüssig wird. Gravierender ist die fehlende Unterstützung der Objekttypisierung, welche im Bereich der PerFlows einen hohen Stellenwert besitzt. Unterstützung vieler der

im Kapitel 3 diskutierten Anforderungen kann somit in häufigen Fällen nur als bedingt gegeben betrachtet werden.

Tabelle 4.1 fasst die von Lokationsmodellen erbrachte Funktionalität sowie, falls unterstützt, die bezüglich der Laufzeitkomplexität erbrachte Effizienz der Funktionsberechnung zusammen. Wie zu erkennen, kann die geforderte Komplexität in nur seltenen Fällen eingehalten werden. Fehlende Unterstützung der geforderten Funktionalität, hohe Laufzeitkomplexität, sowie die erwähnte fehlende Objekttypisierung motivieren uns zur Konzeption eines eigenen Lokationsmodells, welches die sowohl die geforderte Funktionalität als auch Effizienz bietet und somit im Bereich der PerFlows eingesetzt werden kann.

Kapitel 5

SmartGPS – Lokationsmodel für PerFlows

5.1. Grundlagen

Die im letzten Kapitel vorgestellten Lokationsmodelle können die aus dem Bereich der PerFlows motivierten Anforderungen nur bedingt, bzw. nur mit hoher Laufzeit erfüllen. Fehlende Unterstützung des vorgestellten Systemmodells, insbesondere die der Objekttypisierung sowie eine im Folgenden als notwendig dargelegte Unterscheidung zwischen dynamischen und statischen Objekten, erfordern die Konzeption eines eigenen Lokationsmodells. Als Lösungsansatz wird in diesem Kapitel das Lokationsmodel *SmartGPS* vorgestellt. Wie die Bezeichnung bereits suggeriert, ist SmartGPS ein geometrisches Lokationsmodell, dessen Grundlage das WGS84-Koordinatensystem ist. Neben den geometrischen Koordinaten finden zahlreiche Elemente der symbolischen Modelle Einzug in das Lokationsmodell. Auf diese Weise kann, wie im Abschnitt 5.2 gezeigt, die Funktionalität trotz Geometriemodellierung häufig auf eine sehr günstige Weise geboten werden.

Eine weitere Charakteristik des SmartGPS–Lokationsmodells ist die explizite Unterscheidung zwischen statischen und dynamischen Objekten (vgl. Kapitell 2). Entsprechend teilt sich das Modell in zwei größere Bereiche auf: Baumartige Modellierung statischer Objekte (Kapitel 5.2) und Modellierung dynamischer Objekte mittels der im Kapitel 5.3 besprochenen Sensorfusion. Die Motivation dynamische und statische Objekte gesondert zu betrachten ist durch folgende drei Punkte zu begründen:

1. *Unterschiedliche Art der Lokationserfassung.* Zur Erfassung der Lokationen statischer Objekte reicht einmaliges Vermessen und Abspeichern in der Regel aus. Lokationserfassung dynamischer Objekte hingegen erfordert deren permanente Verfolgung mittels Sensoren, welche wiederum Fehler bzw. Störungen (weißes Rauschen) in Lokationsdaten einbringen kann. Infolgedessen kann eine Filterung, gegebenenfalls Fusion mehrerer Datenströme erforderlich sein. Diese Aspekte spielen jedoch keine Rolle in der Lokationserfassung statischer Objekte.

2. *Unterschiedliche räumliche Ausdehnung.* In der Domäne der PerFlows sind dynamische Objekte (Personen, Automobile) in Relation zu statischen (Räume, Gebäude, Städte) meist klein. Die Tatsche motiviert uns Lokationen der ersteren als Punkte, die der zweiten als Flächen bzw. Körper zu modellieren.
3. *Lokationsänderungen dynamischer Objekte.* Aufgrund des im Allgemeinen häufigen Lokationswechsels dynamischer Objekte (fahrende Autos, Personen) erscheint eine Lokationsmodellierung und -speicherung dieser in einer gemeinsam mit statischen Objekten benutzten Datenstruktur, wie beispielsweise in [DR03], als ungeeignet. Immer währende Positionsaktualisierungen der Objekte innerhalb des Lokationsmodells würden in Bezug auf die Anzahl gestellter Lokationsanfragen einen zu hohen Aufwand bedeuten.

Im Folgenden stellen wir das Lokationsmodell SmartGPS im Detail vor. Zunächst wird auf Lokationsmodellierung statischer Objekte, im Kapitel 5.3 anschließend auf die der dynamischen eingegangen. Kapitel 5.4 befasst sich mit der Möglichkeit eines praktischen Einsatzes des Modells.

5.2. Lokationsmodellierung statischer Objekte

5.2.1. Motivation

Betrachten wir die politische Weltkarte, stellen wir eine (meist) strikt hierarchische Aufteilung des Planeten fest. Fast jeder Bereich der Erdoberfläche ist fest der einen oder anderen Autorität zugeordnet. Die hierarchische Lokationsaufteilung setzt sich auch im Kleinen fort. Die Bundesrepublik beispielsweise teilt sich disjunkt in 16 Bundesländer auf, diese organisieren sich wiederum überschneidungsfrei in Bezirke, Kreise, Kommunen, Städte, ... und so weiter bis hin zu Gebäude- und Raumebene. Ausnahmen dieses Ordnungsprinzips bilden auf der anthropologischen Ebene typischerweise Straßen oder Eisenbahnverbindungen. Auf der Seite der Natur greifen häufig Flüsse, Gebirgszüge oder Waldgebiete ineinander über, bzw. ziehen sich über die von Menschen gesetzten Grenzen hinweg. Die in weitem Ausmaß vorzufindende hierarchische Organisation der Lokationsaufteilung motiviert die Struktur des vorgestellten Modells. SmartGPS sieht einen zentralen hierarchischen Baum sowie diverse weitere, kleinere Bäume, welche eine jeweils erweiterte Sicht auf ein bestehendes Gebiet repräsentieren, vor. Im Folgenden betrachten wir die Architektur der SmartGPS-Modellierung detailliert. Wir schauen zunächst die Struktur der Bäume an, diskutieren anschließend auf welche Weise diese miteinander verknüpft und zur Berechnung der im Kapitel 3 geforderten Funktionalität eingesetzt werden können.

5.2.2. Konzept

Die Hauptidee der Modellierung statischer Objekte in SmartGPS besteht darin ein gegebenes Gebiet disjunkt (nicht notwendigerweise vollständig) dem eben beschriebenen

hierarchischen Aufteilungsprozess folgend in Teilgebiete aufzuteilen und ihre Lokationen zu modellieren. Der Prozess setzt sich für die erhaltenen Teilgebiete rekursiv fort, wodurch wir an dessen Ende eine hierarchische (beispielsweise bis auf Gebäudeebene reichende) Baumstruktur erhalten. Jeder Knoten des Baumes ist einer festen Autorität zugeordnet (beispielsweise Bundesregierung, Stadtverwaltung, ...), wodurch der Modellierungsaufwand an einer Stelle sich lediglich auf Modellierung der direkten Teilgebiete beschränkt. Aufgabe zur Modellierung hierarchisch untergeordneten Teilgebiete kann entsprechend an die unterstehenden Autoritäten delegiert werden.

Neben dem eben beschriebenen hierarchisch organisierten *Autoritätsbaum* berücksichtigt SmartGPS weitere (wiederum hierarchische) Lokationsaufteilungen. Eine solche Lokationsaufteilung kann als eine erweiterte Sicht auf ein bestehendes Gebiet betrachtet werden. So lässt sich beispielsweise die Bundesrepublik neben der Möglichkeit zur Aufteilung nach Bundesländern auch nach Naturschutzgebieten oder Autobahnen (nicht vollständig aber disjunkt) aufteilen. Auch in diesen zwei Fällen ist das Ergebnis eine Baumstruktur, welche nun im Folgenden als *Erweiterungsbaum* bezeichnet wird.

Während der Autoritätsbaum eine große Tiefe besitzt (von Staats- bis Gebäudeebene), repräsentiert ein Erweiterungsbaum im Allgemeinen eine eingeschränkte Region, diese auch bis zu einer beschränkter Granularität. Ein Erweiterungsbaum erfüllt somit immer eine spezielle Aufgabe, welche in der Modellierung eines durch den Autoritätsbaum oder andere Erweiterungsbäume nicht abgedeckten Gebietes besteht. Ein Erweiterungsbaum besitzt einen festdefinierten *Einstiegspunkt*, sowie, im Allgemeinen, mehrere *Ausstiegspunkte*. Der Einstiegspunkt eines Erweiterungsbaumes ist der unterste Knoten des Autoritätsbaumes, welcher die durch den Erweiterungsbaum modellierte Lokation(en) vollständig enthält. Dieser Knoten ist eindeutig. Somit besitzt ein Erweiterungsbaum im Allgemeinen keine eigene Wurzel, diese Funktion übernimmt vielmehr der Einstiegspunkt im Autoritätsbaum. Ausstiegspunkte eines Erweiterungsbaumes sind ebenso Knoten im Autoritätsbaum, welche vollständig in einem der Knoten des Erweiterungsbaumes enthalten sind. Während ein Knoten des Erweiterungsbaumes mehrere Ausstiegspunkte haben kann, besitzt ein Knoten im Autoritätsbaum höchstens einen übergeordneten Knoten eines Erweiterungsbaumes, für den er der Ausstiegspunkt ist. Abbildung 5.1 veranschaulicht das Konzept. Ein dreigeschossiges Gebäude wird im Autoritätsbaum zunächst nach Geschossen und anschließend nach Räumen aufgeteilt. Ein Erweiterungsbaum teilt das Gebäude hingegen erst in Flügel A, B, anschließend in Räume auf. Rot markiert sind die Ein- und Ausstiegspunkte des Erweiterungsbaumes im Autoritätsbaum.

Bei der Kombination des Autoritätsbaumes mit einem Erweiterungsbaum lassen sich neben reinen Inklusionsbeziehungen auch partielle Lokationsschnitte modellieren. Jeweils ein Knoten aus dem Autoritäts- und einem Erweiterungsbaum können einen Schnitt bilden, wodurch sich eine allgemeine Gitterstruktur ergibt. In der Abbildung 5.1 bilden Konten B und F_3 einen Schnitt, dessen Resultat in diesem Fall wiederum einen Knoten des Erweiterungsbaumes bildet.

Knoten des (Autoritäts- oder Erweiterungs-) Baumes repräsentiert ein bestimmtes statisches Objekt, welches eine feste Zuordnung zu einer bestimmten Lokation aufweist. Modellierung der Lokation eines solchen Objektes erfolgt mittels geometrischer Koordinaten des WGS84-System. Viele der für das Lokationsmodell relevanten Objekte lassen sich

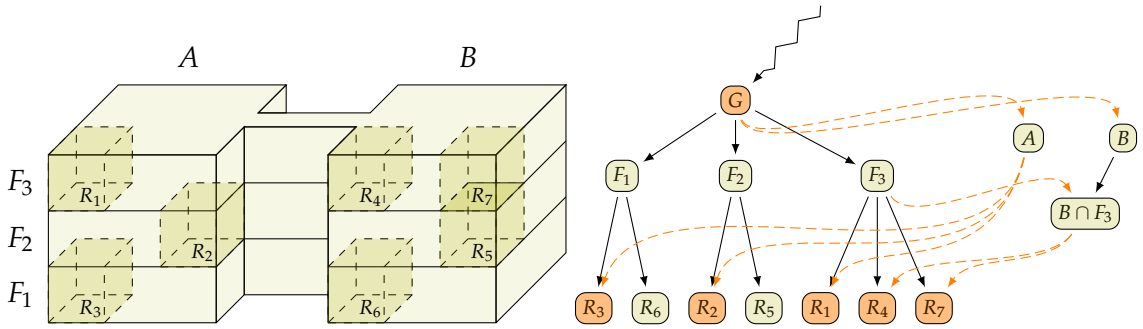


Abbildung 5.1.: Lokationsmodellierung eines Gebäude im Autoritäts- und Erweiterungsbaum

als zweidimensionale Lokationen modellieren. Als Beispiel seien Städte, Seen und Straßen genannt. Andere Objekte lassen sich wiederum als 2.5-dimensionale Objekte [DR03] darstellen. Hierzu gehören vor allem Gebäude und Räume, welche sich meist durch eine Angabe der überdeckten Fläche und der Höhe beschreiben lassen. Zur Modellierung der Lokation dieser Objekte reicht also die Angabe einer aus Paaren der Form (Längengrad, Breitengrad) bestehenden Liste (Polygonzug), welche den zweidimensionalen Umriss des Objektes angeben, mit einer eventuell zusätzlichen (minimalen und maximalen) Höhenangabe aus. Lokation der Universität Stuttgart könnte in einem entsprechenden Knoten beispielsweise grob wie folgt modelliert werden:

```
<location>
  <vertexlist>
    48.45.00, 9.06.55, 48.45.05, 9.06.05, 48.44.44, 9.05.59, 48.44.31, 9.05.18, 48.44.15, 9.05.48
  <\vertexlist>
<\location>
```

Objekte, wie Straßen, Flüsse, Eisenbahnschienen lassen sich hingegen besser durch eine den Verlauf angebende Punktfolge mit einer zusätzlichen Angabe der entsprechenden Breite modellieren. Auch diese Modellierung wird dem das Objekt repräsentierenden Knoten im Autoritätsbaum, bzw. einem Erweiterungsbaum zugeordnet.

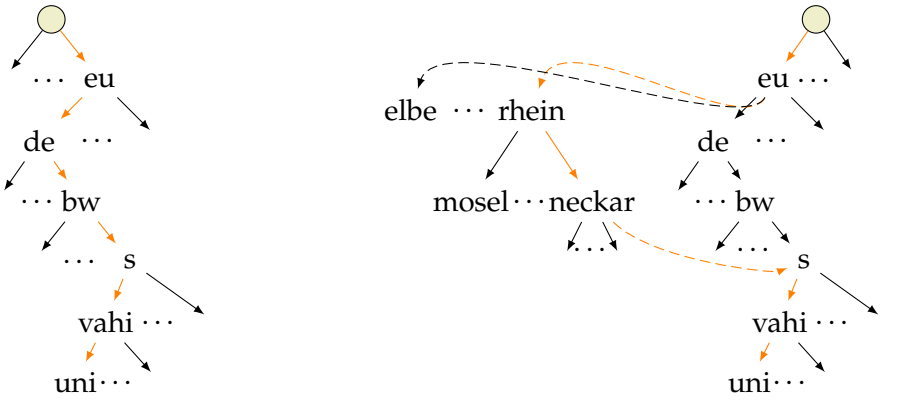
Auf Grund der im Kapitel 4 bereits diskutierten Nachteilen rein hierarchischer Lokationsmodelle sieht SmartGPS neben der durch Inklusionskanten gegebenen hierarchischen Baumstruktur eine weitere Art der Kanten vor, welche Verbindungen zwischen je zwei Lokationen darstellen. Ergebnis der Erweiterung ist eine der im Abschnitt 4.2.5 beschriebenen Kombination aus Graphen- und Gittermodell ähnliche Datenstruktur. Verbindungskanten erhalten ähnlich dem Graphenmodell jeweils ein Gewicht, welches ihre metrische Länge angibt. In einem feingranulierten Modell sind die Kantengewichte idealerweise immer gleich Null (eine Tür). Des Weiteren erhält eine jede Verbindungskante einen Typ, sowie zwei GPS-Angaben, welche die Bereiche angeben, an denen die Kante an der jeweiligen Lokation ansetzt. Der Typ einer Kante kann entsprechend der für den praktischen Einsatz relevanten Klassifikation der Begehbarkeit gewählt werden.

5.2.3. Funktionalität

Im Folgenden zeigen wir, wie mit Hilfe des Lokationsmodells SmartGPS die im Kapitel 3 geforderte Funktionalität implementiert werden kann. Hierzu gehen wir auf die Funktionen einzeln ein und betrachten detailliert die hinsichtlich dieser benötigten Eigenschaften des Modells, erweitern es gegebenenfalls bei Bedarf.

Position

Zu einem gegebenen statischen Objekt ist dessen Lokation gesucht. Sofern der Zugriff auf den das Objekt repräsentierenden Knoten besteht, ist die Aufgabe trivial, da Lokationsbeschreibung ein Teil des Knoteninhalts darstellt, welcher in diesem Fall just ausgelesen werden muss. Soll ein Knoten global referenziert werden können, erweitern wir das Modell für jeden Knoten um ein Bezeichnungsfeld, welcher unter den Geschwisterknoten derselben Ebene eindeutig ist. Mit der eigeführten Bezeichnung lässt sich ein Knoten ähnlich dem im Abschnitt 4.3.1 vorgestellten Lokationsmodell für GeoCast global eindeutig adressieren. Eine Adresse wird aus Konkatenation der Bezeichnungen der auf einem Pfad liegenden Knoten gebildet, welche in Form einer URI-ähnlicher Notation angegeben werden kann (vgl. Abbildung 5.2 (a)). Es ist zu beachten, dass ein Knoten mit Verwendung von Erweiterungsbäumen mehrere Adressen besitzen kann, eine Adresse jedoch immer eindeutig einen Knoten referenziert. Um den Übergang aus dem (globalen) Autoritätsbaum auf einen Erweiterungsbaum in der Adressierung zu markieren, kann das Symbol "//" gefolgt von der eindeutigen Bezeichnung des Erweiterungsbaumes verwendet werden. Abbildung 5.2 (b) stellt beispielhaft die Adressierung der Universität Stuttgart über den die Einzugsgebiete deutscher Flüsse modellierenden Erweiterungsbaum dar.



eu/de/bw/s/vahi/uni

(a) Adressierung im Autoritätsbaum

eu//flussBaum/rhein/neckar/s/vahi/uni

(b) Adressierung mit Erweiterungsbaum

Abbildung 5.2.: Objektadressierung in SmartGPS

Mit Hilfe der Adressierung lassen sich Positionen der Objekte im Autoritätsbaum oder einem Erweiterungsbaum enorm schnell nachschlagen. Die Laufzeit ist logarithmisch in Anzahl der modellierten Objekte bzw. linear in der Länge der Adresse begrenzt. In der Praxis wird der Autoritätsbaum die Tiefe von zehn kaum überschreiten. Das Resultat einer Positionsanfrage für ein statisches Objekt besteht aus dem zuvor im Abschnitt 5.2.2 beschriebenen Polygonzug mit einer optionalen Höhenangabe.

Distanz

Aufgrund der geometrischen Modellierung der Lokationen, kann die Distanz zwischen zwei Objekten unter Angabe entsprechender GPS-Punkte im Fall von SmartGPS einfach berechnet werden. Unter Distanz wird Länge der sich zwischen zwei Punkten A, B befindenden *Orthodrome* verstanden (vgl. Anhang A.2) und kann wie folgt (in Kilometern) berechnet werden:

$$d = \arccos(\sin\phi_A \cdot \sin\phi_B + \cos\phi_A \cdot \cos\phi_B \cdot (\cos\lambda_B - \cos\lambda_A)) \cdot \frac{\pi}{180} \cdot 6370 \tag{5.1}$$

wobei ϕ_A, ϕ_B Breitengrade und λ_A, λ_B Längengrade der beiden Punkte darstellen. Die auf diese Art berechnete Distanz spiegelt zwar in seltenen Fällen die tatsächliche zum Erreichen einer Lokation zurückzulegende Entfernung wider, nichtsdestotrotz ist die Funktionalität sinnvoll und kann als Schätzwert verwendet werden. Berechnung einer begehbaren Entfernung zwischen zwei Lokationen behandelt Abschnitt *Navigation* weiter unten.

Nächster Nachbar

Die Berechnung des nächsten Nachbarn kann (konventionell) mittels des Dijkstra-Algorithmus ermittelt werden. Mit Hilfe der im Abschnitt 5.2.2 besprochenen Typisierung der Verbindungskanten sowie der Lokationen kann die Berechnung selektiv unter Berücksichtigung nur bestimmter Kanten- sowie Knotentypen ablaufen. Die Definition der Begehbarkeit einer Verbindung kann somit in Abhängigkeit von Nutzeranforderungen variieren. Um die Suche im Fall des Nichtvorhandenseins eines Objektes des gewünschten Typs zu begrenzen, kann die Berechnung des Nächsten Nachbarn unter Angabe einer zusätzlichen maximalen Entfernung vollzogen werden. Die Dijkstra-Suche bricht somit beim Erreichen des ersten Knoten mit dem gewünschten Typ oder Überschreiten der maximalen Entfernung ab. Der Aufwand der Suche beschränkt sich somit im schlimmsten Fall auf $O(n \log n)$ Schritte, wobei n die Anzahl der Objekte darstellt, welche sich innerhalb des vorgegebenen maximalen Radius, bzw. innerhalb des Radius, welcher durch die Entfernung zum nahesten Objekt mit dem gewünschten Typ vorgegeben wird, befinden.

Zusätzlich zum eigentlichen Resultat der Funktionsberechnung kann auch der berechnete Pfad zurückgegeben werden. Infolgedessen kann hier die Funktionalität der Navigation bereits vorweggenommen werden.

Navigation

Die Aufgabe der Navigation besteht darin für zwei gegebene Lokationspunkte A, B eine Liste von (Lokations-)Punkten zu bestimmen, welche A und B miteinander verbinden. Im vorigen Abschnitt wurde bereits die Berechnung eines Pfades mittels Dijkstra besprochen. Auch hier kann das Verfahren eingesetzt werden. Der Unterschied zur Berechnung des nächsten Nachbarn besteht in diesem Fall jedoch darin, dass das Ziel bekannt ist. Eine diskriminierungsfreie Suche wie im Fall von Dijkstra bedingt infolgedessen einen zu hohen Aufwand. Einsatz heuristischer Suchalgorithmen (A*-Algorithmus [HNR68, RN03]) bietet aufgrund zielgerichteter Gestaltung der Suche hier einen Vorteil. Infolgedessen kann die Anzahl der besuchten Knoten im Vergleich zum Dijkstra-Verfahren gering gehalten werden.

Auch im Fall der Navigation kann aufgrund der Typisierung der Knoten und Kanten Definition der *Begehbarkeit* eines Pfades, analog zur Berechnung des nächsten Nachbarn, auf Bedürfnisse des Benutzers angepasst werden.

Inklusion

Ziel der Inklusionsberechnung besteht darin für eine gegebene Lokation alle darin enthaltenen Objekte von einem bestimmten Typ zu bestimmen. Die bereits mehrfach angesprochene Typisierung der Objekte erfolgt in einem zusätzlichen Feld, das jedem Knoten des Modells zugeordnet wird. Als Typbezeichnung eines Objektes kommt alles in Frage, was dieses in irgendeiner Form spezifiziert: Schuhladen, Restaurant, Konferenzraum, ... Sollen nun für eine gegebene Lokation alle Objekte eines bestimmten Typs bestimmt werden, so besteht die naheliegende Lösung darin die (im Autoritätsbaum und den eventuell zugeschalteten Erweiterungsbäumen) unterhalb der Lokation liegenden Knoten abzusuchen. Die Suche dauert $O(n)$ Schritte in Abhängigkeit der Anzahl der innerhalb gegebener Lokation modellierten Objekte. Eine Anordnung der Typbezeichnungen in einem speziellen Suchbaum (anstatt direkt in jeweiligen Knoten) verkürzt die Berechnungszeit der Inklusion auf $O(\log m + \log n + l)$ Schritte, wobei m Anzahl möglicher Typbezeichnung, n die Anzahl der Knoten mit dem gewünschten Typ im Gesamtbaum und l die Anzahl der Knoten mit dem gewünschten Typ unterhalb des gegebenen Objektes darstellt. Hierzu verwaltet jeder Knoten innerhalb des Suchbaums bezüglich der Typbezeichnungen eine Menge von Referenzen auf den Autoritätsbaum, welche wiederum (bezüglich der Inklusionsbeziehung) in einer Baumstruktur angeordnet werden.

Abbildung 5.3 illustriert die Vorgehensweise. Gesucht sind alle Objekte von Typ a innerhalb der von Objekt X eingenommenen Lokation (hier: Z und W). Hierzu wird zunächst Knoten a im Suchbaum der Typbezeichnungen bestimmt, anschließend alle von diesem Knoten referenzierten Objekte im Autoritätsbaum zurückgegeben, welche sich unterhalb des Knoten X befinden. Die Bestimmung der Referenzen vom Knoten a aus, welche sich unterhalb des Knotens X befinden, kann bezüglich der im Knoten a enthaltenen Referenzen in logarithmischer Zeit bestimmt werden, wenn diese entsprechend der Inklusionsbeziehungen des Autoritätsbaumes angeordnet sind. Des Weiteren kann auf diese Weise Objektknoten mehrere Typbezeichnungen zugewiesen werden. In Abbildung 5.3

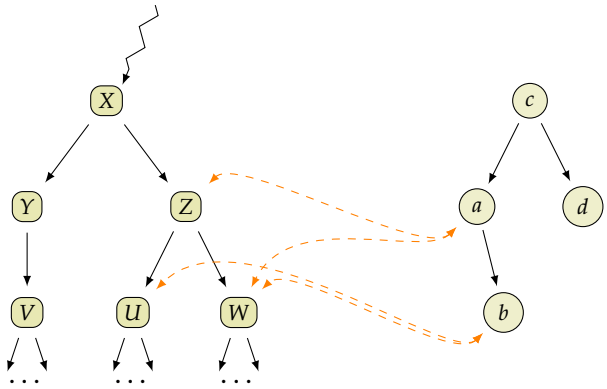


Abbildung 5.3.: Inklusionberechnung in SmartGPS

besitzt Knoten *W* sowohl Typ *a* als auch *b* und wird in beiden Fällen bei der Berechnung der Inklusion der Ergebnismenge beigefügt.

Inverse Inklusion

Für einen gegebenen Lokationspunkt und einen Objekttyp sollen im Fall der inversen Inklusion alle Objekte bestimmt werden, die den gewünschten Typ aufweisen und deren Lokation den gegebenen Punkt umschließt. Die Berechnung vollzieht sich in zwei Schritten:

1. Abstieg im Autoritätsbaum bis zum Blatt, welches den gegebenen Lokationspunkt umschließt. Da Knoten zweier Unterbäume stets disjunkte Lokationen beschreiben, ist das zu findende Blatt eindeutig. Bei jedem Schritt nach unten muss für jedes der Kinder einer Ebene berechnet werden, ob der Lokationspunkt sich innerhalb dieser befindet (Schnittberechnung). Um den Abstieg effizient zu gestalten, werden alle Kinder einer Ebene nach den Kriterien kleinster Längengrad, größter Längengrad, kleinster Breitengrad, größter Breitengrad sortiert angelegt. In einem solchen Fall beschränkt sich die Suche nach dem passenden Abstiegsknoten auf nur wenige Kandidaten, in häufigen Fällen ist es genau einer. Der Sachverhalt wird durch folgendes Beispiel verdeutlicht: Für den Punkt (48°44'41.57", 9°6'23.52") (Informatikgebäude Universität Stuttgart) ist beim Abstieg vom Deutschland-Knoten auf die Bundesland-Ebene der passende Knoten gesucht. Tabelle 5.1 stellt die nach minimalen / maximalen Breiten- und Längengraden sortierten 16 Bundesländer dar. Nach Abgleich dieser Werte (in logarithmischer Zeit) kommt genau ein Knoten zum Abstieg in Frage: nämlich *BW*. Eine genaue Lokationsbeschreibung des Knotens wird in diesem Fall erst gar nicht benötigt.

   Kommen dennoch mehrere Knoten in Betracht, so muss unter diesen derjenige bestimmt werden, in dessen Lokation sich der gegebene Punkt tatsächlich befindet. Da die Beschreibung einer Lokation sehr umfangreich sein kann (beispielsweise

min. Breitengrad		max. Breitengrad		min. Längengrad		max. Längengrad	
47°15′47″	BY	49°37′55″	SAR	5°53′26″	NRW	7°23′23″	SAR
47°31′52″	BW	49°46′53″	BW	6°08′04″	RP	8°26′14″	RP
48°58′14″	RP	50°33′03″	BY	6°22′04″	SAR	8°58′23″	BR
49°05′47″	SAR	50°56′08″	RP	6°43′55″	NI	9°27′49″	NRW
49°22′48″	HE	51°38′24″	TH	7°30′03″	BW	9°38′25″	BW
50°08′54″	S	51°38′52″	HE	7°47′50″	HE	10°14′07″	HE
50°12′14″	TH	51°39′42″	S	8°16′55″	SH	10°18′43″	HH
50°18′14″	NRW	52°30′48″	NRW	8°28′52″	BR	11°19′11″	SH
50°57′19″	SA	52°40′06″	B	9°00′06″	BY	11°35′26″	NI
51°18′16″	NI	53°01′40″	SA	9°44′16″	HH	12°39′49″	TH
51°20′19″	BA	53°31′33″	BA	9°53′36″	TH	13°10′51″	SA
52°20′17″	B	53°36′20″	BR	10°33′59″	SA	13°45′26″	B
53°00′33″	BR	53°44′13″	HH	10°34′52″	MV	13°49′41″	BY
53°07′34″	MV	53°52′41″	NI	11°17′53″	BA	14°44′48″	MV
53°21′14″	SH	54°40′52″	MV	11°52′33″	S	14°45′32″	BA
53°23′28″	HH	55°02′59″	SH	13°05′44″	B	15°01′45″	S

Tabelle 5.1.: Sortierung der Bundesländer nach minimalen, maximalen Längen- und Breitengraden

Polygonzug zur exakten Beschreibung des Umrisses eines Bundeslandes), wird ein iteratives Verfahren eingesetzt, welches für eine Lokation knapp mehr als eine Speicherung der exakten Beschreibung erfordert. Das Verfahren soll an einem Beispiel demonstriert werden: Abbildung 5.4 zeigt wie das Bundesland Baden Württemberg durch Angabe von fünf Punkten approximiert dargestellt werden kann. Die Grenzen des Polygons besitzen hierbei einen Abstand von nicht mehr als d (Kilo)Metern zum tatsächlichen Grenzverlauf. Befindet sich der angegebene Lokationspunkt in einem größeren Abstand als d von jeder der Polygongrenzen entfernt (hier: Punkt A), so kann (durch Anlegen einer Geraden und Abzählen der geschnittenen Kanten) bereits mit Sicherheit bestimmt werden, ob dieser sich innerhalb (ungerade Anzahl der Schnitte) bzw. außerhalb (gerade Anzahl der Schnitte) des Bundeslandes befindet ohne den genauen Verlauf der Lokation betrachten zu müssen. Im Fall, wenn der Lokationspunkt sich nah an einer oder mehreren Polygongrenzen befindet (hier: Punkt B), muss ein genauerer Grenzverlauf in Betracht gezogen werden.

2. Aufstieg vom Blatt zur Wurzel. Hierbei werden Knoten, welche vom gesuchten Typ sind, als Ergebnis zurückgegeben. Im Autoritätsbaum ist der Pfad vom Blatt bis zur Wurzel eindeutig. Sind zusätzliche Erweiterungsbäume zugeschaltet, existieren mehrere Pfade zur Wurzel des Autoritätsbaumes. Bei einem Einstieg in einen Erweiterungsbaum werden dessen Knoten bei Traversieren markiert, so dass mehrfaches Durchlaufen eines (Teil-)Pfades verhindert wird.

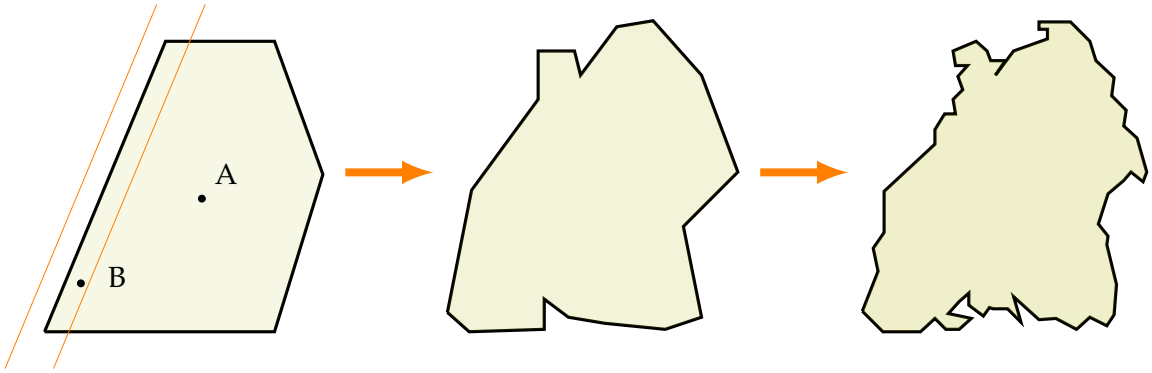


Abbildung 5.4.: Schrittweise Approximation einer Lokation

Ist der Lokationspunkt nicht durch GPS-Koordinaten, sondern als Adresse oder gar als eine Referenz auf einen Knoten im Autoritäts- oder Erweiterungsbaum gegeben, so erübrigt sich der beschriebene Abstieg im Baum.

Mengenoperationen und Metalokationen

Vereinigung und Schnitte von Lokationen sowie Definition von Metalokationen lassen sich in SmartGPS auf eine elegante Weise mittels Erweiterungsbäumen handhaben. Sollen zwei Lokationen vereinigt werden, so genügt das Anlegen eines einzelnen Knotens, wie in Abbildung 5.5 darstellt. Der neu definierte Knoten bildet einen Erweiterungsbaum, dessen Einstiegsknoten der unterste Knoten ist, welcher die zu vereinigenden Lokationen noch als Kinder hat. Ausstiegspunkte des Erweiterungsbaumes sind die zu vereinigenden Lokationen. Durch Vergabe einer Bezeichnung für den neu angelegten Knoten entsteht eine Metalokation. Dem Benutzer des SmartGPS ist nun beispielsweise möglich die Lokation *meine Arbeit* durch Vereinigung aller diesbezüglich relevanten Lokationen zu definieren.

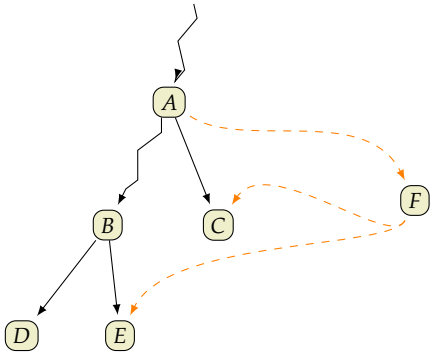


Abbildung 5.5.: Vereinigung mehrerer Lokationen in einer Metalokation

Modellierung eines Schnittes erfolgt auf eine ähnliche Weise (vgl. Abbildung 5.6). Zunächst wird eine mit Knoten des Autoritäts- oder Erweiterungsbaumes (hier: A, B) zu schneidende Lokation L modelliert. Diese bildet die Wurzel des neu entstehenden Erweiterungsbaumes.

Anschließend werden Schnittlokationen (hier: Knoten S_1 und S_2) berechnet. Die so entstandenen Knoten modellieren, sofern sich entsprechende Knoten des Autoritätsbaumes in unterschiedlichen Unterbäumen befinden, disjunkte Lokationen und werden als Kinder des Knoten L abgelegt. Als Letztes müssen Ausstiegspunkte des neuen Erweiterungsbaumes festgelegt werden. Dies sind Knoten des Autoritätsbaumes, welche komplett in den entstandenen Schnittlokationen enthalten sind (hier: C, D).

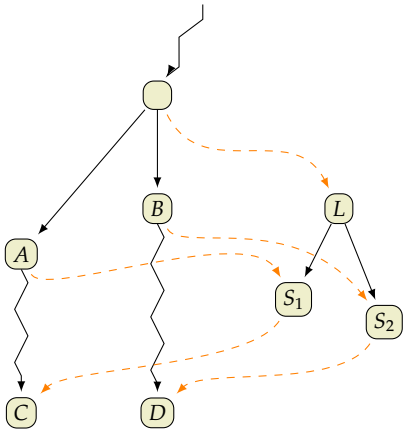


Abbildung 5.6.: Vereinigung mehrerer Lokationen in einer Metalokation

Auch die im Kapitel 3 angesprochenen Funktionen $\subset$, $\subseteq$ können, falls direkter Zugriff auf entsprechende Knoten besteht, in logarithmischer Zeit bezüglich der Anzahl modellierter Objekte berechnet werden. Sind Objekte über ihre Bezeichner im Autoritätsbaum gegeben, lässt sich die Inklusionsbeziehung in linearer Zeit bezüglich der Länge der Bezeichner auflösen. Seien zwei Objekte o_i, o_j mit entsprechenden über Alphabet Σ erzeugten Bezeichnungen b_i, b_j gegeben. Es gilt die folgende Bedingung:

$$loc(o_i) \subseteq loc(o_j) \Leftrightarrow \exists w \in \Sigma^* : b_i = b_j \cdot w \tag{5.2}$$

wobei das Symbol $\cdot$ eine Konkatenation zweier Wörter symbolisiert. Bezüglich der echten Inklusionsbeziehung gilt:

$$loc(o_i) \subseteq loc(o_j) \Leftrightarrow \exists w \in \Sigma^* : b_i = b_j \cdot w \wedge |w| \geq 1 \tag{5.3}$$

Orientierung

Im Bereich der PerFlows spielt Orientierung lediglich in Hinsicht auf die Bewegungsrichtung dynamischer Objekte (z.B. Gehrichtung einer Person) eine Rolle, im Bereich der statischen Objekte indessen spielt sie keine. Infolgedessen kann die Orientierung leicht durch eine Folge von Lokationserfassungen ermittelt werden (siehe Abschnitt 5.3).

Funktion	gefordert	Laufzeit SmartGPS	Ergänzung
$loc(o_i)$	$O(1)$	$O(1)$ $O(\log n)$	bei direktem Zugriff auf Knoten bei Zugriff über Objektname n: Anzahl modellierter Objekte
$dist(l_i, l_j)$	$O(1)$	$O(1)$	Gleichung 5.1
$nearest(l_i, t_j)$	$O(n \log n)$	$O(n \log n)$	Dijkstra-Algorithmus n: Anzahl modellierter Objekte
$incl(l_i, t_j)$	$O(m \log n)$	$O(\log l + \log p + m)$	l: Anzahl Objekttypen p: Anzahl Objekte von Typ t_j n: Anzahl modellierter Objekte m: Anzahl Objekte des Typs t_j innerhalb l_i
$enclosedBy(l_i, t_j)$	$O(\log n)$	$O(\log n)$ $O(\log n \cdot \log m)$	bei direktem Zugriff auf Knoten n: Anzahl modellierter Objekte beim Abstieg im Baum n: Anzahl modellierter Objekte m: Anzahl Kinder einer Ebene
$nav(l_i, l_j)$	$O(n \log n)$	$O(n \log n)$	A*-Algorithmus / Dijkstra n: Anzahl modellierter Objekte
$\cap$, $\cup$	$O(1)$	$O(1)$	bz. Anzahl modellierter Objekte Komplexität der Lokationsmodellierung entscheiden
$\subset$, $\subseteq$	$O(1)$	$O(\log n)$ $O(n)$	bei direktem Zugriff auf Knoten n: Anzahl modellierter Objekte bei Zugriff über Objektname n: Länge der Objektbezeichnung im Autoritätsbaum

Tabelle 5.2.: Laufzeitkomplexität der Funktionsberechnung in SmartGPS

Visualisierung

Aufgrund geometrischer Modellierung einzelner Lokationen können diese auf eine einfache Weise bildlich auf einer Karte dargestellt werden. Visualisierung bildet somit Benutzerschnittstelle, durch die Ergebnisse der meisten besprochenen Funktionen dargestellt werden können.

5.2.4. Effizienz

Laufzeitkomplexität der besprochenen Funktionalität ist in der Tabelle 5.2 zusammengefasst. Wie aus dieser hervorgeht, können Anforderungen bezüglich der Laufzeiten in den meisten Fällen erfüllt werden. Teuerste Operationen sind *nav* und *nearest*, wobei im ersten Fall mittels der Distanzberechnung und des A*-Algorithmus der Aufwand in der Praxis beschränkt werden kann. Im Fall der *nearest*-Berechnung kann seitens des Benutzers eine Schranke als Abbruchbedingung vorgegeben werden.

Es ist hervorzuheben, dass vollständige Lokationsmodellierung von Objekten (z.B. Polygonzug) bis auf Mengenoperationen zu keiner weiteren Funktionsberechnung benötigt wird. Somit erbringt SmartGPS trotz geometrischer Modellierung aller Lokationen mittels GPS-Koordinaten seine Funktionalität auf eine günstige Weise.

5.3. Lokationsmodellierung dynamischer Objekte

Lokationsmodellierung statischer Objekte ist Bestand des Abschnitts 5.2 gewesen. In diesem Abschnitt befassen wir uns mit der Modellierung von dynamischen Objekten, wie sie beispielsweise von Personen, Autos oder Kaffeetassen gebildet werden. Auch hier sollen GPS-Koordinaten als einheitliche Basis des Lokationsmodells verwendet werden. Im Gegensatz zu statischen Objekten verfügen dynamische im Lokationsmodell SmartGPS jedoch über keine Ausdehnung im Raum. Ihre Lokation wird vielmehr durch die Angabe eines Punktes im WGS84 beschrieben.

Wie im Abschnitt 5.1 erwähnt, ist Lokationsbestimmung dynamischer Objekte, im Gegensatz zu den statischen, häufig Fehler behaftet. Auch werden gewöhnlich zur Lokationsbestimmung eines Objektes mehrere Sensoren eingesetzt, deren (unter Umständen teilweise widersprüchlichen) Angaben fusioniert werden müssen. In diesem Abschnitt stellen wir eine einfache Möglichkeit vor, mit der die Handhabung mit Messungenauigkeiten sowie die Fusion mehrerer Quellen bewerkstelligt werden kann. Des Weiteren diskutieren wir eine Methode, mit deren Hilfe der Benutzer die bezüglich seiner Zwecke optimalen Quellen zur Lokation von Objekten bestimmen kann.

5.3.1. Konzept

Die zentrale Idee bei der Lokationserfassung von dynamischen Objekten ist der Verzicht auf Erfassung der Lokation als ein Punkt. Vielmehr wird die Position eines Objektes von einer Lokationsquelle (Sensor) als eine dreidimensionale Verteilung angegeben. Dies erscheint als sinnvoll, da die Wahrscheinlichkeit sich exakt an einem bestimmten Punkt im Raum zu befinden stets gleich Null ist. Durch die Angabe einer (stetigen) Wahrscheinlichkeitsverteilung kann hingegen mittels Integralberechnung über den gewünschten Raumabschnitt die Wahrscheinlichkeit bestimmt werden, mit der sich das gesuchte Objekt in diesem besagten Raumabschnitt befindet. Sei $Z = (X_1, X_2, X_3)$ eine dreidimensionale stetig verteilte Zufallsvariable, welche die Messergebnisse einer Lokationsquelle bezüglich eines gesuchten Objektes in einem dreidimensionalen Raum beschreibt. Sei $f(x_1, x_2, x_3)$ die Dichtefunktion der Zufallsvariable Z, dann gilt:

$$P(X_1 \in (a_1, b_1], X_2 \in (a_2, b_2], X_3 \in (a_3, b_3]) = \int_{a1}^{b1} \int_{a_2}^{b2} \int_{a_3}^{b3} f(x_1, x_2, x_3) dx_1 dx_2 dx_3 \tag{5.4}$$

oder allgemein:

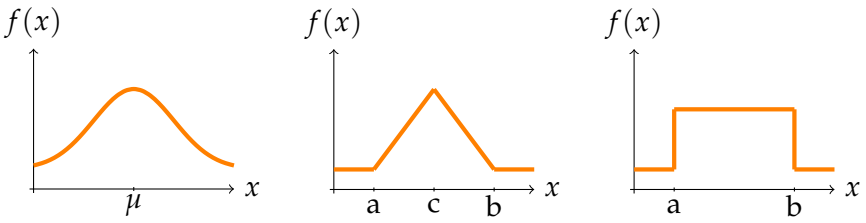
$$P((X_1, X_2, X_3) \in A) = \iiint_A f(x_1, x_2, x_3) dx_1 dx_2 dx_3 \tag{5.5}$$

Unter der Annahme der Unabhängigkeit einzelner Zufallsvariablen gilt:

$$P(X_1 \in (a_1, b_1], X_2 \in (a_2, b_2], X_3 \in (a_3, b_3]) = \prod_{i=1}^{3} P(X_i \in (a_i, b_i]) \tag{5.6}$$

Die hier unterstellte Annahme ist also die, dass Werte, welche beispielsweise zur Bestimmung der Höhe gemessen werden unabhängig von den bezüglich der Längen- und Breitengrades gemessenen Werte eines Objektes sind. Die Annahme erscheint bei genügend hoher Sensorgenauigkeit als sinnvoll und erleichtert entscheidend die Berechnung der Wahrscheinlichkeit. Eine dreidimensionale Verteilung lässt sich in diesem Fall als drei einzelne eindimensionale Verteilungen betrachten: Je eine für Höhe, Breiten- und Längengrad. Im Fall von symmetrischen Verteilungen (vgl. Abbildung 5.7) lässt sich eine Position mittels drei Erwartungswerte und drei Varianz- bzw. Streuungswerte bereits vollständig beschreiben.

Es ist zu bemerken, dass die erwähnten Längen-, Breiten- und Höhenverteilungen durchaus unterschiedlich sein können. Diese können sich sowohl in ihrer Streuung als auch in ihrer Art unterscheiden. So ist es beispielsweise sinnvoll, wie in Abbildung 5.8 dargestellt, dass die Verteilung eines sich bewegenden Objektes in Richtung dessen Bewegung



(a) Normalverteilung (b) Dreiecksverteilung (c) Gleichverteilung

Abbildung 5.7.: Möglichkeiten der Verteilungsarten

eine höhere Streuung besitzt, als die bezüglich der seitlichen Richtung, da aufgrund von beschränkter Aktualisierungsgeschwindigkeit der Lokationsergebnisse Messungenauigkeiten hier größer sein werden. Auch in Bezug auf die Verteilungsart sollen verschiedene Möglichkeiten berücksichtigt werden können. In häufigsten Fällen wird die Normalverteilung am geeignetsten sein, nichtsdestotrotz können auch beispielsweise Gleich- oder Dreiecksverteilungen (vgl. Abbildung 5.7) in Abhängigkeit von der Art der Sensoren ihre Anwendung finden.

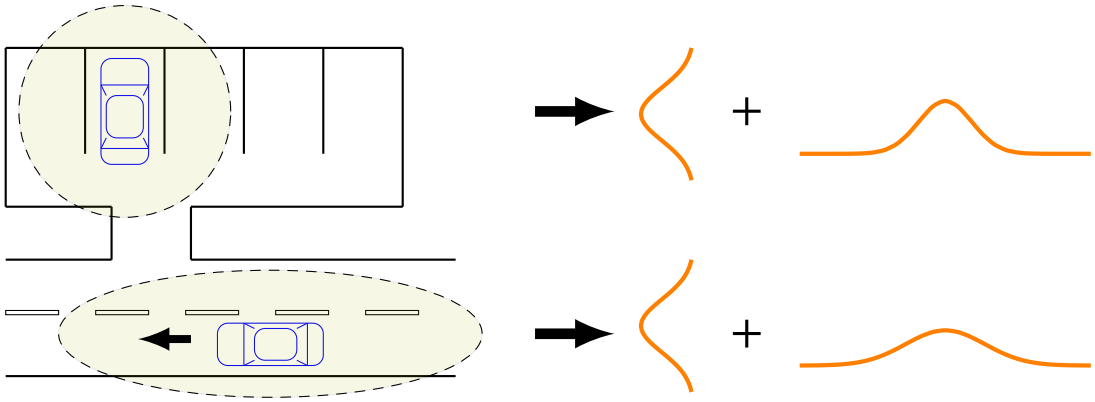


Abbildung 5.8.: Bewegungsabhängige Streuung

Sind nun bezüglich der Position eines Objektes mehrere Verteilungen verfügbar – beispielsweise weil mehrere Sensoren verwendet werden – so müssen diese fusioniert werden. Naheliegende Lösung besteht darin die vorhandenen Verteilungen zu gewichten und anschließend aufzuaddieren (vgl. Abbildung 5.9, 5.11). Sind alle Gewichte positiv und ergeben diese in der Summe 1, so ist es leicht nachzuvollziehen, dass die Aggregation der gewichteten Verteilungen ebenfalls, wie nun gezeigt, eine Verteilung ist.

Seien n Verteilungen durch Dichtefunktionen $f_i(x), i \in \{0, 1, \ldots n-1\}$ gegeben. Seien die Gewichte a_i mit $\sum_{i=0}^{n-1} a_i = 1$ gegeben. Es gilt:

$$\forall i \in \{0, 1, \ldots n-1\} : \int_{-\infty}^{\infty} f_i(x)dx = 1 \wedge \forall x : f_i(x) \geq 0 \tag{5.7}$$

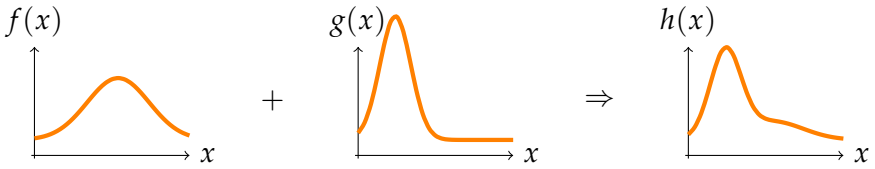
Für die gewichtete Summe $g(x) = \sum_{i=0}^{n-1} a_i f_i(x)dx$ gilt:

$$\forall x : g(x) \geq 0 \tag{5.8}$$

und für das Integral von $g(x)$:

$$\int_{-\infty}^{\infty} g(x) = \int_{-\infty}^{\infty} \sum_{i=0}^{n-1} a_i f_i(x)dx = \sum_{i=0}^{n-1} \int_{-\infty}^{\infty} a_i f_i(x)dx = \sum_{i=0}^{n-1} a_i \int_{-\infty}^{\infty} f_i(x)dx = \sum_{i=0}^{n-1} a_i = 1 \tag{5.9}$$

Die beschriebene Gewichtung der von den Lokationsquellen zurückgegebenen Verteilungen bestimmt im Wesentlichen den Anteil, welche eine Quelle dem Endergebnis beiträgt. Bei der in Abbildung 5.9 dargestellten Fusion zweier Verteilungen mit den von einander geringfügig abweichenden Erwartungswerten beeinflusst die Verteilung mit Dichtefunktion g wegen höherer Gewichtung stärker das Endergebnis, aus welchem Grund dieses ihr äußerlich ähnlich erscheint.



(a) $f = \mathcal{N}(2.7, 1.0)$ (c) $g = \mathcal{N}(1.2, 0.5)$ (e) $h = 0.3f + 0.7g$

Abbildung 5.9.: Fusion zweier Normalverteilungen

Die Gewichtung der von Lokationsquellen zurückgegebenen Verteilungen setzt sich in SmartGPS aus zwei Komponenten zusammen:

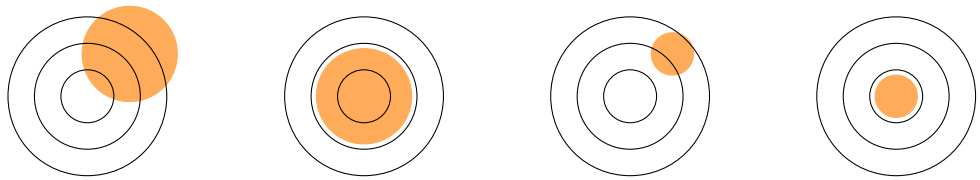
1. Streuung der jeweiligen Verteilung.
2. Genauigkeit der Lokationsquelle bzgl. der Erwartungswerte.

Streuung bzw. Varianz der zurückgegebenen Verteilung beeinflusst negativ das Gewicht dieser. Dies ist sinnvoll, da präzise Verteilungen gegenüber den flacheren aussagekräftiger und somit als wertvoller einzustufen sind. Die Genauigkeit einer Lokationsquelle setzt sich wiederum aus zwei Parametern zusammen:

1. *Richtigkeit* ist ein Maß für Übereinstimmung zwischen dem aus einem großen Datensatz erhaltenen Mittelwert und dem anerkannten Referenzwert. In unserem Fall ist die Richtigkeit der Kehrwert der erwarteten Abweichung des Erwartungswertes einer zurückgegebenen Verteilung von der tatsächlichen Position des zu lokalisierenden Objektes.

2. *Präzision* verhält sich antiproportional zur Streuung der Erwartungswerte der von der Quelle zurückgegebener Verteilungen.

Eine Lokationsquelle ist genau, wenn sie sowohl richtig als auch präzise ist. Abbildung 5.10 verdeutlicht den Sachverhalt. Nur im letzten Fall handelt es sich um eine genaue Quelle.



(a) niedrige Präzision, schlechte Richtigkeit

(b) niedrige Präzision, gute Richtigkeit

(c) hohe Präzision, schlechte Richtigkeit

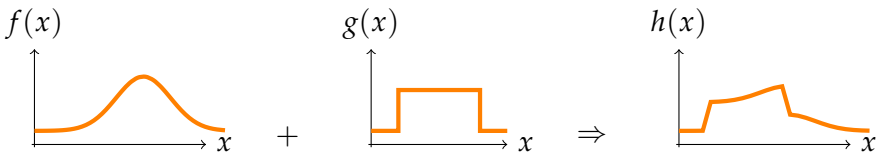
(d) hohe Präzision, gute Richtigkeit

Abbildung 5.10.: Unterschied zwischen Genauigkeit, Richtigkeit und Präzision

Richtigkeit bzw. Erwartungstreue einer Lokationsquelle kann im Folgenden vorausgesetzt werden, da im Fall der Abweichung des Erwartungswertes der von einer Lokationsquelle zurückgegebener Erwartungswerte dieser durch Nachjustieren entsprechend angepasst werden kann. Somit hängt die Gewichtung der von einer Quelle zurückgegebener Verteilung nur noch von der Streuung der Verteilung sowie der Präzision der Quelle ab. Während die Streuung einer Verteilung mit jedem Rückgabewert variieren kann, ist die Präzision ein Quellen-spezifisches Charakteristikum und kann lediglich durch mehrfache Lokationsbestimmung ein und desselben Referenzobjektes ermittelt werden. Anhang A.1 zeigt wie die optimale Gewichtung hinsichtlich der Präzision von Quelle berechnet werden kann. In Bezug auf die Streuung der zurückgegebenen Verteilung kann hingegen analog Abschnitt 5.3.3 eine *Nutzenfunktion* angelegt werden, aus der anschließend die Gewichtung abgelesen wird. Beide Gewichte müssen im Anschluss durch ein Nutzer abhängiges Priorisieren normiert werden.

Als Letztes ist anzumerken, dass die Art der zu fusionierenden Verteilungsfunktionen keine Rolle spielt und somit auch Verteilungsfunktionen verschiedener Arten miteinander kombiniert werden können, wie in Abbildung 5.11 gezeigt.

Das in diesem Abschnitt präsentierte Konzept der Lokationsbestimmung dynamischer Objekte bietet gleich mehrere Vorteile. Durch die Modellierung einer Position als eine (dreidimensionale) Verteilung bietet es die Möglichkeit mit Messungenauigkeiten aber auch symbolischen Lokationsangaben umgehen zu können. Durch die Gewichtung und Aggregation der Verteilungen erlaubt es Ergebnisse von mehr als nur einer Quelle gleichzeitig zu



(a) $f = \mathcal{N}(4.0, 1.0)$ (c) $g = \mathcal{G}(1.0, 4.0)$ (e) $h = 0.3f + 0.7g$

Abbildung 5.11.: Fusion einer Normal- und Gleichverteilung

berücksichtigen und auch bei den sich (evtl. teilweise) widersprechenden Lokationsangaben auf willkürliche Auswahl zu verzichten. Durch die Aufspaltung der dreidimensionalen Verteilungen in jeweils drei eindimensionale gestaltet sich die Fusion, sowie die Berechnung der Wahrscheinlichkeit für den Aufenthalt eines Objektes im bestimmten Intervall als relativ einfach. Die konkrete Umsetzung des Konzepts in SmartGPS diskutieren wir nun im nächsten Abschnitt.

5.3.2. Architektur

Die primäre Aufgabe von SmartGPS im Bereich der dynamischen Objekte besteht, wie aus dem vorigen Abschnitt hervorgeht, darin für ein zu lokalisierendes Objekt eine dreidimensionale (zusammengesetzte) Verteilung zurück zu geben, welche die Position des Objektes im WGS84 beschreibt. Dieser Rückgabewert soll im Folgenden als *LoCa* (location characteristic) bezeichnet werden. Die dreidimensionale Verteilung kann unter den erwähnten Annahmen als drei eindimensionale Verteilungen beschrieben werden, welche sich jeweils wiederum aus mehreren gewichteten einfachen Verteilungen (Normal-, Dreiecks-, Gleichverteilung) zusammensetzen. In Abbildung 5.12 ist die Generierung eines LoCa-Objektes für den zweidimensionalen Fall beispielhaft dargestellt. Die einfachen Verteilungen lassen sich bereits mit nur wenigen Werten vollständig beschreiben. So genügt es in den meisten Fällen die Angabe des Typs, des Erwartungswertes und der Standardabweichung. Alternativ genügt bei Dreiecks- und Gleichverteilung die Angabe der Punkte a und b, eventuell c, falls die Dreiecksverteilung asymmetrisch ist (vgl. Abbildung 5.7). Prinzipiell besteht ein LoCa also aus drei für Länge, Breite, Höhe bestimmten, nicht notwendigerweise gleich langen, Vektoren bestehend jeweils aus mehreren einfachen Verteilungen mit je einem Gewicht. Im Folgenden gehen wir auf die Architektur des SmartGPS im Detail ein.

Wie aus der Abbildung 5.13 ersichtlich, besteht SmartGPS aus vier Schichten, welche als DataProvider, DataParser, LocationProvider (im Folgenden auch als (Lokations-)Quelle bezeichnet) und SmartGPS benannt sind. In der hier gewählten Nomenklatur steht eine Schichtbezeichnung sowohl für die Funktionalität einer Schicht als solche, als auch für das Konstrukt, welches aus dieser und allen darunter liegenden Schichten entsteht. Die vier Schichten betrachten wir nun im Einzelnen:

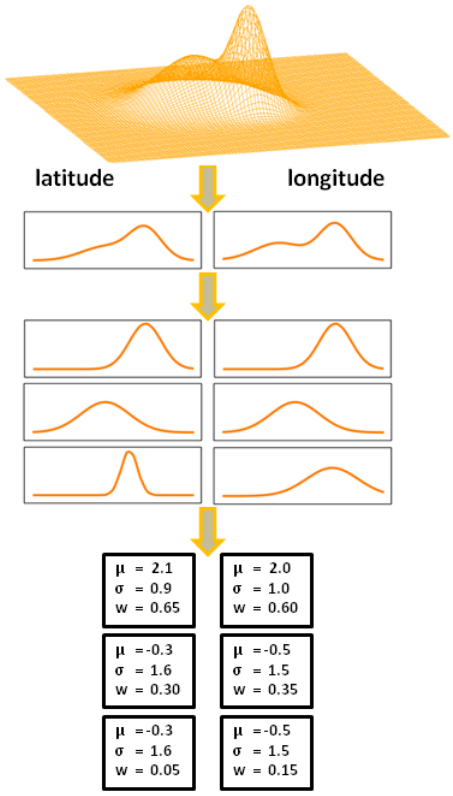


Abbildung 5.12.: Generierung eines LoCa–Objektes

DataProvider

DataProvider bilden die Schnittstelle zwischen SmartGPS-Anwendung und der Außenwelt. Die Aufgabe eines DataProviders besteht darin eine Verbindung zu einem Sensor herzustellen und einen Datenstrom zu generieren. Aus diesem sollen im weiteren Verlauf Lokationsangaben extrahiert werden können. Die jeweilige Implementierung eines DataProvider ist, aufgrund der Vielfalt an möglichen Sensoren, sehr spezifisch. Im Fall eines GPS-Empfängers beispielsweise liefert der DataProvider einen NMEA-Datenstrom (siehe Anhang A.4). In anderen Fällen kann es sich um eine vom Bewegungsmelder oder Infrarotbake stammende boolesche Variable, einen (vorverarbeiteten) Videostrom einer Webcam oder um aus der Anfrage eines WebServices nach GPS-Koordinaten einer Straße gewonnene Daten handeln. Die Generizität des DataProvider ist beabsichtigt und soll Erweiterbarkeit der SmartGPS-Anwendung um beliebige Datenquellen gewährleisten.

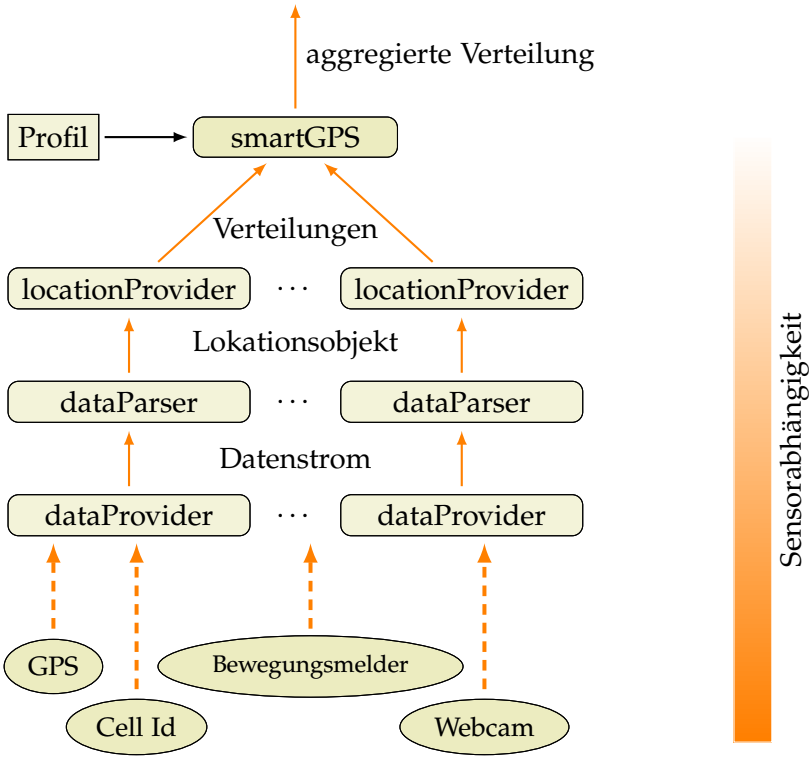


Abbildung 5.13.: Architektur des SmartGPS zur Lokationserfassung dynamischer Objekte

DataParser

In ihrer Implementierung sind DataParser ebenfalls Sensor spezifisch. Ihre zentrale Funktion, wie angedeutet, besteht darin den von DataProvider generierten Datenstrom zu lesen und ein (Sensor spezifisches) Lokationsobjekt zu extrahieren, in dem bereits Informationen bezüglich des Längengrades, Breitengrades und der Höhe des zu lokalisierenden Objektes enthalten sind. Das erstellte Lokationsobjekt wird anschließend an den LocationProvider gereicht.

LocationProvider

Ein LocationProvider bildet eine Schnittstelle zwischen den durch die Außenwelt gekennzeichneten unteren Schichten und der SmartGPS-Schicht. Nach unten hin stellt ein LocationProvider den Empfänger des Sensor spezifischen Lokationsobjektes dar, nach oben dagegen liefern alle LocationProvider die erwähnten Verteilungen bezüglich Längengrad, Breitengrad und Höhe, welche zusammen ein bereits einfaches LoCa-Objekt darstellen. LocationProvider ist insofern eine bedeutende Schicht, da hier aus diversen Sensor spezifischen Lokationsangaben Verteilungen über WGS84 generiert werden. Um dieses zu

ermöglichen, besitzt ein LocationProvider Zugriff auf Eigenschaften des Sensors, von welchem Lokationsangaben letztlich stammen. Die Verteilung, welche ein LocationProvider an die SmartGPS-Schicht zurückgibt, hängt also sowohl vom speziellen Lokationsobjekt als auch von allgemeinen Eigenschaften des Sensors ab. Im Fall eines GPS-Empfängers als Sensor kann die Standardabweichung bezüglich Längen- und Breitengrades auf zehn Meter voreingestellt werden. Zusätzlich kann diese in Abhängigkeit der Anzahl gesehener Satelliten – gelesen aus dem Lokationsobjekt – verkleinert oder vergrößert werden.

SmartGPS

SmartGPS-Schicht ist nun die letzte Schicht der SmartGPS-Anwendung und bildet zugleich eine Schnittstelle nach außen – beispielsweise als Client eines PerFlowManagers. Hier besteht die zentrale Funktionalität darin die von diversen Lokationsquellen erzeugten Verteilungen zu fusionieren. Dieses wird durch Gewichtung und eine anschließende Addition erreicht. Um eine Gewichtung der Lokationsquellen vornehmen zu können, verwaltet SmartGPS für jeden LocationProvider eine Spezifikation, welche unter anderem die bereits besprochene Präzision einer Quelle verwaltet. Weitere Punkte dieser sind:

- Durchschnittliche benötigte Zeit zur Beantwortung einer Anfrage
- Mittlere Streuung einer zurückgegebenen Verteilung
- Wahrscheinlichkeit bezüglich der Verfügbarkeit
- Durchschnittlicher Energieaufwand pro Anfrage (wichtig bei mobilen Geräten)
- Mittlere monetäre Kosten pro Anfrage

Die meisten der genannten Charakteristika werden nach einer jeden beantworteten Anfrage aktualisiert. Spezifikationen der Lokationsquellen werden auf diese Weise frisch gehalten. So kann beispielsweise bei jeder Anfrage die Zeit gemessen werden, welche eine Lokationsquelle zu Beantwortung dieser benötigt. Mit Hilfe der entstandenen Zahlenfolge wird ein gewichteter gleitender Durchschnitt berechnet, welcher in die Spezifikation der Quelle geschrieben wird. Tabelle 5.3 veranschaulicht an einem Beispiel mögliche Glättung der von einer Quelle benötigter Antwortzeiten: Erste Zeile stellt chronologisch die von einer Quelle jeweils benötigte Zeit (zum Zeitpunkt i) in Sekunden dar. Zweite Zeile gibt jeweils den nach der rekursiven Formel $\overline{t_i} = \alpha t_i + (1-\alpha)\overline{t_{i-1}}$ berechneten gewichteten Durchschnitt der benötigten Zeit mit Glättungsfaktor $\alpha = 0.5$ an.

Benötigte Zeit t_i	0.17	0.12	0.42	0.31	0.15	0.24	0.18	0.09	0.2
Gewichtete Zeit $\overline{t_i}$	0.17	0.145	0.283	0.296	0.223	0.232	0.206	0.148	0.174

Tabelle 5.3.: Exponentielle Durchschnittsbildung mit Glättungsfaktor 0.5

Um Lokationsquellen zu bestimmen, welche ein gewünschtes Objekt lokalisieren können, verwaltet SmartGPS eine Mapping-Tabelle. Diese ist dafür zuständig für eine ihr gegebene Objekt-Id eine Liste an Zweiertupeln der Form (LP-Id, Objekt-Id) zurückzugeben. Mit der aus einem solchen Tupel erhaltenen Objekt-Id kann sich SmartGPS anschließend an den

im selben Tupel enthaltenen LocationProvider richten, um die Lokation des gewünschten Objektes zu erhalten. Zu bemerken ist, dass die in der Liste enthaltenen Objekt-Ids weder miteinander noch mit der an SmartGPS übergeben Objekt-Id übereinstimmen müssen, im Allgemeinen es nicht tun. Eine konkrete Abbildung der Mapping-Tabelle könnte wie folgt aussehen:

$$\textit{MEIN AUTO} \rightarrow \boxed{\text{Mapping-Tabelle}} \rightarrow (LP_{12}, tr\_983002), (LP_{02}, \text{S-AC-100})$$

Eine vollständige Abarbeitung einer an SmartGPS gestellten Positionsanfrage stellt das Zeitdiagramm in Abbildung 5.14 graphisch dar: Zum Zeitpunkt t_0 erhält SmartGPS eine Anfrage mit einer Objekt-Id. Es wählt mit Hilfe der Mapping-Tabelle die hierzu in Frage kommenden Quellen aus, an welche zum Zeitpunkt t_1 parallel je eine Anfrage gestellt wird. Zugleich legt SmartGPS die maximale Zeit t_{max} fest, welche eine Lokationsquelle zur Bearbeitung der Anfrage nicht überschreiten darf und ignoriert alle Antworten, welche später als zum Zeitpunkt $t_1 + t_{max}$ ankommen (hier: von LP_0). Spätestens nach Ablauf von t_{max} Zeit zum Zeitpinkt t_4 generiert SmartGPS aus den bis dahin erhaltenen Verteilungen durch Gewichtung und Aggregation ein LoCa-Objekt und gibt dieses im Anschluss zurück. Zuvor aktualisiert er die Spezifikationen der von ihm befragten Lokationsquellen bezüglich Antwortzeit, Varianz, Verfügbarkeit, ...

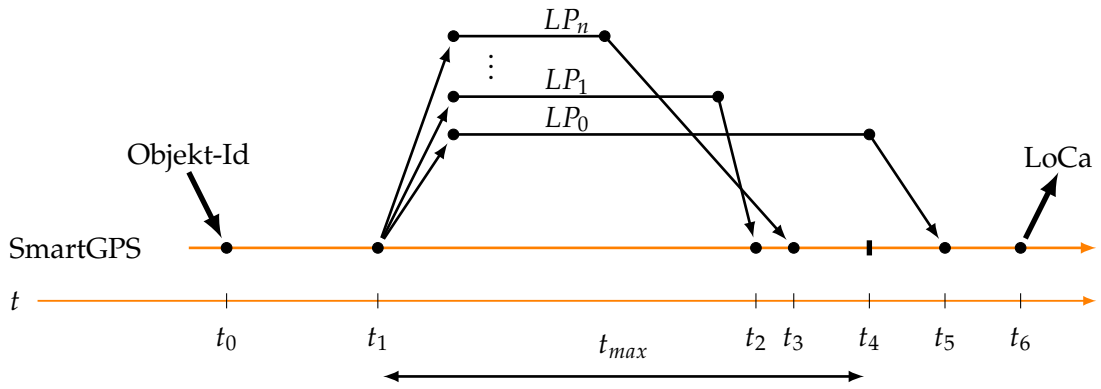


Abbildung 5.14.: Ablauf einer Lokalisierung

5.3.3. Multidimensionale Auswahl der Loktionsquellen

Stehen SmartGPS zur Lokalisierung eines Objektes mehrere Lokationsquellen zur Verfügung, so erwarten wir zunächst, dass dieser nach Möglichkeit alle Quellen befragt. In einem solchen Fall ist das der erhaltenen Antwort gegenüber erbrachte Vertrauen höher, als im Fall von nur sehr wenig befragten Quellen. Nichtsdestotrotz ist es nicht immer sinnvoll alle Lokationsquellen zu befragen. Man denke beispielsweise an den Energieaufwand oder monetäre Kosten, die mit einer solchen Anfrage in Verbindung stehen mögen. Wenn jedoch nicht alle möglichen Lokationsquellen befragt werden sollen, stellt

sich die Frage nach der optimalen Untermenge der zu befragenden Quellen. In diesem Kapitel setzen wir uns mit der Fragestellung auseinander wie unter mehreren zu berücksichtigenden Gesichtspunkten die Menge der Lokationsquellen ausgewählt werden kann, welche bezüglich der Bedürfnisse des Benutzers optimal ist, das heißt den höchsten *Nutzen* erbringt. Zur Bewertung verschiedener Alternativen wenden wir eine Kombination aus *Nutzwertanalyse* [Zan76] und *Analytic Hierarchy Process (AHP)* [Saa90b, Saa90a] an. Wir stellen zunächst vor wie unter vorgegebenen Kriterien die beste Lokationsquelle ausgewählt werden kann und diskutieren im Anschluss Möglichkeiten das Verfahren auf eine Menge von Lokationsquellen auszuweiten.

Nutzenberechnung einer Lokationsquelle

Wie im Abschnitt 5.3.2 bereits beschrieben, verwaltet SmartGPS für jede Lokationsquelle je eine Spezifikation. Attribute einer solchen Spezifikation können sein: Präzision, mittlere Varianz der zurückgegebener Verteilungen, mittlere Antwortzeit, Energieaufwand, Verfügbarkeit, monetäre Kosten. Für vereinfachende Analyse gehen wir davon aus, dass einzelne Attribute (auch Kriterium genannt) *nutzenunabhängig* [Zan76] sind. Die Bestimmung des Nutzens einer Lokationsquelle läuft nun in vier Schritten ab:

1. **Paarweiser Vergleich** Für jedes der Kriterienpaare K_i, K_j wird mittels einer Gewichtsvergabe ein Vergleich gezogen, wodurch beide Kriterien ins Verhältnis gesetzt werden. Das angegebene Gewicht drückt die subjektive Wertschätzung des Benutzers eines Kriteriums in Relation zu dem jeweils anderen aus. Fragestellungen der Art *Wie wichtig ist eine schnelle Beantwortung der Anfrage gegenüber einer präzisen?* werden in diesem Schritt geklärt. Das Resultat eines solchen Vergleiches ist eine $n \times n$-Matrix M mit 1en in der Diagonale, wobei n Anzahl der zu vergleichenden Kriterien ist. Matrix M muss zwei Bedingungen genügen:

   a) Kein Kriterium darf unendlich mehr wert sein als ein anderes: $\forall i, j \in \{0, 1, \ldots n-1\} : m_{ij} \neq \infty$.

   b) Ist ein Kriterium A r-mal so wichtig wie ein Kriterium B, so ist B $1/r$ mal so wichtig wie das Kriterium A: $\forall i, j \in \{1, 2, \ldots n\} : m_{ij} = 1/m_{ji}$. Der Entscheidungsträger verhält sich *reziprok*.

2. **Gewichtsberechnung** Das Ziel der Gewichtsberechnung ist ein Gewichtsvektor w der Länge n, welcher aus der im vorigen Schritt gewonnen Matrix berechnet wird. Für den Vektor muss gelten: $\sum_{i=0}^{n-1} w_i = 1$. In w stehen die letztendlichen Gewichte, welche über den Beitrag des jeweiligen Kriteriums zum Endresultat – dem Nutzen einer Lokationsquelle – bestimmen. Im Idealfall ist die berechnete Matrix M *konsistent*, d.h. es gilt: $\forall i, j, k \in \{0, 1, \ldots n-1\} : m_{ik} = m_{ij} \cdot m_{jk}$. In einem solchen Fall stellen alle Zeilen der Matrix ein Vielfaches der ersten dar und die Gewichte w_i können direkt ausgelesen werden. Die Stärke des AHP-Verfahrens basiert jedoch gerade darauf vom Entscheidungsträger keine konsistenten Vergleiche zu verlangen. Dieser kann somit beispielsweise folgende Gewichtsvergabe vornehmen: Kriterium A zu B: 3, B zu C : 2, aber A zu C nur 0.5 anstatt des erwarteten Gewichtes 6. Mittels der Berechnung des normierten Eigenvektors der Matrix M mit dem höchsten Eigenwert erhalten wir

den gewünschten Gewichtsvektor. Ausführliche Begründung des Vorgehens erfolgt in [Saa90a]. Die Berechnung erfolgt in drei Schritten:

a) Quadrieren der Matrix M.

b) Aufsummieren der Zeilenwerte.

c) Normieren des gewonnen Vektors auf 1.0.

Das Resultat des Verfahrens konvergiert gegen den beschriebenen Eigenvektor der Matrix M und kann abgebrochen werden, wenn sich zwei in Runde i und $i+1$ berechneten normierten Vektoren nur noch marginal unterscheiden. In der Praxis reichen bereits zwei Iterationen meist aus (vgl. Beispiel weiter unten).

Der Grund für die Berechnung des Gewichtsvektors über die aus den paarweisen Vergleichen gewonnene Matrix statt dessen direkten Angabe liegt in der besseren Übersicht für den Benutzer, somit der höheren Qualität der errechneten Gewichte. Anstelle einer gleichzeitigen Betrachtung und Bewertung aller Kriterien beschränkt sich der (qualitative) Aufwand auf den Vergleich lediglich zweier Kriterien. Der (quantitative) Aufwand steigt jedoch quadratisch mit der Anzahl der zu berücksichtigenden Kriterien.

3. **Schrankendefinition** Für jedes der Attribute K_i definiert der Benutzer eine untere und obere Schranke der zu berücksichtigenden Werte. Nicht sinnvolle (z.B. negative Wartezeit) und nicht zu akzeptierenden (z.B. 5 Minuten Wartezeit) Werte sollen in diesem Schritt für die spätere Betrachtung eliminiert werden. Der Wertebereich W_i eines Attributs wird eingeschränkt.

4. **Nutzenfunktionen** Wiederum für jedes Kriterium K_i erstellt der Benutzer eine Nutzenfunktion u_i. Diese bildet physikalische Werte aus den im vorigen Schritt bestimmten Intervallen auf ein für alle Kriterien einheitlich festgelegtes einheitsloses Intervall I (zum Beispiel $I = [0 : 100]$). Der Funktionswert einer Nutzenfunktion repräsentiert den Nutzen, den eine Attributausprägung für den Benutzer darstellt. Im Allgemeinen sind solche Nutzenfunktionen nicht linear (vgl. Beispiel weiter unten).

5. **Nutzenberechnung** Im letzten Schritt wird für gegebene n Kriterien $K_1, K_2, \ldots, K_n$ mit den Ausprägungen $k_1, k_2, \ldots, k_n$ der Gesamtnutzen u, mit

$$u : W_1 \times W_2 \times \cdots \times W_n \rightarrow I \tag{5.10}$$

wie folgt berechnet:

$$u = \sum_{i=1}^{n} w_i \cdot u_i(k_i) \tag{5.11}$$

Die so berechnete Zahl $u \in I$ gibt den Nutzen einer Lokationsquelle an, mit dessen Hilfe verschiedene Lokationsquellen nun *multidimensional* untereinander verglichen werden können. Im Folgenden soll das Verfahren anhand eines Beispiels veranschaulicht werden.

Beispiel: Seien drei Kriterien einer Lokationsquelle Zeit, Kosten und die Wahrscheinlichkeit der Verfügbarkeit gegeben. Im ersten Schritt erstellen wir einen paarweisen Bewertungsvergleich:

$$\text{Zeit} \overset{3/1}{\longleftrightarrow} \text{Kosten}$$

$$\text{Verfügbarkeit} \overset{3/2}{\longleftrightarrow} \text{Zeit}$$

$$\text{Kosten} \overset{1/4}{\longleftrightarrow} \text{Verfügbarkeit}$$

Aus den paarweisen Vergleichen gewinnen wir die 3×3-Matrix:

	Zeit	Kosten	Verfügbarkeit
Zeit	1	3	$2/3$
Kosten	$1/3$	1	$1/4$
Verfügbarkeit	$3/2$	4	1

aus der nun einzelne Gewichte der Kriterien berechnet werden. Quadrieren der Matrix:

$$\begin{pmatrix} 1 & 3 & 2/3 \\ 1/3 & 1 & 1/4 \\ 3/2 & 4 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 3 & 2/3 \\ 1/3 & 1 & 1/4 \\ 3/2 & 4 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 8\,2/3 & 2\,1/2 \\ 1\,1/24 & 3 & 13/18 \\ 4\,1/3 & 12\,1/12 & 3 \end{pmatrix}$$

Aufsummieren der Zeilen und anschließendes Normieren des Vektors:

$$\begin{pmatrix} 3 & + & 8\,2/3 & + & 2\,1/12 \\ 1\,1/24 & + & 3 & + & 13/18 \\ 4\,1/3 & + & 12\,1/2 & + & 3 \end{pmatrix} \Rightarrow \begin{pmatrix} 13\,3/4 \\ 4\,55/72 \\ 19\,5/6 \end{pmatrix} \Rightarrow \begin{pmatrix} 0.3586 \\ 0.1242 \\ 0.5172 \end{pmatrix} = E_1$$

Wiederholen der Quadratur der Matrix und Berechnung der Eigenvektornäherung:

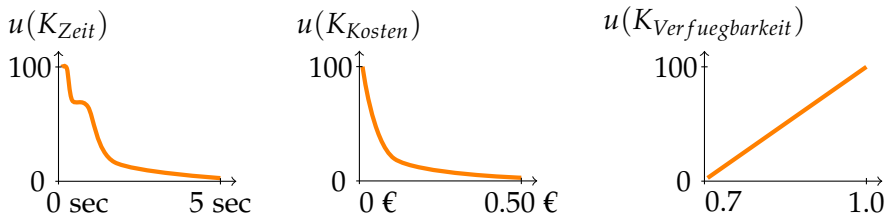
$$\begin{pmatrix} 3 & 8\,2/3 & 2\,1/12 \\ 1\,1/24 & 3 & 13/18 \\ 4\,1/3 & 12\,1/2 & 3 \end{pmatrix} \times \begin{pmatrix} 3 & 8\,2/3 & 2\,1/12 \\ 1\,1/24 & 3 & 13/18 \\ 4\,1/3 & 12\,1/2 & 3 \end{pmatrix} = \begin{pmatrix} 27\,1/18 & 77\,25/26 & 18\,41/54 \\ 9\,41/108 & 27\,1/18 & 6\,145/288 \\ 39\,1/48 & 112\,5/9 & 27\,1/18 \end{pmatrix}$$

$$\begin{pmatrix} 27\,1/18 & + & 77\,25/26 & + & 18\,41/54 \\ 9\,41/108 & + & 27\,1/18 & + & 6\,145/288 \\ 39\,1/48 & + & 112\,5/9 & + & 27\,1/18 \end{pmatrix} \Rightarrow \begin{pmatrix} 123.7764 \\ 42.9387 \\ 178.6319 \end{pmatrix} \Rightarrow \begin{pmatrix} 0.3584 \\ 0.1243 \\ 0.5173 \end{pmatrix} = E_2$$

Die Differenzen zwischen den Koeffizienten der Vektoren E_1 und E_2 liegen bereits nach der zweiten Iteration des Verfahrens unter 0.01 %, weswegen das Verfahren mit E_2 als Ergebnis an dieser Stelle beendet wird. E_2 enthält somit die für weitere Berechnung benötigten Gewichte der Kriterien Zeit, Kosten, Verfügbarkeit. Wir erkennen, dass das Kriterium Verfügbarkeit unter den drei berücksichtigten mit Abstand das wichtigste ist, gefolgt von Zeit und Kosten. Als nächstes definieren wir für jedes Attribut obere und untere Schranke der möglichen Ausprägungen sowie je eine Nutzenfunktion:

Für das Kriterium Zeit setzen wir die Schranken durch das Intervall $[0\text{ sec} : 5\text{ sec}]$. 5 Sekunden ist also die Zeit, die wir maximal bei einer Anfrage zu warten bereit sind. Längere Wartezeiten sollen erst gar nicht in Betracht gezogen werden. Analog setzen wir die untere Schranken der Kosten auf 0€, die obere auf 0.50€. Für das Kriterium Verfügbarkeit akzeptieren wir 0.7 als den minimalen Wert, womit das Intervall auf $[0.7, 1.0]$ festgelegt ist.

Definitionen der Nutzenfunktionen der drei Kriterien sind in Abbildung 5.15 dargestellt. In Bezug auf die Zeit sind drei Stufen zu erkennen, welche jeweils etwa denselben Nutzen aufweisen: $[0\text{ sec} : 0.2\text{ sec}]$, $[0.2\text{ sec} : 0.5\text{ sec}]$ und $[1.5\text{ sec} : 5\text{ sec}]$. Die Nutzenfunktion hinsichtlich der monetären Kosten besitzt bei 0 den höchsten Wert, fällt jedoch schnell bei bereits leichten Steigerungen. Der Nutzen der Verfügbarkeit verhält sich dagegen linear in Bezug auf Wahrscheinlichkeit der Verfügbarkeit einer Lokationsquelle. Strategien zur Entwerfen der Nutzenfunktionen werden ausführlich in [Zan76] diskutiert.



(a) K_{Zeit} (b) K_{Kosten} (c) $K_{Verfuegbarkeit}$

Abbildung 5.15.: Nutzfunktionen der Kriterien Zeit, Kosten, Verfügbarkeit

Für eine Spezifikation mit den Werten

Durchschnittliche Antwortzeit = 1.2sec
monetäre Kosten = 0€
Wahrscheinlichkeit der Verfügbarkeit = 0.87

berechnet sich der Nutzen einer entsprechenden Quelle wie folgt:

$$u = 0.3584 \cdot u_{Zeit}(1.2) + 0.1243 \cdot u_{Kosten}(0) + 0.5173 \cdot u_{Verfuegbarkeit}(0.87)$$

Mit $u_{Zeit}(1.2) \approx 75$, $u_{Kosten}(0) = 100$ und $u_{Verfuegbarkeit}(0.87) = 56^2/3$ beträgt der Gesamtnutzen der Lokationsquelle 68.62.

Nutzenberechnung mehrerer Lokationsquellen

Das bisher besprochene Verfahren beschränkt sich zunächst auf die Bestimmung des Nutzens lediglich einer Quelle. Dessen Anwendung auf eine Lokationsquellenmenge soll nun im Folgenden besprochen werden.

Der Nutzen einer Menge von Lokationsquellen lässt sich im Allgemeinen nicht durch Aufsummieren der Nutzen einzelner Quellen ermitteln. Eine solche Berechnung wäre beispielsweise der parallelen Ausführung der Positionsanfragen im Bezug auf die aufgewandte Zeit nicht gerecht. Stattdessen berechnet sich der Gesamtnutzen durch Summe der Nutzen der Aggregation einzelner Kriterien aller Quellen, wobei die Art der Aggregation in Abhängigkeit vom jeweiligen Kriterium variiert. Seien Lokationsquellen $q_0, \ldots, q_{n-1}$ mit jeweils m Kriterien $K_0, \ldots, K_{m-1}$ gegeben. Seien die bereits ermittelten Kriteriumgewichte w_i, sowie die Nutzenfunktionen u_i mit $i \in \{0, \ldots, m-1\}$ gegeben. Es gilt:

$$u_{gesamt} = \sum_{i=0}^{m-1} w_i \cdot u_i(K_{i0} \oplus_i K_{i1} \oplus_i \cdots \oplus_i K_{i(n-1)}) \tag{5.12}$$

wobei K_{ij} das i.te Kriterium der Quelle j und $\oplus_i$ den Aggregationsoperator für das Kriterium i darstellen. SmartGPS unterscheidet hierzu drei Arten der Aggregation einzelner Kriterien, mit deren Hilfe Nutzen einer Lokationsquellenmenge errechnet wird:

- Addition: monetäre Kosten, Energieaufwand
- Maximum: Zeit, Mittlere Streuung der Verteilungen, Präzision der Quelle
- Multiplikation: Verfügbarkeit

Die besprochene Ermittlung des Gewichtsvektors und der Nutzenfunktionen muss seitens des Benutzers einmalig gemacht werden. Die Ergebnisse können anschließend als ein Profil abgespeichert und vom SmartGPS bei einer Lokalisierungsanfrage zur Ermittlung der optimalen Menge der zu befragenden Lokationsquellen verwendet werden (vgl. Abbildung 5.13). Es ist vorstellbar mehrere – beispielsweise je ein Kriterium betonend – Profile anzulegen, welche SmartGPS bei einer Lokationsanfrage zusammen mit der Objekt-Id übergeben werden. Die Art der Lokationsbestimmung kann somit, speziell in Hinsicht auf Präzision und Zeit, situationsabhängig gesteuert werden.

5.3.4. Variationen der Lokationserfassung

Die in diesem Kapitel besprochene Berechnung einer aggregierten dreidimensionalen Verteilung als Lokationsbeschreibung eines dynamischen Objektes ist lediglich eine, wenn auch die umfassendste, Art Lokation eines Objektes darzustellen. Es lassen sich jedoch weitere, zum Teil auf der Berechnung der aggregierten Verteilung basierende, Möglichkeiten der Lokationserfassung dynamischer Objekte finden. Im Folgenden sind vier verzeichnet:

1. Aggregation der von einer optimalen Untermenge an vorhandenen Lokationsquellen zurückgegebenen Verteilungen, so wie in diesem Kapitel beschrieben.

2. Erwartungswert der im ersten Punkt beschriebenen aggregierten Verteilung.

3. Einzelne Verteilung der Lokationsquelle mit dem höchsten Nutzen nach der im Abschnitt 5.3.3 beschriebenen Methode.

4. Erwartungswert der Verteilung der Lokationsquelle mit dem höchsten Nutzen.

Während Verfahren nach Punkt eins und drei Verteilungen als Rückgabewert einer Lokationserfassung zurückliefern, auf diese Weise dem Benutzer umfangreiche Informationen bieten, liefern Methoden der Punkte zwei und vier als Lokationsbeschreibung je eine einzelne GPS-Koordinate. Der Vorteil hiervon besteht jedoch in der Möglichkeit zurückgegebene Koordinaten auf eine einfache Weise als Eingangsparameter der im Abschnitt 5.2.3 besprochenen Funktionsberechnung zu übergeben, somit dynamische Objekte neben der reinen Lokationserfassung in die Funktionalität des SmartGPS einzubinden.

5.4. Einsatz des SmartGPS

Funktionalität des Lokationsmodells SmartGPS sowie Lokationserfassung dynamischer Objekte wurde in den Abschnitten 5.2 und 5.3 ausführlich besprochen. Ziel des letzten Abschnitts dieses Kapitels ist die Beschreibung einer Möglichkeit des praktischen Einsatzes von SmartGPS.

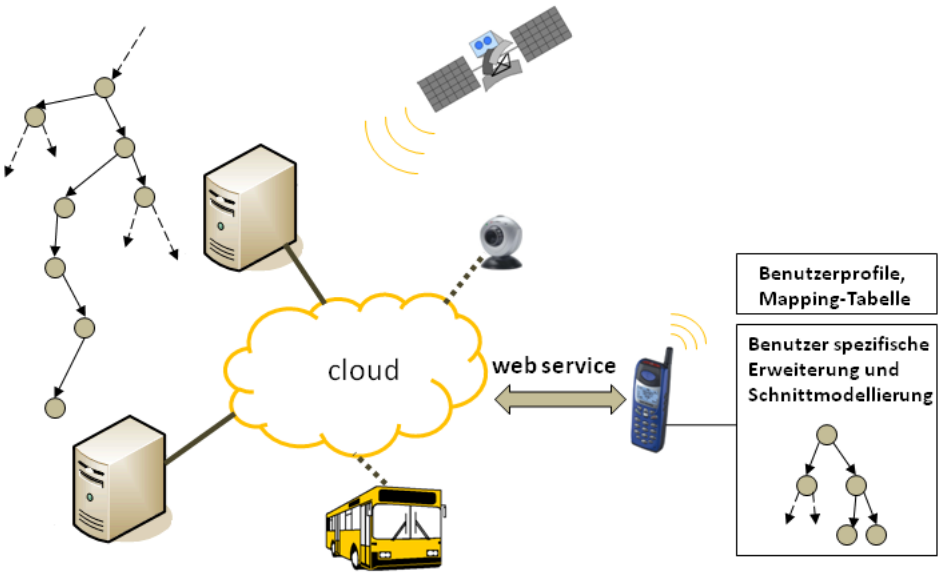
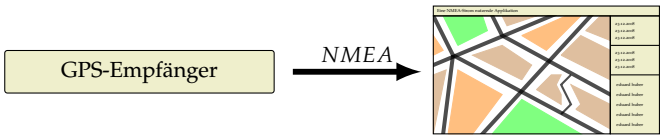


Abbildung 5.16.: Einsatz des Lokationsmodells SmartGPS

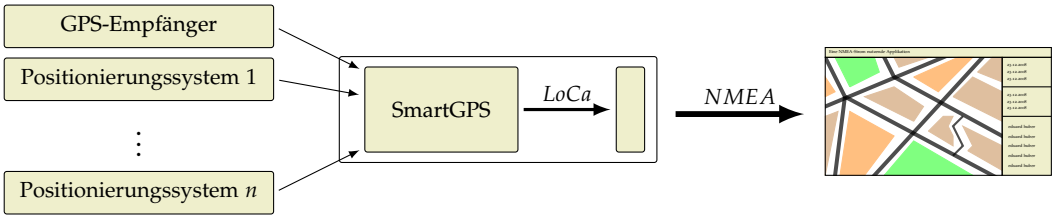
SmartGPS ist ein Lokationsmodell, welches in erster Linie an Anforderungen und Bedürfnisse des PerFlow-Managements angepasst ist. Dessen zentrale Aufgabe ist, wie im Kapitel 1 erwähnt, eine situations- und lokationsabhängige Unterstützung des Benutzers in dessen täglichen Abläufen und Prozessen. Die Ausführung eines PerFlow-Managers, so die Idee, läuft hierbei auf einem dem Benutzer jeweils aktuell zur Verfügung stehenden Rechner, in häufigen Fällen ein mobiles Endgerät. Auch an SmartGPS, dem Client eines PerFlow-Managers, besteht die Anforderung auf einem mobilen Gerät ausführbar zu sein. Wegen der immensen Anzahl der in der Praxis zu modellierenden Objekte, soll infolgedessen das gesamte Lokationsmanagement sowohl in Bezug auf die Verwaltung der Daten als auch auf die Funktionsberechnung, wie Abbildung 5.16 veranschaulicht, verteilt eingesetzt werden.

Das veranschaulichte Konzept sieht vor die Verwaltung statischer Objekte (Authoritätsbaum) durch leistungsstarke Rechner zu übernehmen. Ihre Funktionalität kann beispielsweise als Webservice frei über das Internet angeboten werden. Der durch die Baumstruktur erzeugte Adressraum erlaubt eine der DNS-Architektur [TS06] ähnliche baumartige Fragmentierung, sowie Replizieren des Authoritätsbaumes, was eine Verteilung über mehrere Knoten hinweg erlaubt. Ähnlich kann die Lokationserfassung der Öffentlichkeit zur Verfügung stehender dynamischer Objekte (beispielsweise öffentliche Verkehrsmittel) über WebServices angeboten werden.

Im Gegensatz dazu wird die persönliche Lokationserfassung (GPS-Tracking, Kalender-Einträge) zusammen mit Verwaltung der Benutzerprofile, Mapping-Tabelle sowie der Charakteristiken der Lokationsquellen auf dem mobilen Endgerät des Benutzers ausgeführt. Auch die Benutzer-spezifischen Erweiterungen des Autoritätsbaumes in Form von Erweiterungsbäumen werden lokal verwaltet. Für einen PerFlow-Manager ist die Verteilung der Lokationsdaten indessen transparent. Seine Schnittstelle zu SmartGPS bilden lediglich die im Kapitel 3 besprochen Funktionen.



(a) NMEA-Strom aus einem GPS-Empfänger



(b) Transparente NMEA-Strom-Erzeugung aus mehreren Positionierungssystemen

Abbildung 5.17.: Einsatzmöglichkeit des SmartGPS zur Erzeugung eines NMEA-Stroms

Eine weitere, auf persönliche Lokationserfassung zielende Einsatzmöglichkeit des SmartGPS ist in Abbildung 5.17 dargestellt. Die Idee hierbei besteht darin für diverse GPS basierten Anwendungen, welche im Normalfall den von einem GPS-Empfänger generierten NMEA-Datenstrom als Eingabe erhalten, SmartGPS als Datenquelle zu verwenden. Eine zwischen SmartGPS und der Applikation geschaltete Schnittstelle wandelt die vom SmartGPS erzeugten *LoCa*-Objekte in *RMC*-Datensätze (vgl. Anhang A.4.1) um, diese werden anschließend der Applikation als Eingabe übergeben. Für eine Applikation indessen bleibt die Änderung transparent. Allerdings verfügt diese nun, da sie indirekt auf mehrere Lokationsquellen zurückgreift, über eventuell genauere Positionsangaben und kann des Weiteren auch in Umgebungen, in denen kein GPS-Signal zur Verfügung steht, eingesetzt werden.

Kapitel 6

Implementierung

Lokationserfassung dynamischer Objekte, wie im Abschnitt 5.3 vorgestellt, ist im Rahmen der Diplomarbeit implementiert worden. In diesem Kapitell wollen wir kurz auf die eingesetzten Lokationsquellen und ihre Spezifikationen sowie die Konfiguration des verwendeten Profils eingehen.

Abbildung 6.1.: Sun Java Wireless Toolkit for CLDC

Die Implementierung erfolgte in *JavaME*[BMo8] mittels des online frei verfügbaren Emulators *WTK 2.5.2* der Firma *Sun* [wtk]. Eine einfache Benutzeroberfläche der Anwendung ist in Abbildung 6.1 dargestellt. Zu erkennen ist das Textfeld zur Eingabe der Bezeichnung des gesuchten Objektes unten, sowie Datenfelder zur Ausgabe der aktuellen Höhe, des Längen- und Breitengrades wie auch der Güte der aktuellen Lokationserfassung in Form eines Zahlenwertes im Intervall $[0, 100]$.

Die Architektur der Implementierung entspricht im Wesentlichen der, wie sie in Abschnitt 5.3.2, insbesondere in Abbildung 5.13, dargestellt ist. Als DataProvider kommen in der aktuellen Implementierung zum einen ein über die serielle COM-Schnittstelle angeschlossener GPS-Empfänger, zum Anderen der online frei verfügbare WebService *GeoNames* (vgl. Anhang A.3) zum Einsatz. Auf diesen aufbauend erfolgte je eine Implementierung entsprechender DataParser und LocationProvider, welche die Dateneingangsströme in dreidimensionale Verteilungen umwandeln. Im Fall des GPS-Empfängers wird hierzu der von diesem erzeugte NMEA-Strom, speziell die Datensätze *RMC* und *GGA*, ausgelesen (vgl. Anhang A.4) und in ein LoCa-Objekt umgewandelt. Im Fall des *GeoNames*-WebServices erfolgt eine Simulation eines Terminkalenders, aus welchem, falls vorhanden, symbolische Lokationsangaben

ausgelesen und in eine Anfrage an *GeoNames* umgewandelt werden. Initiale Spezifikationen der beiden Lokationsquellen, wie diese vor den Aktualisierungen vorzutreffen sind, sind in Abbildung 6.2 dargestellt.

```
<lpspec>                                  <lpspec>
  <id> local GPS receiver </id>             <id> GeoNames </id>
  <alt_var> 2500 </alt_var>                 <alt_var> 1000000000 </alt_var>
  <lat_var> 8.1E-9 </lat_var>               <lat_var> 8.1E-5 </lat_var>
  <long_var> 1.87E-8 </long_var>            <long_var> 1.87E-4 </long_var>
  <alt_dev> 50 </alt_dev>                   <alt_dev> 1000000000 </alt_dev>
  <lat_dev> 9.0E-5 </lat_dev>               <lat_dev> 9.0E-3 </lat_dev>
  <long_dev> 1.37E-4 </long_dev>            <long_dev> 1.37E-2 </long_dev>
  <alt_distr> uniform </alt_distr>          <alt_distr> invalid </alt_distr>
  <lat_distr> uniform </lat_distr>          <lat_distr> normal </lat_distr>
  <long_distr> uniform </long_ distr>       <long_distr> normal </long_ distr>
  <time> 3000 </time>                       <time> 500 </time>
  <power> 50 </power>                       <power> 10 </power>
  <availability> 1.0 </availability>        <availability> 1.0 </availability>
  <com_port> 4 </com_port>                  <address> http://ws.geonames.org </address>
  <baud_rate> 4800 </baud_rate>             <type> full </type>
</lpspec>                                 </lpspec>
```

Abbildung 6.2.: Spezifikationen verwendeter Lokationsquellen

Die Spezifikationen unterscheiden sich in erster Linie durch die Art der erzeugten Verteilungen (hier: Gleich- und Normalverteilung, bzw. die bzgl. der Höhe ungültige Verteilung im Fall von *GeoNames*), in der Varianz der Verteilungen (angegeben in Metern, respektive Graden), sowie der erwarteten Abweichung des Erwartungswertes von der tatsächlichen Position. Weitere Punkte sind: aufgewendete Zeit (in Millisekunden), Energieaufwand (angegeben auf einer Skala von 0 bis 100), Wahrscheinlichkeit der Verfügbarkeit, sowie einige Quellen-spezifische Konfigurationen wie die Portnummer im Fall des GPS-Empfängers oder die Adresse des WebServices *GeoNames*.

Die Erzeugung einer Verteilung erfolgt, wie im Abschnitt 5.3.2 bereits kurz erwähnt, zum einen anhand der in der Spezifikation vorzufindenden Werte bzgl. der Varianz, zum Anderen anhand des aktuellen vom DataParser erzeugten Datensatzes. Im Fall des GPS-Empfängers geht die Anzahl gesehener Satelliten sowie die im GGA-Datensatz enthaltene Angabe bezüglich der Signalqualität in die Bestimmung der Varianz zurückgegebener Verteilung ein. Im Fall des WebService *GeoNames* wird die Varianz anhand der Typs des zurückgegebenen Datensatzes und, falls es sich um ein besiedeltes Gebiet handelt, anhand der Einwohneranzahl bestimmt.

Zuletzt wollen wir auf das in der Implementierung eingesetzte Profil eingehen (vgl. Abschnitt 5.3.3), auf dessen Basis die Auswahl der Lokationsmengen und die Bewertung der Lokationserfassung erfolgt. Abbildung 6.3 stellt die Spezifikation des verwendeten

```
<profile>
  <altitude weight = „0.1“>
    <var weight = „0.5“>0, 100, 400, 75, 1000, 50, 4000, 25, 10000, 0</var>
    <prec weight = „0.5“>0, 100, 400, 75, 1000, 50, 4000, 25, 10000, 0</prec>
  </altitude>
  <latitude weight = „0.2“>
    <var weight = 0.5>8.1e-11, 100, 8.1e-9, 90, 8.1e-7, 50, 8.1e-5, 10, 2.025e-3, 0</var>
    <prec weight = 0.5>8.1e-11, 100, 8.1e-9, 90, 8.1e-7, 50, 8.1e-5, 10, 2.025e-3, 0</prec>
  </latitude>
  <longitude weight = „0.2“>
    <var weigth = 0.5>1.87e-10, 100, 1.87e-8, 90, 1.87e-6, 50, 1.87e-4, 10, 4.7e-2, 0</var>
    <prec weigth = 0.5>1.87e-10, 100, 1.87e-8, 90, 1.87e-6, 50, 1.87e-4, 10, 4.7e-2, 0</prec>
  </longitude>
  <power weight = „0.05“>0, 0, 100, 100</power>
  <time weight = „0.2“>0, 100, 100, 95, 1000, 70, 2000, 40, 3000, 20, 5000, 0</time>
  <availability weight = „0.25“>0.7, 0, 0.8, 20, 0.9, 40, 1.0, 100</availability>
</profile>
```

Abbildung 6.3.: Spezifikation des verwendeten Profils

Profils in XML-Notation dar. Zu sehen sind die sechs verwendeten Kriterien *Höhe*, *geographische Breite*, *geographische Länge*, *Energie*, *Zeit* und *Verfügbarkeit*, wobei die ersten drei sich nochmals je in das Kriterium *Varianz* und *Präzision* aufteilen. Zu jedem der Kriterien ist ein Gewicht und eine Nutzenfunktion angegeben. Die Gewichte geben die Relevanz des jeweiligen Kriteriums in Relation zu anderen Kriterien wider und ergeben in der Summe 1.0. Die verwendeten Nutzenfunktionen sind alle *konvex* und werden als Liste bestehend aus Paaren der Form $(x, f(x))$ angegeben. Zwischen den einzelnen Punkten erfolgt eine lineare Interpolation.

Kapitel 7

Zusammenfassung und Ausblick

Erarbeitung und Präsentation eines Lokationsmodells zum Einsatz im Bereich des PerFlow-Managements ist das Ziel dieser Arbeit gewesen. Hierzu analysierten wir im Kapitel 3 zunächst die aus dem Bereich des PerFlow-Managements resultierenden Anforderungen, leiteten aus diesen, meist in Form einer formalen Funktionsbeschreibung, die von einem Lokationsmodell zu erbringende Funktionalität ab. Das erarbeitete Lokationsmodell – SmartGPS – unterteilte sich aufgrund der im Abschnitt 5.1 erörterten Unterschiede in der Modellierung dynamischer und statischer Objekte entsprechend in zwei größere Bereiche. Wir zeigten auf, wie die geforderte Funktionalität mittels Autoritäts- und Erweiterungsbäumen im Bereich der statischen Objekte erbracht werden kann (vgl. Abschnitt 5.2). Lokationsmodellierung dynamischer Objekte, wie im Abschnitt 5.3 besprochen, erfolgte mittels Erzeugung, Gewichtung und Aggregation von Verteilungen über den WGS84-Raum. Dies ermöglichte uns diverse Lokationsquellen, geometrische so wie auch symbolische, zur Lokationserfassung dynamischer Objekte mit einzubeziehen. Aufgrund der prinzipiellen Nutzungsmöglichkeit aller Lokationsquellen ist SmartGPS somit ein sehr universelles Lokationsmodell.

Ein weiterer wichtiger Punkt im Bereich der Lokationserfassung dynamischer Objekte war die Bewertung einzelner Lokationsquellen nach verschiedenen Kriterien. Mittels der Erstellung eines Gewichtsvektors und entsprechender Nutzenfunktionen sowie der Abspeicherung dieser als Profil waren wir in der Lage Lokationsquellen multidimensional zu vergleichen und somit unter den gegebenen Quellen die in Bezug auf gestellte Bedingungen (situationsbedingt) optimale Untermenge auszuwählen (Abschnitt 5.3.3). Infolge permanenter Aktualisierung der Lokationsquellenspezifikationen ist SmartGPS in der Lage sich auf veränderte Umgebung – wie beispielsweise Ausfall einer Lokationsquelle – automatisch einzustellen.

In der im Abschnitt 5.4 geführten Diskussion bezüglich verschiedener Einsatzmöglichkeiten des SmartGPS zeigten wir auf, dass SmartGPS neben dem PerFlow-Management auch für diverse das NMEA-Protokoll nutzende Applikationen als Lokationsmodell verwendet werden kann.

Zukünftig ist Implementierung und Einsatz weiterer Lokationsquellen, wie WLAN-Triangulation oder Infrarotbaken in SmartGPS wünschenswert. Hinsichtlich Bewertung

einzelner Lokationsquellen können weitere Kriterien herangezogen werden, wodurch eine noch bessere Auswahl der Lokationsquellen getroffen werden kann. Eine interessante zu untersuchende Möglichkeit bezüglich Lokationserfassung dynamischer Objekte besteht in der Verwendung von SmartGPS als Lokationsquelle. In der hierdurch entstehenden Graphenstruktur wären den aus der Domäne der MANETs [Roto2] bekannten Routing-Algorithmen [RT99, JMB01] ähnliche Verfahren, unter anderem zur Zykluserkennung, notwendig. Des Weiteren wäre in diesem Fall zu untersuchen auf welche Weise das Profil einer SmartGPS-Instanz modifiziert werden müsste, um bei einer Lokationsanfrage an eine andere SmartGPS-Instanz mit übergeben werden zu können. Natürlich ist eine Untersuchung des Zusammenspiels zwischen SmartGPS und einem PerFlow-Manager in Alltagssituationen besonders erstrebenswert.

Anhang A

Anhang

A.1. Sensorfusion

A.1.1. Annahmen

Für eine Messgröße x seien zwei Sensoren geben, deren Messwerte sich durch Zufallsvariablen X_1, X_2 beschreiben lassen. X_1, X_2 seien beliebig verteilt mit $E(X_1) = E(X_2) = x$, bzw. $E(X_1 - x) = E(X_2 - x) = 0$. Messfehler $X_1 - x$ und $X_2 - x$ seien unabhängig. Seien des Weiteren σ_1, σ_2 Standardabweichungen der Zufallsvariablen X_1, X_2, d.h.: $E((X_1 - x)^2) = \sigma_1$ und $E((X_2 - x)^2) = \sigma_2$.

A.1.2. Ziel

Das Ziel besteht nun darin eine Zufallsvariable Z zu definieren, welche aus einer Linearkombination der Zufallsvariablen X_1, X_2 besteht, den selben Erwartungswert und die minimale Standardabweichung besitzt. Es soll also gelten:

$$Z = \alpha X_1 + \beta X_2 \tag{A.1}$$

$$E(Z - x) = 0 \tag{A.2}$$

$$\sigma_z{}^2 = E((Z - x)^2) \rightarrow min \tag{A.3}$$

A.1.3. Herleitung

$$
\begin{aligned}
0 = E(Z - x) &= E(\alpha X_1 + \beta X_2 - x) && (A.4)\\
&= \alpha E(X_1) + \beta E(X_2) - x = \alpha x + \beta x - x && (A.5)\\
&= x(\alpha + \beta - 1) && (A.6)\\
&\Rightarrow (x = 0) \vee (1 = \alpha + \beta) && (A.7)\\
&\Rightarrow \alpha = 1 - \beta \qquad \text{da für alle } x && (A.8)\\
&\Rightarrow Z = \alpha X_1 + (1 - \alpha) X_2 && (A.9)
\end{aligned}
$$

Es gilt nun ${\sigma_z}^2$ zu minimieren:

$$
\begin{aligned}
{\sigma_z}^2 &= E((Z - x)^2) && (A.10)\\
&= E((\alpha X_1 + (1 - \alpha) X_2 - x)^2) && (A.11)\\
&= E((\alpha(X_1 - x) + (1 - \alpha)(X_2 - x))^2) && (A.12)\\
&= E(\alpha^2 (X_1 - x)^2 + 2\alpha(x_1 - x) + (1 - \alpha)^2 (X_2 - x)^2) && (A.13)\\
&= \alpha^2 E((X_1 - x)^2) + 2\alpha E(X_1 - x) + (1 - \alpha)^2 E((X_2 - x)^2) && (A.14)\\
&= \alpha^2 {\sigma_1}^2 + (1 - \alpha)^2 {\sigma_2}^2 && (A.15)
\end{aligned}
$$

Bestimmung des Minimums:

$$
\begin{aligned}
0 &= \frac{d}{d\alpha}[\alpha^2 \sigma_1^2 + (1 - \alpha)^2 \sigma_2^2] && (A.16)\\
&= 2\alpha\sigma_1^2 + (2\alpha - 2)\sigma_2^2 && (A.17)\\
&= 2\alpha(\sigma_1^2 + \sigma_2^2) - 2\sigma_2^2 && (A.18)\\
&\Rightarrow (\alpha = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}) \wedge (\beta = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}) && (A.19)\\
&\Rightarrow Z = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} X_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} X_2 && (A.20)
\end{aligned}
$$

Standardabweichung der Zufallsvariable Z lässt sich nun wie folgt berechnen:

$$\sigma_z = E((Z-x)^2) = \alpha^2\sigma_1^2 + (1-\alpha)^2\sigma_2^2 \tag{A.21}$$
$$= \frac{\sigma_2^4\sigma_1^2}{(\sigma_1^2+\sigma_2^2)^2} + \frac{\sigma_1^4\sigma_2^2}{(\sigma_1^2+\sigma_2^2)^2} \tag{A.22}$$
$$= \frac{(\sigma_1^2+\sigma_2^2)\sigma_1^2\sigma_1^2}{(\sigma_1^2+\sigma_2^2)^2} \tag{A.23}$$
$$= \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2} = {}^1/_{1/\sigma_1^2+1/\sigma_2^2} \tag{A.24}$$
$$\Rightarrow \frac{1}{\sigma_z^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} \tag{A.25}$$

A.2. World Geodetic System 1984

Das *World Geodetic System 1984* (WGS84) [wgs04] ist ein oblates Rotationsellipsoid, welches die Grundlage des *Global Positioning System* (GPS) mittels *NAVSTAR-Satelliten* bildet.

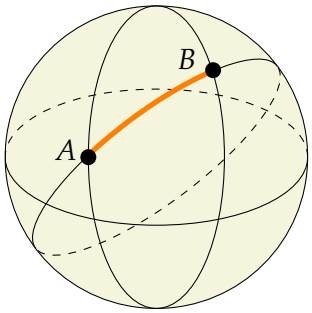


Abbildung A.1.: Orthodrome

Ein Rotationsellipsoid wird mittels der Angabe der Längen dessen zweier Halbachsen a (lange Achse), b (kurze Achse) bereits vollständig beschrieben. Im Fall des WGS84 gilt: $a = 6.378.137,0$ m, $b = 6.356.752,314$ m. Zur Abstandsberechnung zweier Koordinaten kann die Erde näherungsweise als Kugel mit einem Radius von 6370 Kilometern betrachtet werden. Die kürzeste Verbindung zwischen Punkten A und B – sogenannte *Orthodrome* – definiert sich als ein Teilstück des durch die beiden Punkte verlaufenden *Großkreises* (vgl. Abbildung A.1). Großkreis einer Kugel ist ein Kreis auf der Kugeloberfläche, dessen Mittelpunkt mit dem der Kugel übereinstimmt. Die Orthodrome kann als Winkel ς angegeben werden, wobei folgende Gleichung gilt:

$$\varsigma = \arccos(\sin\phi_A \cdot \sin\phi_B + \cos\phi_A \cdot \cos\phi_B \cdot (\cos\lambda_B - \cos\lambda_A)) \tag{A.26}$$

Zur Berechnung des Abstandes in Kilometern muss mit dem Radius der Erde ≈ 6370 km, bzw. zusätzlich, falls ς nicht als Bogenmaß sondern Grad angegeben, mit Faktor ${}^\pi/_{180}$ multipliziert werden.

A.3. GeoNames

GeoNames [geo] ist ein Open-Source-Projekt zur Speicherung und Verwaltung geographischer Merkmale wie der Städte, Parks oder öffentliche Gebäude. Daten der gespeicherten Einträge können über eine graphische Web-Schnittstelle oder ein API abgefragt werden. Nach eigenen Angaben verfügt GeoNames über 6.5 Millionen Einträge, darunter ca. 175.000 Deutschland bezogen. Alle der gespeicherten Einträge werden nach neun Klassen (beispielsweise *P (Populated Place Feature)*, oder *S (Spot Feature)*), sowie insgesamt 645 Unterklassen kategorisiert. Der im Folgenden abgebildete XML-Code stellt den in GeoNames vorhandenen Eintrag bzgl. Universität Stuttgart dar:

```
<geoname>
  <name>Stuttgart University of Applied Science</name>
  <lat>48.7798852697993</lat>
  <lng>9.173412322998O5</lng>
  <geonameId>6269548</geonameId>
  <countryCode>DE</countryCode>
  <countryName>Germany</countryName>
  <fcl>S</fcl>
  <fcode>UNIV</fcode>
  <fclName>spot, building, farm</fclName>
  <fcodeName>university</fcodeName>
  <population/>
</geoname>
```

Neben den GPS-Koordinaten der Lokation, sind in einem Eintrag Landbezeichnung (<countryCode>, <countryName>), Klassifizierung (<fcl>, <fclName>) und Subklassifizierung der Lokation (<fcode>, <fcodeName>) sowie, falls es sich um ein bewohntes Gebiet handelt, eine eventuelle Populationsangabe zu finden.

A.4. NMEA-0183-Protokoll

Das NMEA-0183-Protokoll ist ein von der National Marine Electronics Association (NMEA) im März 1983 freigegebener zur Kommunikation in der Schiffart eingesetzter Standard [nme]. Viele der eingesetzten GPS-Geräte (beispielsweise sogenannte *GPS-Mäuse*) verwenden zur Kommunikation ebenso den NMEA-0183 Standard. Dieser besteht aus der Definition der RS-422-Schnittstelle [ANSI/TIA/EIA-422-B] sowie der Definition von Datensatz-Formaten. Ein Datensatz kann bis zu 80 ASCII-Zeichen enthalten, beginnt jeweils mit einem $-Symbol und endet mit <CR><LEF>. Dem $-Symbol folgt die Geräte-Id, bestehend meist aus zwei Zeichen, direkt gefolgt von einer Datensatz-Id (meist drei Zeichen). Im Anschluss folgen mehrere durch Kommata getrennte Datenfelder, gefolgt von einer optionalen durch „*“ getrennten Prüfsumme als Hexadezimalzahl.

Im Folgenden sind zwei der in SmartGPS verwendeten Datensatz-Formate vorgestellt: Zum einen der *Recommended Minimum Sentence C* (RMC), welcher von allen NMEA-Geräten implementiert werden muss, zum Anderen *Global Positioning System Fix Data* (GGA). Neben den reinen Positionsdaten enthält ein GGA-Datensatz zusätzliche Angaben bezüglich Genauigkeit und Qualität der Lokationsdaten.

A.4.1. RMC

Ein RMC-Datensatz besitzt folgenden Aufbau:

$GPRMC,HHMMSS,A,BBBB.BBBB,b,LLLLL.LLLL,l,GG.G,RR.R,DDMMYY,M.M,m*CC

Beispiel: $GPRMC,231609,A,4844.4156,N,0906.2351,E,0.0,0.0,231208,0.4,E*13

Symbol	**Bedeutung**	**Beispiel**
GP	Präfix GP: Global Positioning System	
RMC	Präfix RMC: Recommended Minimum Sentence C	
HHMMSS	Uhrzeit der Positionsbestimmung	23:16:09 (UTC-Zeit)
A	Status: A für OK, V für Warnung	A
BBBB.BBBB	Breitengrad	48° 44′ 41.56″
b	Ausrichtung (N für Norden, S für Süden)	Nord
LLLL.LLLL	Längengrad	9° 6′ 23.51″
l	Ausrichtung (E für Osten, W für Westen)	Ost
GG.G	Geschwindigkeit in Knoten	0.0
R.R	Richtung in Grad	0.0
DDMMYY	Datum	23.12.2008
M.M	Magnetische Abweichung	0.4°
m	Ausrichtung der Abweichung (E oder W)	E
CC	Zwei Hexadezimalziffern als Prüfsumme	13

A.4.2. GGA

Ein GGA-Datensatz baut sich wie folgt auf:

$GPGGA,HHMMSS.ss,BBBB.BB,b,LLLL.LL,l,Q,NN,D.D,H.H,h,G.G,g,A.A,RRRR*CC

Beispiel: $GPGGA,191410,4844.4156,N,0906.2351,E,1,04,4.7,378.3,M,12.0,M,,*45

Symbol	Bedeutung	Beispiel
GP	Präfix GP: Global Positioning System	
GGA	Präfix GGA: Global Positioning System Fix Data	
HHMMSS.ss	Uhrzeit der Positionsbestimmung	23:16:09 (UTC-Zeit)
BBBB.BBBB	Breitengrad	48° 44′ 41.56″
b	Ausrichtung (N für Norden, S für Süden)	Nord
LLLL.LLLL	Längengrad	9° 6′ 23.51″
l	Ausrichtung (E für Osten, W für Westen)	Ost
Q	GPS-Qualität: - 0 für ungültig - 1 für GPS fix - 2 für DGPS fix - 6 für geschätzt (ab Version 2.3)	1
NN	Anzahl der benutzen Satelliten	4
D.D	Horizontale Abweichung (dilution of precision)	4.7
H.H	Höhe der Antenne über Meer (über Geoid)	378.3
h	Einheit der Antennenhöhe	Meter
G.G	Höhe des Geoid über des WGS84-Ellipsoid (geoidal separation)	12.0
g	Einheit der geoidal separation	Meter
A.A	Alter des DGPS-Datensatzes in Sekunden	-
RRRR	DGPS-Referenzstation (0000 bis 1023)	-
CC	Prüfsumme	45

Literaturverzeichnis

[ABC08] *Mobile phone users top 3.3 billion: report*. Website, Online: `http://www.abc.net.au/news/stories/2008/05/24/2254494.htm`, May 2008

[AHU83] Aho, A.V. ; Hopcroft, J.E. ; Ullman, J. D.: *Data Structures and Algorithms*. Addison Wesley, 1983. – ISBN 0–201–00023–7

[AMOT90] Ahuja, Ravindra K. ; Mehlhorn, Kurt ; Orlin, James ; Tarjan, Robert E.: Faster algorithms for the shortest path problem. In: *J. ACM* 37 (1990), Nr. 2, S. 213–223. – ISSN 0004–5411

[BD05] Becker, C. ; Dürr, F.: On location models for ubiquitous computing. In: *Personal and Ubiquitous Computing* 9 (2005), S. 20–31

[BHC+05] Busso, C. ; Hernanz, S. ; Chu, C. W. ; Kwon, S. I. ; Lee, S. ; Georgiou, P. G. ; Cohen, I. ; Narayanan, S.: Smart room: Participant and speaker localization and identification. In: *in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP*, 2005

[BM08] Breymann, Ulrich ; Mosemann, Heiko: *JavaME. Anwendungsemtwicklung für Handys, PDA und Co.* Carl Hanser Verlag, 2008. – ISBN 978–3–446–41376–4

[CK00] Chen, G. ; Kotz, D.: *A Survey of Context-Aware Mobile Computing Research*, 2000. – Technical Report: TR2000-381

[Com08] *Mobile World: Geschäftsmodell der Mobile Carrier auf dem Prüfstand*. Website, Online: `http://www.computerwoche.de/knowledge_center/wireless/1855043`, February 2008

[Dij59] Dijkstra, Edsger W.: A note on two problems in connexion with graphs. In: *Numerische Mathematik* 1 (1959), S. 269–271

[Dom01] Domnitcheva, Svetlana: Location Modeling: State of the Art and Challenges. In: *In Proceedings of the Workshop on Location Modeling for Ubiquitous Computing*, Springer, 2001, S. 13–19

[DR03] Dürr, F. ; Rothermel, K.: On Location Model for Fine-Grained Geocast. In: *UbiComp 2003: Ubiquitous Computing*, 2003

[Dro03] Drosdol, T.: *Unterstützung symbolischer Koordinaten im Lokationsmanagement*, Univerity of Stuttgart, Germany. 2003

[Dud06] Dudzik, Stefan: *Visuelle Modellierung von kontextsensitiven Prozessen in ubiquitären Umgebungen*, Univerity of Stuttgart, Germany. 2006

[geo] *GeoNames*. Website, Online: `http://www.geonames.org/`,

[HNR68] Hart, P.E. ; Nilsson, N.J. ; Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In: *IEEE Transactions on Systems Science and Cybernetics* 4 (1968), July, S. 100–107. – ISSN 0536–1567

[JMB01] Johnson, D. ; Maltz, D. ; Broch, J.: DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. In: *Ad Hoc Networking* (2001), S. 139–172

[KV03] Ko, Young-Bae ; Vaidya, Nitin H.: Anycasting-based protocol for geocast service in mobile ad hoc networks. In: *Comput. Netw.* 41 (2003), Nr. 6, S. 743–760. `http://dx.doi.org/http://dx.doi.org/10.1016/S1389-1286(02)00437-1`. – DOI http://dx.doi.org/10.1016/S1389–1286(02)00437–1. – ISSN 1389–1286

[Leo98] Leonhardt, U.: *Supporting Location-Awareness in Open Distributed Systems*, Department of Computing, Imperial College, London, Diss., May 1998

[LL07] Ludewig, J. ; Lichter, H.: *Software Engineering*. 1. dpunkt.Verlag, 2007. – ISBN 3–89864–165–1

[LR00] Leymann, Frank ; Roller, Dieter: *Production workflow: concepts and techniques*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2000. – ISBN 0–13–021753–0

[Maa98] Maas, H.: Location-aware mobile applications based on directory services. In: *Mobile Networks and Applications* 3 (1998), August, S. 157–173. – ISSN 1383–469X

[NI97a] Navas, J. C. ; Imielinski, T.: GeoCast — geographic addressing and routing. In: *Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*, 1997, S. 66 – 76

[NI97b] Navas, Julio C. ; Imielinski, Tomasz: GeoCast—geographic addressing and routing. In: *MobiCom '97: Proceedings of the 3rd annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA : ACM, 1997. – ISBN 0–89791–988–2, S. 66–76

[nme] *NMEA*. Website, Online: `http://www.nmea.org`,

[RN03] *Kapitel* Problem Solving. In: Russell, Stuart ; Norvig, Peter: *Artificial Intelligence: A Modern Approach*. 2nd edition. Prentice-Hall, Englewood Cliffs, NJ, 2003

[Rol95] Roller, Dieter: *CAD. Effiziente Anpassungs- und Variantenkonstruktion*. Springer, 1995. – ISBN 3–54058–779–9

[Roto2] Roth, J.: *Mobile Compting: Grundlagen, Technik, Konzepte*. 1. Auflage. dpunkt.Verlag, 2002. – ISBN: 3-89864-165-1

[RT99] Royer, E. M. ; Toh, C-K: A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks. In: *IEEE Personal Communications Magazine* 6 (1999), April

[Saa90a] Saaty, Thomas L.: *The Analytic Hierarchy Process - Planing, Priority Setting, Ressource Allocation*. RWS Publications, 1990. – ISBN 0–9620317–2–0

[Saa90b] Saaty, Thomas L.: How to make a decision: The analytic hierarchy process. In: *European Journal of Operational Research* 48 (1990), September, Nr. 1, S. 9–26

[Sato5] Satoh, Ichiro: A Location Model for Pervasive Computing Environments. In: *PerCom*, 2005, S. 215–224

[Scho1] Schöning, U.: *Theoretische Informatik - Kurzgefasst*. 4. Spektrum Akademischer Verlag GmbH, 2001. – ISBN 3–8274–1099–1

[TS06] *Kapitel* Naming. In: Tanenbaum, Andrew S. ; Steen, Maarten van: *Distributed Systems. Principles and Paradigms.* 2. Prentice Hall International, 2006. – ISBN 0132392275

[UBR06] Urbanski, Stephan ; Becker, Christian ; Rothermel, Kurt: Sentient Processes - Process-based Applications in Pervasive Computing. In: *PERCOMW '06: Proceedings of the 4th annual IEEE international conference on Pervasive Computing and Communications Workshops*. Washington, DC, USA : IEEE Computer Society, 2006. – ISBN 0–7695–2520–2, S. 608

[UHSR06] Urbanski, Stephan ; Handte, Marcus ; Schiele, Gregor ; Rothermel, Kurt: Experience using Processes for Pervasive Applications. In: *GI Jahrestagung (1)*, 2006, S. 60–64

[Wei91] Weiser, Mark: The Computer for the 21st Century. In: *Scientific American* (1991), February. `http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html`

[wgs04] Department of Defense World Geodetic System 1984 – Its Definition and Relationships with Local Geodetic Systems. Bethesda, U.S.A. : National Imagery and Mapping Agency, 2004. (NIMA Technical Report). – Forschungsbericht. – 3rd amended edition, Online: `http://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html`

[wtk] *Java ME Technology APIs & Docs*. Website, Online: `http://java.sun.com/javame/reference`,

[Zan76] Zangenmeister, Christof: *Nutzwertanalyse in der Systemtechnik – Eine Methodik zur multidimensionalen Bewertung und Auswahl von Projektalternativen*. 4. Zangemeister & Partner, 1976

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Eduard Huber)