Bereitstellung von Umgebungsinformationen und Positionsdaten für ortsbezogene Dienste in Gebäuden

Moritz Kessel

München 2013

Bereitstellung von Umgebungsinformationen und Positionsdaten für ortsbezogene Dienste in Gebäuden

Moritz Kessel

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Moritz Kessel

1. Berichterstatter:	Prof. Dr. Claudia Linnhoff-Popien, LMU München
2. Berichterstatter:	Prof. Dr. Uwe Baumgarten, TU München
Tag der Einreichung:	10.06.2013
Tag der Disputation:	22.07.2013

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

. . . . . .

(Name, Vorname)

Mit dem Aufkommen und der steigenden Verbreitung von Smartphones, haben ortsbezogene Dienste einen festen Platz im täglichen Leben vieler Nutzer erhalten. Dabei werden auf Basis des Aufenthaltsortes gezielt Informationen gefiltert, Umgebungsinformationen verfügbar gemacht oder Suchergebnisse nach Lokalität bewertet. Zudem werden bestimmte Dienste, wie mobile Routenfindung und Navigation, ermöglicht. Viele Dienste beziehen nicht nur die Position eines Nutzers mit ein, sondern erlauben es, die Position von Freunden anzuzeigen oder automatische Benachrichtigungen beim Betreten bestimmter Regionen zu erzeugen. Erfordert ein ortsbezogener Dienst eine hohe Positionsgenauigkeit, so wird die Position globale Satellitennavigationssysteme bestimmt.

Auch in großen komplexen Gebäuden, wie Museen, Flughäfen oder Krankenhäusern, besteht Bedarf an ortsbezogenen Informationen. Beispiele hierfür sind die Suche nach einem speziellen Ausstellungsstück im Museum, die Navigation zum richtigen Gate am Flughafen oder das Treffen mit einem Freund im selben Gebäude. Solche ortsbezogene Dienste in Gebäuden werden im folgenden auch mit dem englischen Begriff Indoor-Location Based Services (I-LBS) bezeichnet. Sie vereinfachen in vielen Situationen unser Leben und werden zukünftig eine ähnliche Verbreitung wie herkömmliche ortbezogene Dienste erlangen. Derzeit existiert jedoch keine Lösung, die I-LBS flächendeckend ermöglicht. Dazu gibt es vor allem zwei Gründe: Zum einen gibt es im Gegensatz zu Außenbereichen keine allgemein verfügbare Kartenbasis. Die Baupläne sind oftmals unter Verschluss und eignen sich mehr für die Planung und Überwachung von Baumaßnahmen als für den semantischen Informationsgewinn. Zum anderen ist der Empfang von Satellitensignalen in Gebäuden so schlecht, dass damit im allgemeinen keine genügend genaue Position bestimmt werden kann. Eine alternative kostengünstige und überall verfügbare Positionsbestimmung von genügend hoher Genauigkeit existiert derzeit nicht.

In dieser Arbeit werden Lösungsmöglichkeiten für beide Probleme vorgestellt und evaluiert, die einem Nutzer eine vergleichbare Dienstnutzung erlauben sollen, wie er es in Außenbereichen bereits gewöhnt ist. Anhand der Anforderungen von I-LBS und Ortungssystemen werden zwei verschiedene Umgebungsmodelle entwickelt. Eines basiert auf der Geography Markup Language (GML) und bietet eine flexible Vektor-basierte Repräsentation eines Gebäudes mit hierarchischen und Graph-basierten Elementen. Zudem wird die vollautomatische Erzeugung eines solchen Modells aus Bauplänen vorgestellt, die einen weiteren Schritt zur flächendeckenden Bereitstellung von Plänen für I-LBS darstellt. Das andere Modell basiert auf einer Bitmap als Raster-basierter Kartendarstellung, welche mithilfe von Bildbearbeitungsalgorithmen und Konventionen in der Farbgebung semantisch angereichert wird. Auch hier werden Möglichkeiten zur automatischen Erzeugung des semantischen Modells, beispielsweise aus abfotografierten Fluchtplänen, erörtert. In einem letzten Schritt werden beide Modelle in einem flexiblen hybriden Umgebungsmodell kombiniert, um Anfragen je nach Datenbasis möglichst effizient beantworten zu können.

Die Positionsbestimmung in Gebäuden wird anhand von einigen Verbesserungen für Fingerprinting-Ansätze auf Smartphones behandelt. Das Fingerprinting basiert dabei entweder auf Kamerabildern oder auf WLAN-Signalen. Zudem werden zusätzliche Sensoren, wie Kompass und Beschleunigungssensor, zur Verbesserung der Genauigkeit und Robustheit hinzugenommen. Um die Positionsbestimmung für den Einsatz in I-LBS verfügbar

zu machen, ist jedoch nicht nur eine hohe Genauigkeit, sondern vor allem eine große Flexibilität die Hauptanforderung. Zu diesem Zweck wurde ein Ansatz entwickelt, welcher ohne Nutzerinteraktion allein auf Basis von Kartenmaterial und inertialen Sensoren ein oder mehrerer Nutzer eine Fingerprint-Datenbank erzeugt, welche anderen Nutzern zur Verfügung gestellt werden kann. Mit dem Ziel der Kosten- und Komplexitätsreduktion, sowie der Lösung des Problems der Aktualität von Daten in Fingerprint-Datenbanken, hilft der Ansatz bei der automatischen flächendeckenden Ausbringung von Referenzdaten zur Positionsbestimmung.

Um die Brücke zwischen I-LBS und LBS zu schlagen, reicht es allerdings nicht aus, beide Arten von Diensten getrennt zu betrachten. Eine nahtlose Dienstnutzung muss möglich sein und somit werden sowohl eine nahtlose Positionsbestimmung, als auch eine nahtlose Bereitstellung von Kartenmaterial notwendig. Zu diesem Zweck wurde ein Plattform entwickelt, welche auf Basis einer Sensorbeschreibungssprache automatisch die Auswahl und Kombination der zu nutzenden Sensoren zur Positionsbestimmung ermittelt. Zudem verfügt die Plattform über eine Komponente, die auf Basis der Positionsdaten passende Umgebungsmodelle zur Verfügung stellt und die Transformation von Positionsdaten zwischen verschiedenen Modellen ermöglicht.

With the growing spread of smartphones, location based services (LBS) found their place in our every-day life. Location information is used for filtering relevant information or increasing the user's awareness of the surrounding environment. Some services such as navigation or friend spotter only become possible with the help of accurate and timely location information, which is offered by the Global Positioning System (GPS).

Large and complex buildings such as museums, airports or hospitals offer also a plenty of opportunities for specialized Indoor-LBS (I-LBS). These indoor services can ease the our life in many situations and will spread as fast as their outdoor counterparts as soon as the remaining two obstacles are resolved. One obstacle is the limited availability of indoor map data. While national and private provider offer a full range of maps for streets, floor plans are often the intellectual property of the architect and neither publicly available nor in a format suitable for semantic information gain. Another problem is the shortage of accurate location information in indoor areas on smartphones. A cheap and globally available alternative to GPS does not yet exist.

This thesis offers and evaluates approaches to solve both problems while offering users the same experience as they are accustomed from outdoor areas. First, the topic of indoor maps is discussed. Two different models with complementary strengths and weaknesses are introduced: One is based on the Geography Markup Language (GML) and offers a flexible vector-based representation of buildings combining hierarchical and graph-based components. The automated creation of such a semantic model from plain CAD-data is also presented as a step towards the provision of map data for indoor-LBS. The other model is based on bitmaps as grid representation of single floors which are semantically enriched with certain colors by image processing methods. Finally, both models are combined in a hybrid model for efficiently answering queries depending on the available data.

For accurate indoor positioning different enhancements for fingerprinting systems are presented which work either with WLAN or images. Additional sensors such as compass or accelerometer are added for increased accuracy, precision and robustness. To make indoor positioning fit for indoor-LBS, not only a high accuracy but also a large flexibility is needed. For this reason it is proposed to automatically create a fingerprint database without manual interaction of any user. The approach is based on map knowledge and inertial sensors as an ingredient for crowd-sourced exhaustive provision of reference data.

To bridge the gap between indoor- and outdoor-LBS it is not sufficient to look at both kinds of services separately. A seamless usage has to be provided concerning seamless positioning as well as the seamless provision of map data. This is achieved by a platform for hybrid positioning based on a sensor description language which determines the choice and combination of sensors based on the available sensors and user requirements. What is more, one component of the platform offers suitable environmental information based on the users' location and coordinate transformations between several representations or reference models.

Inhaltsverzeichnis

Danksagung

Ich bedanke mich herzlich bei allen, die mich auf dem Weg zu meiner Dissertation begleitet haben. Allen voran danke ich Frau Prof. Claudia Linnhoff-Popien für die freundliche Aufnahme an Ihrem Lehrstuhl, die vielen spannenden Aufgaben und die vielen wertvollen Hinweise zu meiner Arbeit. Herrn Prof. Uwe Baumgarten danke ich für die Übernahme der Zweitberichterstattung, die vielen wertvollen Hinweise zu meiner Arbeit und die interessanten Gespräche bei meinen Besuchen.

Ich bedanke mich bei meinen Kollegen Dr. Martin Werner, Philipp Marcus, Florian Gschwandtner, Kevin Wiesner, Kim Schindhelm, Kay Weckemann, Robert Lasowski, Markus Duchon, Anna Schwanengel, Michael Dürr, Florian Dorfmeister, Marco Maier, Mirco Schönfeld, Sebastian Feld, Thomas Mair, Thomas Zimmermann, Chadly Marouane, Lorenz Schauer und Dr. Stephan Verclas für die unzähligen ertragreichen Gespräche und Diskussionen sowie die gemeinsamen Publikationen, betreuten Arbeiten, Lehrveranstaltungen und Projekte. Außerdem möchte ich Dr. Peter Ruppel danken, der mich nicht nur bereits während des Studiums für die Arbeit am Lehrstuhl begeistert hat, sondern der mir auch in seiner Zeit am Lehrstuhl stets ein Mentor war.

Zuletzt geht ein ganz besonderer Dank an meine Freundin und meine Familie für die liebevolle Unterstützung und das entgegengebrachte Verständnis und Vertrauen.

Kapitel 1

Einleitung

Mit dem Aufkommen und der steigenden Verbreitung von Smartphones haben auch ortsbezogene Dienste einen festen Platz im täglichen Leben vieler Nutzer erhalten. Als ortsbezogenen Dienst bezeichnet man im Allgemeinen einen IT-Dienst, der bestimmte Informationen auf Basis von Ortsinformationen zusammenstellt oder verarbeitet [73]. Nach einer Studie [150] aus dem Jahr 2012 nutzen bereits 74 % der erwachsenen Smartphone-Nutzer in den USA ortsbezogene Dienste, wobei die Verbreitung von Smartphones in der Nutzergruppe 46 % entspricht. Damit nutzt bereits mehr als ein Drittel der erwachsenen US-Amerikaner ortsbezogene Dienste bei stetig steigender Tendenz. So konnte allein zwischen 2011 und 2012 ein Anstieg in der Nutzung ortsbezogener Dienste um 19 % festgestellt werden. Der Großteil nutzt dabei Ortsinformationen in Echtzeit, wie Navigations- und Informationsdienste. In vielen Diensten gewinnt zudem die soziale Komponente an Bedeutung, wie in geosozialen Check-In-Diensten und ortsbezogenen sozialen Netzen.

Die steigende Verbreitung fördert auch die Diversität der Dienste. Manche Dienste stellen bestimmte Informationen über die direkte Umgebung des Nutzers zur Verfügung, andere bewerten Suchergebnisse nach der Lokalität des Ergebnisses, wieder andere ermöglichen Navigation oder das Auffinden von Personen. Viele Dienste beziehen dabei nicht nur die Position eines Nutzers mit ein, sondern erlauben zudem die Position von Freunden anzuzeigen, dynamisch Treffpunkte zu vereinbaren oder automatische Benachrichtigungen nach bestimmten Regeln für die Aufenthaltsorte zu erzeugen. Dabei unterscheiden sich die Anforderungen an die Positionsbestimmung und die notwendigen Informationen über die Umgebung. Während die meisten Informationsdienste mit einer groben Positionsschätzung auskommen, benötigen beispielsweise Navigationsdienste deutlich genauere Positions- und Kartendaten.

1.1 Motivation

Die meisten ortsbezogenen Dienste beziehen sich explizit auf Außenbereiche, sei es die Navigation im Straßennetz oder Informationen zu Restaurants oder Sehenswürdigkeiten im Umkreis. Jedoch gibt es in Innenbereichen oftmals ebenfalls Bedarf an ortsbezogenen

Informationen. Gerade in großen komplexen Gebäuden können diese Informationen das Leben eines Nutzers deutlich vereinfachen und viel Zeit einsparen. Beispiele hierfür sind die Suche nach einem Ausstellungsstück im Museum oder gar lebensrettender mobiler medizinischer Ausrüstung in Krankenhäusern. Ferner ist die Navigation zum richtigen Gate am Flughafen unter Zeitdruck genauso von Interesse, wie die Vereinbarung eines spontanen Treffpunkts mit einer Freundin in einem unbekannten Gebäude. Die meisten Anwendungsfälle für ortsbezogene Dienste lassen sich problemlos auf Innenbereiche innerhalb von Gebäuden übertragen und führen auch hier zu einer deutlichen Zeit- und Kostenersparnis. Solche ortsbezogenen Dienste in Gebäuden werden im Folgenden auch als Indoor-Location Based Services (I-LBS) bezeichnet.

Bislang existiert jedoch noch keine Lösung, I-LBS flächendeckend auszubringen. Dazu gibt es vor allem zwei Gründe: Zum einen gibt es im Gegensatz zu Außenbereichen keine allgemein verfügbare Kartenbasis. Die Baupläne sind oftmals unter Verschluss und eignen sich mehr für die Planung und Überwachung von Baumaßnahmen als zum semantischen Informationsgewinn. Zum anderen gibt es keine kostengünstige und überall verfügbare Positionsbestimmung, die auch die hohen Genauigkeitsanforderungen von Navigationsanwendungen erfüllt. Diese beiden Problemstellungen werden in dieser Arbeit untersucht und Lösungsmöglichkeiten erarbeitet, welche die technischen Schwierigkeiten bei I-LBS ausräumen und eine schnelle und weite Verbreitung fördern. Dazu sind insbesondere Automatismen zu finden, mit denen die Bereitstellung von Umgebungsinformationen und Positionsdaten vereinfacht wird und die Hürden, die Dienstanbieter bislang überwinden müssen, verringert werden.

1.2 Zielsetzung

Diese Arbeit hat das Ziel, bestehende Probleme ortsbezogener Dienste in Innenbereichen zu lösen, so dass Dienste einem Nutzer so angeboten werden können, wie dieser es aus Außenbereichen gewohnt ist. Die konkreten Teilziele dieser Arbeiten gliedern sich in drei Kategorien: Die möglichst automatische Generierung und effiziente Bereitstellung von Umgebungsinformationen, eine einfache und flexible Positionsbestimmung in Gebäuden von genügender Genauigkeit ohne zusätzliche Kosten und der nahtlose Übergang zwischen Innen- und Außenbereichen.

Für die **automatische Erzeugung und effiziente Bereitstellung von Umgebungsinformationen** sind Mechanismen notwendig, mit denen diese automatisch aus bereits existierenden Daten, wie CAD Daten oder Fluchtplänen, gewonnen werden. Zudem muss für Umgebungsmodelle ein einheitlicher Standard geschaffen werden, in dem die für I-LBS notwendigen Informationen analog zu bestehenden Kartendaten bereit gestellt werden. Zu diesem Zweck müssen die Modelle so spezifiziert werden, dass sie alle wichtigen Informationen für ortsbezogene Dienste beinhalten. Um die Verfügbarkeit von Umgebungsmodellen zu erhöhen, muss die Erzeugung eines solchen Modells automatisiert werden. Eine effiziente Bereitstellung der Umgebungsdaten ist dabei eine weiter Grundvoraussetzung für skalierbare I-LBS. Hierfür sollen verschiedene Methoden zur Verfügung stehen,

Umgebungsmodelle auf die mobilen Endgeräte zu übertragen. Ebenso ist eine serverseitige Anfragebearbeitung in den Fällen notwendig, in welchen eine direkte Übertragung des Modells nicht gewünscht wird.

Um eine **einfache und flexible Positionsbestimmung in Gebäuden von genügender Genauigkeit und ohne zusätzliche Kosten** für I-LBS zu ermöglichen, müssen passende Technologien erforscht werden. Nur Methoden, die weitgehend unabhängig von den Gegebenheiten vor Ort sind, ermöglichen eine globale Dienstverfügbarkeit ähnlich zu bestehenden Außendiensten. Die Unabhängigkeit bezieht sich auch auf die benötigte technische Infrastruktur. Zudem muss die Positionsbestimmung mit dem Smartphone erfolgen, um die Notwendigkeit zusätzlicher Sensorik und somit zusätzlicher Kosten für den Nutzer zu vermeiden. Ist solch eine Lösung mit den aktuellen technischen Gegebenheiten nicht möglich, so ist eine Lösung zu finden, die in Bezug auf Verfügbarkeit, Aufwand und Kosten den angestrebten Zielen am nächsten kommt.

Der nahtlose Übergang zwischen Innen- und Außenbereichen ist notwendig, um eine Dienstbereitstellung ohne Bruch und Verlust der Verfügbarkeit bei solchen Übergängen zu ermöglichen. Dadurch wird die Nutzung von LBS transparent für alle Bereiche ermöglicht und somit die Brücke zwischen I-LBS und herkömmlichen LBS geschlagen. Auch die kontinuierliche Nutzung von Diensten über bestehende Grenzen hinweg wird so umsetzbar. Ein Beispiel hierfür ist die Navigation von Büro zu Büro über verschiedene Gebäude, ja sogar Städte oder Länder hinweg. So ist es das Ziel, den Unterschied zwischen ortsbezogenen Diensten in Innen- und Außenbereichen aus Nutzersicht sukzessive zu beseitigen. Aufgrund der unterschiedlichen technischen Gegebenheiten werden dennoch verschiedene Mechanismen auf Dienstanbieterseite zum Einsatz kommen.

1.3 Aufbau der Arbeit

Der Aufbau dieser Arbeit orientiert sich an den im vorangehenden Abschnitt genannten Zielen und wird im Folgenden kurz vorgestellt.

Zunächst werden in Kapitel 2 die technischen Grundlagen erörtert und es wird auf bestehende Arbeiten aus den Themengebieten der einzelnen Ziele eingegangen. Im Kapitel 2 werden zunächst die bekannten Methoden zur Positionsbestimmung, wie Triangulation, Fingerprinting und Koppelnavigation vorgestellt. Nach einer Einführung in die Techniken der Sensorfusion folgt die Vorstellung verschiedener Technologien zur Positionsbestimmung in Gebäuden. Anschließend werden verschiedene Formen von Umgebungsmodellen vorgestellt und auf die Anforderungen an Umgebungsmodelle und Positionsbestimmung aus Sicht der ortsbezogenen Dienste eingegangen.

In Kapitel 3 folgt ein Beitrag zum Themengebiet der Umgebungsmodelle, wobei zunächst ein Vektor-basiertes und ein Raster-basiertes Umgebungsmodell vorgestellt werden. Anschließend werden beide Modelle in einem hybriden Modell vereint, um die Anforderungen von Dienstseite und der Positionsbestimmung optimal zu erfüllen.

Kapitel 4 beschäftigt sich mit der Thematik der Positionsbestimmung in Gebäuden. Dabei wird zuerst ein Ansatz zur infrastrukturlosen relativen Positionsbestimmung vorge-

stellt und evaluiert. Anschließend wird auf zwei Probleme dieser Lösung eingegangen, die durch die Einbeziehung einer bestehenden Infrastruktur gelöst werden. Dazu werden neue Methoden zur absoluten Positionsbestimmung mithilfe der Wiedererkennung bestimmter Orte durch natürliche Marker und auf Basis von WLAN-Fingerprinting vorgestellt.

Kapitel 5 verfolgt das Ziel einer einfachen Verbreitung von Positionsdaten und einer nahtlosen Dienstanbietung durch verschiedene Mechanismen. Zunächst wird ein Mechanismus vorgestellt, welcher die Einbeziehung von vorhandener Infrastruktur durch Fingerprinting weitestgehend automatisiert und somit die Verbreitung von Positionsdaten fördert. Anschließend wird eine Plattform vorgestellt, welche die dynamische Kombination von verfügbarer Sensorik zur Laufzeit ermöglicht und auf wechselnde Umgebungen und Sensoren reagieren kann. Damit kann bereits eine nahtlose Bereitstellung von Umgebungs- und Positionsdaten sowie eine nahtlose Dienstnutzung erfolgen. Zuletzt folgt ein Ansatz zur expliziten Übergangserkennung zwischen Außen- und Innenbereichen, um während des Übergangs genauere Positionsdaten zur Verfügung stellen zu können.

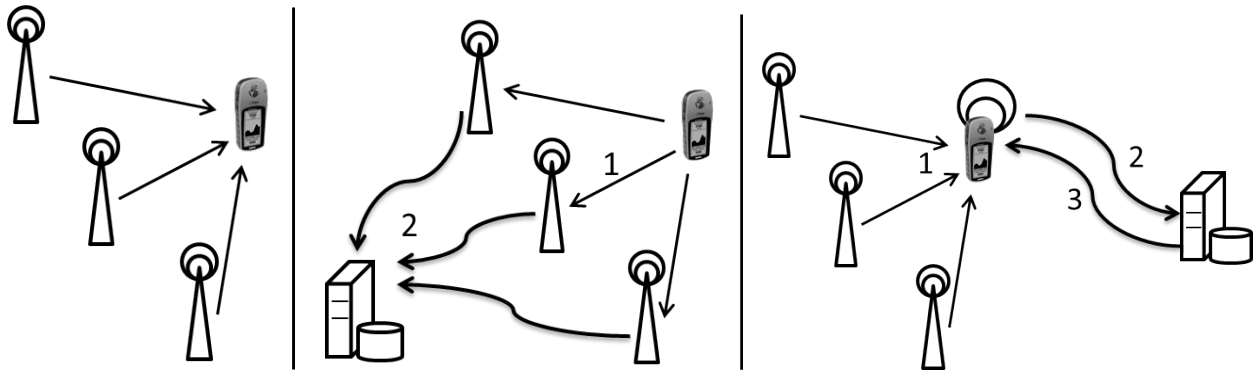
Kapitel 6 fasst schließlich die wichtigsten Ergebnisse dieser Arbeit zusammen und bietet einen Ausblick auf die weitere Entwicklung und weiterführende Ideen in dem Forschungsgebiet der ortsbezogenen Dienste in Gebäuden.

Kapitel 2

Grundlagen und Stand der Technik

In den letzten Jahren wurde das Thema der Positionsbestimmung innerhalb von Gebäuden aus verschiedenen Fachgebieten erforscht, wobei unterschiedliche Anwendungsfälle, Ziele und Schwerpunkte im Fokus lagen. Frühe Arbeiten auf diesem Gebiet beschäftigen sich vielfach mit der Lokalisierung von Robotern [32], aber auch mit der Ortung von Personen [133]. Die Randbedingungen unterschieden sich jedoch erheblich, da die Roboter mit einer Vielzahl an Sensoren und einer leistungsfähigen Berechnungseinheit ausgestattet waren, um die eigene Position selbstständig bestimmen zu können. Personen wurden im Gegensatz dazu mit günstigen Sendern, wie Infrarot-, Ultraschall- oder Radiofrequenzsendern, ausgestattet. Die Signale der Sender wurden in einer leistungsstarken Infrastruktur aufgefangen und ausgewertet [134]. Diese beiden grundlegenden Ansätze werden als *Endgeräte-zentrische* bzw. *Infrastruktur-basierte* Positionsbestimmung bezeichnet [73]. Im Laufe der Zeit und der immer größeren Verfügbarkeit von mobilen Endgeräten mit wachsender Rechenkapazität und Kommunikationsmöglichkeiten entwickelte sich zudem ein hybrider Ansatz, in welchem die Sensoren zum mobilen Endgerät gehören und die Messungen direkt vom Endgerät erhoben werden. Diese Daten werden jedoch an eine leistungsstarke Infrastruktur zur Auswertung und Positionsbestimmung übertragen. Die ermittelte Position wird dem Endgerät anschließend zur weiteren Verarbeitung zur Verfügung gestellt. Dieser hybride Ansatz wird auch als *Endgeräte-unterstützte* Positionsbestimmung bezeichnet [73]. Die drei Ansätze werden in Abbildung 2.1 dargestellt.

Alle aktuellen Systeme zur Positionsbestimmung lassen sich in diese drei Ansätze einordnen, wobei die Ansätze unterschiedliche Vor- und Nachteile haben und in unterschiedlichen Domänen skalieren. Endgeräte-zentrierte Ansätze haben üblicherweise einen hohen Energieverbrauch aufgrund von Messungen und Berechnungen auf dem Endgerät. Allerdings erlauben sie dafür einen größtmöglichen Schutz der Privatsphäre, da die Positionsdaten nur auf dem Endgerät vorliegen. Da jeder Nutzer seine Position selbst berechnet, ist der zentrale Verwaltungsaufwand unabhängig von der Nutzerzahl. Dafür muss aber jeder Nutzer über ein leistungsfähiges und somit meist teures Endgerät verfügen. Diese Kosten pro Endgerät sind im Fall eines Infrastruktur-basierten Ansatzes aufgrund der geringeren Anforderungen an das Gerät deutlich geringer. Dafür müssen die Kapazitäten der Infrastruktur an die Nutzeranzahl angepasst werden. Um dem Nutzer seine Positionsdaten zur



(a) Endgeräte-zentrisch (b) Infrastruktur-basiert (c) Endgeräte-unterstützt

Abbildung 2.1: Verschiedene Ansätze zur Positionsbestimmung

Verfügung zu stellen, ist zudem ein funktionsfähiger Kommunikationskanal notwendig, was nicht in allen Anwendungsszenarien sichergestellt werden kann. Zudem ist die Positionsinformation auf jeden Fall in der Infrastruktur vorhanden und somit außerhalb der Kontrolle des Nutzers. Ein Endgerät-unterstützter Ansatz benötigt ebenfalls einen Kommunikationskanal. Der Nutzer hat keine vollständige Kontrolle über die Positionsinformation, und eine größere Zahl an Nutzern resultiert sowohl in mehr Endgeräten, als auch in einer größeren Belastung der Infrastruktur. Zudem kann es aufgrund der notwendigen Datenübertragung zu einer zusätzlichen Latenzzeit kommen. Dafür lassen sich durch diesen Ansatz auch Positionierungsmethoden umsetzen, welche das Endgerät bezüglich Rechenkapazität oder Speicherauslastung überfordern, und die zur Positionsbestimmung benötigten Daten, wie Kartenmaterial oder Trainingsdaten für Patternmatching, müssen dem Endgerät nicht zur Verfügung stehen.

Im Folgenden wird zunächst eine Einführung in vier grundlegende Methoden zur Positionsbestimmung gegeben, namentlich Triangulation, Musterabgleich, Nahbereichsentdeckung und Koppelnavigation, in welche sich alle bestehenden Systeme einordnen lassen. Diese wird durch die Beschreibung von Sensorfusionsmechanismen ergänzt und durch die Vorstellung bestehender Systeme zur Positionsbestimmung abgerundet. Anschließend wird auf den Bedarf an Umgebungsinformationen eingegangen und es werden verschiedene Arten von Umgebungsmodellen vorgestellt. Den letzten Teil des Kapitels bildet eine Einführung in die verschiedenen Arten von ortsbezogenen Diensten, deren Anforderungen und die Herausforderungen zur Diensterbringung innerhalb von Gebäuden.

2.1 Methoden zur Positionsbestimmung

Neben der Unterscheidung, wo Sensordaten gemessen und wo die Positionsbestimmung durchgeführt wird, gibt es noch eine ganze Reihe von weiteren Klassifikationsmöglichkeiten für Systeme zur Positionsbestimmung. Hightower und Boriello unterscheiden in [49] drei grundlegende Methoden zur Positionsbestimmung: Triangulation, Szenenanalyse oder

Musterabgleich und Nahbereichsentdeckung, auch als Presence oder Proximity Detection bekannt. In der letzten Zeit findet man zudem immer häufiger Methoden, die auf Koppelnavigation, auch Dead Reckoning genannt, basieren und sich als solches keiner der obigen drei Methoden zuordnen lassen. Sämtliche in den letzten Jahren entwickelte Systeme zur Positionsbestimmung lassen sich einer oder mehreren der vier Methoden zuordnen. Aus diesem Grund ist das grundlegende Verständnis der Methoden eine wichtige Voraussetzung zur Bewertung von Positionsbestimmungssystemen und deren zu erwartenden Eigenschaften. Im Folgenden werden die Methoden vorgestellt und bewertet.

2.1.1 Triangulation

Unter Triangulation versteht man die Positionsbestimmung anhand der geometrischen Eigenschaften eines Dreiecks. Dabei wird die Entfernung oder der Winkel des Ziels zu mindestens drei bekannten nicht linear abhängigen Referenzpunkten bestimmt und somit eine theoretisch eindeutige Position in einer Ebene ermittelt. Bei vier oder mehr Referenzpunkten, die nicht in einer Ebene liegen, lässt sich sogar eine Position in drei Dimensionen bestimmen. Da die Messung jedoch meist mit einem Fehler behaftet ist, kommt als Aufenthaltsort des Ziels nicht ein Punkt, sondern eine gewisse Region in Frage, die durch Ausgleichsrechnung zu dem wahrscheinlichsten Aufenthaltsort reduziert werden kann. Die Bestimmung der Position anhand von Winkeln wird im allgemeinen als Angulation bezeichnet, während die Nutzung von Entfernungsmessungen Lateration genannt wird. Natürlich ist aufgrund der Regeln der Trigonometrie auch eine Kombination von Winkel- und Entfernungsmessungen bezüglich dreier Referenzstationen oder eine kombinierte Winkel- und Entfernungsmessung zu einer Station für die Positionsbestimmung ausreichend.

Triangulation bietet das beste Beispiel für eine Methode zur Positionsbestimmung, die entweder Endgeräte-zentrisch, Endgeräte-unterstützt oder Infrastruktur-basiert ablaufen kann. Im ersten Fall muss das Endgerät die Messung von Pilotsignalen der Referenzstationen übernehmen und deren Position kennen, wobei diese auch im Signal kodiert werden kann. Das beste Beispiel hierfür ist GPS. In einer Endgeräte-unterstützten Realisierung wird der Zeitstempel, die Signalstärke oder bereits die Entfernungsschätzung an eine Infrastruktur geschickt, der die Positionen der Referenzstationen bekannt sind. Dort wird die Position anhand der Signale berechnet und an das Endgerät zurückgeschickt. Für die Infrastruktur-basierte Umsetzung muss das Endgerät Signale senden, die in diesem Fall von den Referenzstationen empfangen werden und zur Positionsbestimmung in der Infrastruktur nutzbar sind. Um die einzelnen Entfernungsschätzungen untereinander auszutauschen, müssen die Referenzstationen miteinander kommunizieren.

Lateration

Lateration bezeichnet die Positionsbestimmung aufgrund von Entfernungsschätzungen des Ziels zu Referenzstationen mit bekannter Position. Dabei können verschiedene Arten von Signalen und Signaleigenschaften genutzt werden, um eine Entfernung zu ermitteln. Grundsätzlich eignet sich dazu jedes Signal, dessen Ausbreitungsgeschwindigkeit bekannt ist, wie

Licht, Funksignale oder Schall, da somit durch die Zeit, die zwischen Senden und Empfangen des Signals verstreicht, die Entfernung ermittelt werden kann. Mögliche Störquellen sind hier durch mangelnde Synchronisation zwischen Sender und Empfänger, Uhrenversatz, Mehrwegeausbreitung und Reflexion gegeben. Des Weiteren kann eine Entfernung auch aufgrund der Signalstärke ermittelt werden, wenn die Sendeleistung und der Pfadverlust bekannt sind. Hierbei werden weder genaue Uhren noch Synchronisation benötigt. Dafür sind wieder die Mehrwegeausbreitung, aber auch Streuung, Überlagerung und Abschattung problematisch. Zudem hängen die Pfadverlusteigenschaften von den physikalischen Gegebenheiten in der Umgebung ab. So kann eine hohe Luftfeuchtigkeit Radiowellen im 2,4 GHz-Bereich aufgrund des Dipoleffekts von Wasser stark abschwächen.

Kennt man die Distanz des Ziels zu einer Referenzstation, so befindet sich dieses auf einer Kreislinie bzw. Kugeloberfläche um die Station mit der Entfernung als Radius. Im ungestörten Fall schneiden sich zwei Kreislinien in einem oder zwei Punkten, wird ein dritter Kreis hinzugenommen, so bleibt ein Schnittpunkt übrig. Betrachtet man hingegen den realistischen Fall mit Messfehlern, so müssen sich die Kreise nicht zwingend schneiden. Selbst falls dies der Fall sein sollte, so kommt es bei drei Kreisen eher zu einer Schnittfläche, in der der Aufenthaltsort des Ziels liegen kann (siehe Abbildung 2.2). Diese kann durch die Methode der kleinsten Quadrate iterativ weiter verkleinert werden, bis am Ende eine sehr wahrscheinliche Position des Endgerätes übrig bleibt. Ein ausführliches Beispiel dazu findet sich in [135].

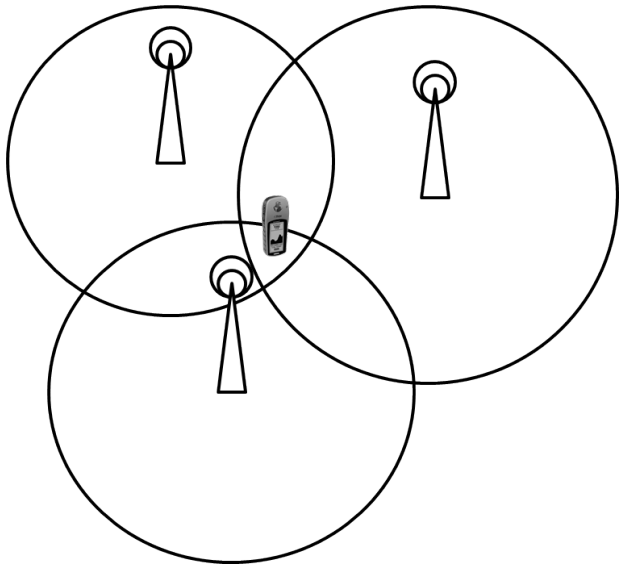


Abbildung 2.2: Zirkuläre Lateration am Beispiel von drei Basisstationen.

Eine spezielle Form der Lateration ist die hyperbolische Lateration. In diesem Fall liegt die Entfernungsschätzung nicht direkt vor, sondern lediglich eine paarweise Differenz der Entfernungen zwischen je zwei Referenzstationen und dem Endgerät. Dies ist beispielsweise der Fall, falls die Uhren zwischen Referenzstationen und Endgerät nicht synchron gehen und somit nicht die Laufzeit der Signale, sondern nur der Zeitversatz berechnet werden

kann. Durch die paarweisen Entfernungsdifferenzen wird jeweils eine Hyperbel definiert, auf der die Position des Endgerätes liegt. Bei drei Stationen entspricht dies drei Hyperbeln, wodurch auch im Fall von gestörten Messungen durch die Methode der kleinsten Quadrate eine eindeutige Position wie im Fall der normalen Lateration bestimmt werden kann.

Im Fall der Lateration kann man drei verschiedenen Methoden unterscheiden:

- Im Fall von *Time-of-Arrival* werden die Signale zu einem bekannten Zeitpunkt von einer bekannten Position gesendet und die Ausbreitungsgeschwindigkeit der Signale ist bekannt. Damit kann die Zeit zwischen dem Senden und Empfangen des Signal genutzt werden, um auf eine Entfernung zu schließen und die Lateration angewandt werden. Die Entfernungsschätzung muss dabei nicht der Realität entsprechen, da Asynchronität der Uhren, Mehrwegeausbreitung und ähnliches die wahre Laufzeit des Signals verfälschen.

- Im Fall von *Time-Difference-of-Arrival* kann nur der Zeitversatz zwischen dem Eintreffen von gleichzeitig gesendeten Nachrichten ermittelt werden, wodurch nur der Unterschied in der Entfernung zwischen den Referenzstationen bekannt ist. In diesem Fall wird hyperbolische Lateration angewandt.

- Zuletzt gibt es noch den Fall von *Roundtrip-Time-of-Flight*. Hier werden die Signale nach Erhalt vom Empfänger zurück an den Sender geschickt, so dass dieser anhand des Zeitversatzes zwischen Senden und Empfangen die doppelte Entfernung zum Empfänger berechnen kann. Hierbei ist allerdings noch zusätzlich die Verarbeitungszeit des Empfängers bis zum Zurückschicken der Nachricht enthalten. Der große Vorteil ist, dass keine Form von Synchronisation erforderlich ist und die ermittelten Entfernungen immer größer oder gleich der tatsächlichen Entfernung sind. Anhand mehrerer Entfernungsschätzungen kann wieder Lateration angewandt werden.

Angulation

Angulation bezeichnet die Positionsbestimmung durch den Empfangswinkel von Signalen. Dazu bestimmt entweder das Endgerät die Empfangsrichtung der Signale von mindesten zwei Referenzstationen, oder die Signale des Endgerätes werden an den Referenzstationen gemessen. Dabei dürfen die Stationen und das Endgerät nicht auf einer Geraden liegen, damit die Strahlen, die den Weg der Signale von den Referenzstationen zum Endgerät bzw. andersrum beschreiben, sich in genau einem Punkt schneiden. Für die Signale werden meist Radiowellen und Antennenarrays genutzt, aber es existieren auch Verfahren, die mit Laser oder Infrarot Sensoren arbeiten [59]. Mögliche Fehlerquellen liegen darin begründet, dass der Winkel nur mit einer gewissen Genauigkeit bestimmt werden kann und so anstelle von Strahlen Kegel genutzt werden, um die Signalausbreitung zu beschreiben. In diesem Fall ergibt die Schnittfläche der Kegel den möglichen Aufenthaltsbereich des Ziels (siehe Abbildung 2.3). Liegen mehr als zwei Winkelangaben vor, so kann das überbestimmte Gleichungssystem iterativ durch die Methode der kleinsten Quadrate zur wahrscheinlichsten

Lösung gelöst werden. Bei Angulation ergeben sich insbesondere Probleme mit Reflexionen, da so der Aufenthaltsort des Zieles unter einem falschen Winkel angenommen wird.

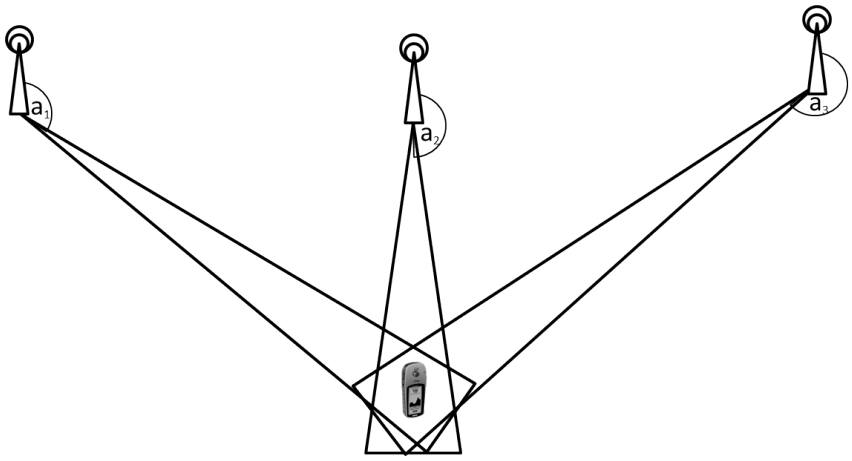


Abbildung 2.3: Angulation mit Winkelbestimmung an drei Basisstationen.

2.1.2 Musterabgleich

Unter Musterabgleich versteht man das Wiedererkennen bestimmter Muster in der Umgebung, was einen Rückschluss auf den aktuellen Aufenthaltsort erlaubt. Diese Muster können visueller Art sein, also Landmarken, auffällige Objekte oder auch künstliche oder natürliche Features in einem Kamerabild, die gemessene Signalstärke von mehreren umliegenden Stationen oder auch spezielle Eigenschaften eines Ortes beinhalten, wie die Reflexion von Schallwellen [127] oder die Verzerrung des Erdmagnetfelds [126]. Im Beispiel von visuellen Systemen sprich man häufig auch von der Szenenanalyse, bei der Wiedererkennung von Signalstärkemustern von Fingerprinting.

Zum Zweck der Positionsbestimmung arbeiten solche Systeme in zwei Phasen. In der ersten Phase werden Muster in der Umgebung aufgenommen, mit Ortsinformationen verknüpft und anschließend als Referenz für die spätere Positionsbestimmung gespeichert. In manchen Fällen kann anstelle einer tatsächlichen Aufnahme das erwartete Muster anhand von Modellen berechnet werden. Im Beispiel der visuellen Muster kann dies durch dreidimensionale Gebäudemodelle oder Baupläne und einen Kantenabgleich erreicht werden [52], im Fall von Signalstärkemustern durch Ausbreitungsmodelle bei bekannter Position der Referenzstationen in der Umgebung und der Position und Abschottung der Wände [10]. Diese Phase wird allgemein auch als Kalibrierungsphase bezeichnet, da hier die Grundlage für die Positionsbestimmung gelegt wird und die Phase meist von Administratoren oder Bereitstellern des Lokalisierungssystems durchgeführt wird, welche die Muster größtenteils manuell mit den richtigen Ortsinformationen verknüpfen. Diese Phase wird bei größeren Bereichen schnell zeitaufwändig. Zudem kann eine erneute Kalibrierung im Fall von Änderungen in der Umgebung notwendig werden, falls diese Änderungen auch die Muster beeinflussen.

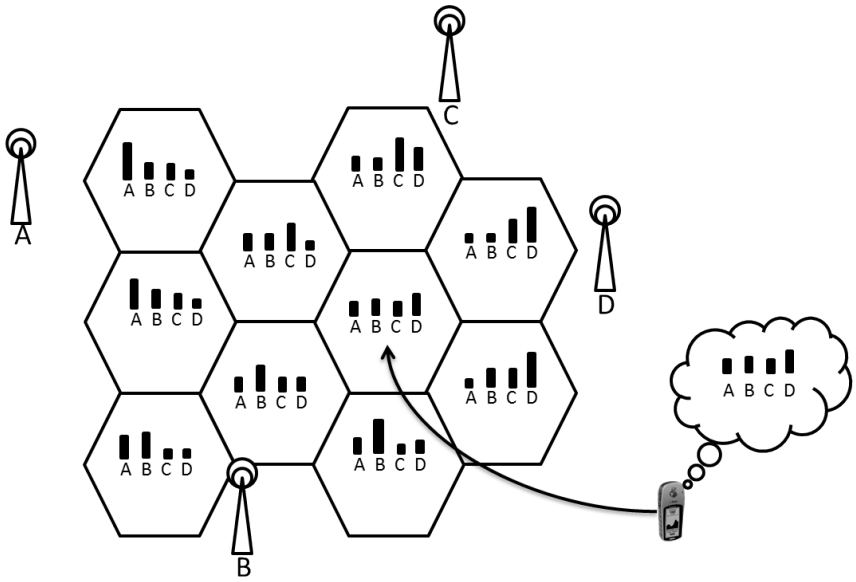


Abbildung 2.4: Musterabgleich anhand der gemessenen Signalstärke von Referenzstationen in einem hexagonalen Gitter.

In der zweiten Phase findet die Positionsbestimmung statt. Hier wird vom Ziel der Positionsbestimmung mit dem gleichen Mechanismus wie in der Kalibrierungsphase ein Muster aufgenommen. Das aufgenommene Muster kann nun mit den bereits bekannten Mustern aus der Kalibrierungsphase verglichen werden. Dazu ist ein Ähnlichkeitsmaß auf den Mustern notwendig, welches einen Grad der Ähnlichkeit ermitteln lässt. Der einfachste Weg zur Position des Ziels ist die Annahme, dass dieses sich an dem Ort befindet, an dem das ähnlichste Muster bezüglich des Maßes aufgenommen wurde (siehe Abbildung 2.4). Weitere Methoden bilden den Durchschnitt der Position mehrerer ähnlicher Muster oder geben eine diskrete Wahrscheinlichkeitsverteilung für den Aufenthalt an Orten mit ähnlichen Mustern an. Neben diesen gebräuchlichen Ansätzen eignet sich prinzipiell jede Technik des maschinellen Lernens, die auf Basis von Trainingsdaten Entscheidungen schlussfolgern kann.

Die Genauigkeit der Positionsbestimmung ist beim Musterabgleich durch Messungenauigkeiten beschränkt, zusätzlich jedoch durch die Dichte der Referenzpositionen und die lokale Ähnlichkeit der Muster. Umgebungseigenschaften, die sich lokal stark ändern, sind zum Musterabgleich nicht geeignet, wenn die Dichte der Referenzpositionen nicht höher ist als die Dichte lokal ähnlicher Gebiete. Variieren die Umgebungseigenschaften jedoch lokal nur sehr wenig, so kann der mögliche Aufenthalt auch nur auf ein großes Gebiet eingeschränkt werden. Die Genauigkeit des Sensors muss in jedem Fall hoch genug sein, um lokale Änderungen erkennen zu können. Das bedeutet, dass der Einfluss der lokalen Änderungen über dem Rauschlevel des Sensors liegen muss. Zudem ist ein sinnvoller Musterabgleich nur dann möglich, wenn Trainingsdaten im Aufenthaltsbereich vorliegen. In manchen Fällen kann es sogar sein, dass an entfernten Orten ähnliche Muster entdeckt werden und so eine

falsche Position geschätzt wird. Zudem ist die Genauigkeit dieses Ansatzes mit tatsächlich empirisch aufgenommenen Daten meist deutlich höher als bei Trainingsdaten, die mithilfe von Modellen berechnet wurden. Ein gutes Beispiel dafür ist WLAN-Fingerprinting, wo die erreichbare Genauigkeit bei empirisch ermittelten Daten deutlich über denen eines Modells liegt [10].

Musterabgleich lässt sich am besten als Endgeräte-zentrierte oder Endgeräte-unterstützte Positionsbestimmung umsetzen, da das Muster lokal am Ort der Positionsbestimmung gemessen werden muss. Ist der Prozess des Musterabgleichs komplex oder ist die Referenzdatenbank mit den Trainingsmustern nicht am Endgerät verfügbar, so kommt die Endgeräte-unterstützte Positionsbestimmung in Frage. Ist dem Endgerät die Referenzdatenbank bekannt und der Abgleich auf dem Endgerät schnell genug berechenbar, so kann auch eine Endgeräte-zentrische Lösung umgesetzt werden.

2.1.3 Nahbereichserkennung

Die Nahbereichserkennung, auch als Proximity oder Presence Detection bekannt, umfasst das Erkennen der Anwesenheit oder Nähe des Ziels im Verhältnis zu einer Referenzstation mit bekannter Position. Dazu werden eindeutig identifizierbare Signale mit begrenzter Reichweite eingesetzt, die den Aufenthaltsort auf das Sendegebiet einschränken (siehe Abbildung 2.5). Ob Signale dabei von dem Ziel der Positionsbestimmung gesendet werden und im Netz der Referenzstationen empfangen und ausgewertet werden oder ob das Ziel selbst die Signale der Referenzstationen empfängt, hängt vom Anwendungsfall ab. Beispiele für Nahbereichserkennung sind Funksysteme mit Funkzellenidentifikatoren, wie WLAN, Bluetooth oder Mobilfunknetze, aber auch RFID oder stationäre Kameras. In letzterem Fall muss zusätzlich eine Identifikation des Ziels stattfinden, welche beispielsweise durch Musterabgleich erreicht werden kann. Ist ein Ziel auf dem Kamerabild identifiziert, so ist damit auch der Aufenthaltsort im Sichtbereich der Kamera ermittelt.

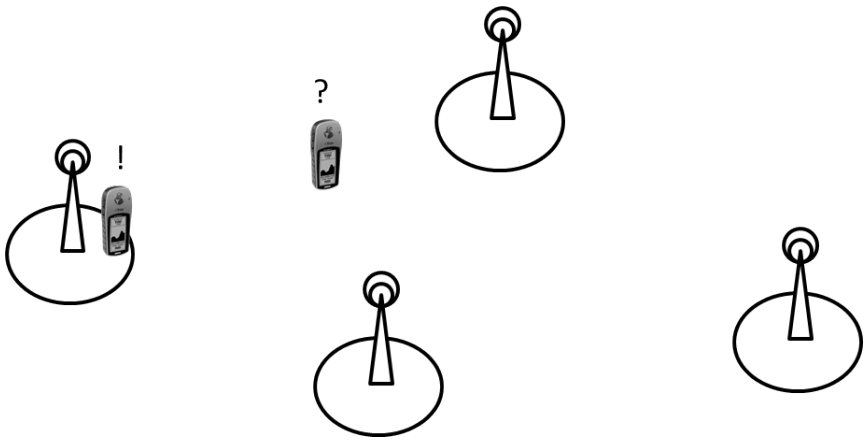


Abbildung 2.5: Nahbereichserkennung mit Sendern begrenzter Reichweite. Außerhalb der Reichweite ist keine Positionsbestimmung möglich.

Die Genauigkeit des Ansatzes hängt stark von der Größe des Gebiets ab, in dem die Signale empfangen werden können. Ist die Reichweite des Signals hoch, so ist das Empfangsgebiet groß. Somit kann bei Empfang des Signal der Aufenthaltsort nur auf das große Gebiet beschränkt werden. Zudem ist zu erwähnen, dass die Nahbereichserkennung nur dort Informationen über den Aufenthaltsort liefern kann, wo auch Signale zur Positionsbestimmung verfügbar sind. Dies kann insbesondere bei Sendern einer geringen Reichweite in einer großen Anzahl an Referenzstationen resultieren, um eine flächendeckende Abdeckung zu erreichen.

Die Nahbereichserkennung lässt sich wiederum Endgeräte-zentrisch, -unterstützt oder Infrastruktur-basiert umsetzen. Im ersten Fall muss das Endgerät die Position der einzelnen Referenzstationen kennen und diese identifizieren können. Dabei kann die Position analog zur Triangulation im Signal enthalten sein. Im zweiten Fall wird die Messung des Signals einer Infrastruktur mitgeteilt und die Antwort dem Endgerät übermittelt. Dies ist vor allem dann sinnvoll, wenn man dem Endgerät die genaue Position der Referenzstationen vorenthalten möchte oder die Datenbasis zu umfangreich zur Auswertung auf dem Endgerät ist. In einem Infrastruktur-basierten Ansatz messen die Referenzstationen die Signale, die vom Endgerät ausgesendet werden, und nehmen als Aufenthaltsort entsprechend den Empfangsbereich der Empfängerstation an.

2.1.4 Koppelnavigation

Die Koppelnavigation, auch als Dead Reckoning bekannt, ist eine Methode zur relativen Positionsbestimmung. Zu diesem Zweck wird die Positionsänderung über die Zeit ausgehend von einer initial bekannten Startposition gemessen und somit ein Rückschluss auf die absolute Position zu jedem Zeitpunkt möglich. Als Maß der Positionsänderung wird die zurückgelegte Strecke und die Bewegungsrichtung ermittelt und daraus der neue Aufenthaltsort berechnet (siehe Abbildung 2.6). Die Strecke kann auch durch ihre Ableitungen wie Geschwindigkeit oder Beschleunigung dagestellt sein, die Bewegungsrichtung durch die Winkelgeschwindigkeit oder Winkelbeschleunigung. Koppelnavigationssysteme benötigen keine Infrastruktur zur Positionsbestimmung und lassen sich somit in Gebieten einsetzen, in denen keine zusätzliche Infrastruktur existiert. Zusätzliches Wissen über die Umgebung kann dennoch sehr nützlich sein, indem beispielsweise die Plausibilität der Fortbewegung mithilfe von topologischen und/oder geometrischen Kartendaten überprüft und bewertet wird. Solche Verfahren werden auch als Mapmatching oder Kartenabgleich bezeichnet.

Die Genauigkeit hängt bei der Koppelnavigation von der Genauigkeit der Sensoren ab, mit denen die Positionsänderung ermittelt wird. Zudem spielt zusätzlich die vergangene Zeit seit der letzten bekannten Startposition eine große Rolle. Das liegt daran, dass sich Fehler in der relativen Positionsbestimmung über die Zeit addieren und so mit fortschreitender Dauer schlechtere Positionsschätzungen ermittelt werden. Gerade in diesem Fall kann eine Abgleich der durch Koppelnavigation geschätzten Position mit Kartenmaterial helfen, Fehler zu erkennen und möglicherweise sogar zu korrigieren. Gerade in Innenbereichen ist die Bewegung des Ziels auf Freiflächen beschränkt. Bewegt sich die geschätzte Position des Ziels durch eine Wand, so ist die Positionsschätzung offensichtlich falsch und

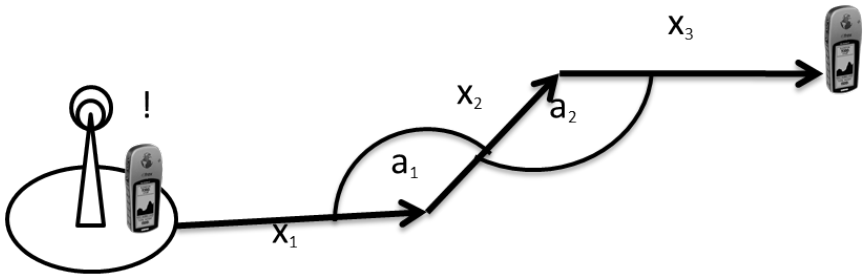


Abbildung 2.6: Koppelnavigation durch Messung der zurückgelegten Strecke und der jeweiligen Richtungswechsel ausgehend von einer bekannten Position.

muss korrigiert werden.

Koppelnavigation lässt sich am besten als Endgeräte-zentrierte Positionsbestimmung umsetzen, da im Zweifelsfall neben der fehlenden Infrastruktur zur Positionierung auch keine Kommunikationsmöglichkeit vorhanden ist. In wenigen Fällen kann es jedoch auch einen Sinn ergeben, die gemessene Geschwindigkeit und Bewegungsrichtung an eine Infrastruktur zu übertragen, um komplexe Berechnungen durchführen zu können. Ein solcher Anwendungsfall ist der im Abschnitt 2.2.2 eingeführte Partikelfilter mit einer hohen Zahl an Partikeln. Die Berechnungen auf der Partikelwolke erfordern eine hohe Leistungsfähigkeit und werden so besser auf Serverkomponenten durchgeführt.

2.2 Techniken zur Sensorfusion

Neben den eben vorgestellten Grundtechniken zur Positionsbestimmung existieren viele Ansätzen, die verschiedene Technologien innerhalb einer solchen Technik nutzen oder mehrere Methoden für eine genauere und/oder flexiblere Positionsbestimmung kombinieren. Diese Methodik wird auch als Sensorfusion bezeichnet. Die in der Positionsbestimmung gebräuchlichsten Ansätze basieren auf stochastischen Modellen und bedingten Wahrscheinlichkeiten, da sich so heterogene Sensordaten mittels Ansätzen der Wahrscheinlichkeitstheorie, wie Markov-Ketten, Bayes Regel, Kalmanfiltern und Monte-Carlo Methoden, kombinieren lassen. Im Folgenden werden die Grundlagen für die verschiedenen Sensorfusionsalgorithmen vorgestellt.

2.2.1 Markov-Ketten und Bayes Regel

Eine Positionsbestimmung wird auf der Basis von Messwerten verschiedener Sensoren durchgeführt. Dabei können die Sensoren einen physikalischen Effekt nur auf eine gewisse Genauigkeit hin bestimmen, und die Messwerte können zudem durch Fehler, Umgebungseinflüsse und Rauschen verfälscht werden. Dieser Effekt wird dadurch verstärkt, dass mit einem unbekanntem Ort keine Aussage über die erwarteten Einflüsse getroffen werden kann. Somit besteht die Aufgabe der Positionsbestimmung unter anderem darin, aus

verfälschten Messungen die tatsächliche Position eines Ziels zu ermitteln. Dies ist nur auf eine bestimmte, meist unbekannte, Genauigkeit möglich, da diese von den Umgebungseinflüssen und Störungen abhängt. Statt wie in einem deterministischen Verfahren von einer festen Position auszugehen, entspricht eine Wahrscheinlichkeitsverteilung besser dem tatsächlichen Wissen über den Aufenthaltsort der Ziels. Zudem erlaubt die Darstellung der Position als Aufenthaltswahrscheinlichkeit den Einsatz von mathematischen Modellen der Wahrscheinlichkeitstheorie, wie Bayes Regel und Markov-Ketten.

Bayes Regel besagt, dass die Wahrscheinlichkeit einer Beobachtung an einem bestimmten Ort einen Rückschluss auf den Ort der Beobachtung erlaubt. Sei X der Ort einer Beobachtung I, so gilt für die bedingte Wahrscheinlichkeit $p(X|I)$ des Aufenthalts am Ort X bei Beobachtung von I:

$$p(X|I) = \frac{p(I|X)p(X)}{p(I)} \tag{2.1}$$

Für den Anwendungsfall der Ortsbestimmung kann der Aufenthaltsort als der nach Bayes Regel wahrscheinlichste Ort bestimmt werden oder direkt die Wahrscheinlichkeitsverteilung der Orte als Verteilung des Aufenthaltsortes angenommen werden. Dieses Vorgehen findet seine Anwendung vor allem im Musterabgleich, da die bedingte Wahrscheinlichkeit $p(I|X)$ der Messung I an einem bestimmten Ort X bereits in der Kalibrierungsphase ermittelt werden kann. Auf diese Regel bauen auch die gebräuchlichsten Methoden zur probabilistischen Positionsbestimmung auf, die Bayesische Filter genannt werden. In diesem Fall wird $p(X)$ auch die a priori Verteilung genannt, I bezeichnet eine Messung und $p(X|I)$ die a posteriori Verteilung von X. Zwei Arten dieser Filter, der Kalman- und der Partikelfilter werden im Anschluss genauer vorgestellt. Eine gute Einführung in die probabilistische Positionsbestimmung wird in [128] gegeben.

Wird eine kontinuierliche Positionsbestimmung durchgeführt, so kann man den Zustand x_t des Ziels zu jedem Zeitpunkt t als kontinuierlichen stochastischen Prozess modellieren. Der Zustand beinhaltet möglicherweise nicht nur die Position, sondern ebenfalls die Ausrichtung im Raum, die Geschwindigkeit der Fortbewegung und ähnliches. Hängt der weitere Verlauf des stochastischen Prozesses zum Zeitpunkt t nur vom aktuellen Zustand x_t ab, so erfüllt dieser die *Markov-Eigenschaft* und wird auch als Markov-Kette bezeichnet. Obwohl sich der Zustand des Ziels oftmals kontinuierlich ändert, kann eine Aussage über den Zustand nur zu bestimmten diskreten Zeitpunkten getroffen werden. Dies ist beispielsweise der Fall, wenn der Systemzustand aus Beobachtungen ermittelt wird und zwischen zwei Messungen keine Informationen über den Zustand vorliegen. Um den stochastische Prozess angeben zu können, benötigt man zudem ein Modell für die Wahrscheinlichkeit $p(x_k|x_{k-1})$ des Zustandsübergangs zwischen zwei aufeinanderfolgenden diskreten Zuständen x_{k-1} und x_k. Zustände zu diskreten Zeitpunkten erhalten im Folgenden den Index k, um die Diskretisierung zu betonen. Erfüllt der Prozess die Markov-Eigenschaft, so ist der Nachfolgezustand einzig vom aktuellen Zustand abhängig. In diesem Fall charakterisiert die Wahrscheinlichkeit $p(x_k|x_{k-1})$ vollständig die Entwicklung des Zustands über die Zeit. Diese Wahrscheinlichkeit kann somit zum Zeitpunkt k einer Messung z_k als a priori Verteilung hergenommen werden, die durch die bedingte Wahrscheinlichkeit $p(z_k|x_k)$ und die normalisierende Kon-

stante $p(z_k)$ in Bayes Regel zur Berechnung der a posteriori Verteilung genutzt werden kann (siehe Gleichung (2.2)).

$$p(x_k|z_k) = \frac{p(z_k|x_k) \int p(x_k|x_{k-1})p(x_{k-1}|z_{k-1})dx_{k-1}}{p(z_k)} \tag{2.2}$$

Ein solcher Prozess wird auch als rekursiver Bayesischer Filter bezeichnet, da $p(x_k|z_k)$ von $p(x_{k-1}|z_{k-1})$ abhängt.

2.2.2 Bayesische Filter

Nach [4] wird ein Bayesischer Filter durch zwei Funktionen f und h charakterisiert: f beschreibt den Zustandsübergang und h den Zusammenhang zwischen Messungen und dem Zustand. Dementsprechend arbeitet eine solcher Filter mit zwei Modellen: Das erste Modell ist das sogenannte Vorhersagemodell, das zweite Modell wird oftmals als Messmodell oder Korrekturmodell bezeichnet. Diese beiden Modelle werden nun formal und in Anlehnung an [4] vorgestellt, woraus sich letztlich Gleichung (2.2) ergibt.

Das Vorhersagemodell beruht auf der Wahrscheinlichkeit $p(x_k|x_{k-1})$, in einem bestimmten Zustand x_k zu einem diskreten Zeitpunkt k zu landen, wenn der vorherige Zustand zum Zeitpunkt $k-1$ x_{k-1} war. Diese Wahrscheinlichkeit kann von Gleichung (2.3) gefolgert werden. Dabei ist f_{k-1} eine beliebige Funktion, welche den zeitlichen Zustandsübergang ausgehend vom Vorzustand x_{k-1} und einem unabhängigen identisch verteilten Rauschen v_k modelliert.

$$x_k = f_{k-1}(x_{k-1}, v_{k-1}) \tag{2.3}$$

Das Messmodell schätzt den tatsächlichen Systemzustand x_k durch eine Zustandsmessunge z_k über eine Funktion h_k des Zustands x_k und einer unabhängigen identisch verteilten Zufallsvariable n_k, die das Rauschen im Messmodell darstellt (siehe Gleichung (2.4)).

$$z_k = h_k(x_k, n_k) \tag{2.4}$$

Von Gleichung (2.4) kann nun die Wahrscheinlichkeit $p(z_k|x_k)$ gefolgert werden, im Zustand x_k eine Messung z_k zu erhalten. Mithilfe der Chapman-Kolmogorov Gleichung (2.5), wobei $z_{1:k} = \{z_1, ..., z_k\}$ alle Messungen bis zum Zeitpunkt k darstellt, wird der Einfluss der vorhergehenden Messungen und des Vorhersagemodells auf den aktuellen Systemzustand x_k abgebildet.

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{2.5}$$

Nimmt man nun die Markov-Eigenschaft $p(x_k|x_{k-1}, z_{1:k-1}) = p(x_k|x_{k-1})$ an, nämlich dass ein Nachfolgezustand x_k nur vom Vorzustand x_{k-1} abhängt und dieser bereits alle Informationen der Vergangenheit $z_{1:k-1}$ enthält, so kann man mit Bayes Regel aus Gleichung (2.6) die a posteriori Verteilung für den Systemzustand berechnen, welche einzig von der

a priori Verteilung $p(x_k|z_{1:k-1})$ und der Messung z_k abhängt.

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{2.6}$$

Dabei bildet $p(z_k|z_{1:k-1})$ aus Gleichung (2.7) eine normalisierende Konstante.

$$p(z_k|z_{1:k-1}) = \int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k \tag{2.7}$$

Setzt man Gleichung (2.5) in Gleichung (2.6) ein und legt die Unabhängigkeit der einzelnen Messungen sowie die Markov-Eigenschaft zugrunde, so ergibt sich Gleichung (2.2).

Doch wie lassen sich Bayesische Filter für die Fusion heterogener Sensordaten zur Positionsbestimmung einsetzen? Eine Möglichkeit ist durch den Einsatz verschiedener Messmodelle gegeben. Zu jedem Zeitpunkt einer Messung durch einen beliebigen Sensor wird das zugehörige Messmodel auf die a priori Verteilung angewendet. Diese wird wiederum anhand des Zustandsübergangsmodells und des Vorzustands, sprich der a posteriori Verteilung nach der letzten Messung, ermittelt. Zudem ist es ebenso denkbar, dass gerade Koppelnavigationssysteme als Berechnungshilfe des Zustandsübergangssystems genommen werden. Die gemessene relative Bewegungsänderung eignet sich hervorragend, um den erwarteten Zustand für jeden Zeitpunkt aus einer vorhergehenden Zustandsschätzung zu ermitteln. Somit lässt sich jede Form der absoluten Positionsbestimmung in Form eines Messmodells darstellen und die relative Positionsbestimmung als Zustandsübergangsmodell. In beiden Fällen ist es jedoch notwendig, anstelle einer festen Position eine Aufenthaltswahrscheinlichkeit oder Übergangswahrscheinlichkeit zu bestimmen. Je besser die entsprechenden Modelle die Realität abbilden, desto realistischer ist auch die Verteilungsfunktion des Zustands.

Im Folgenden werden mit dem Kalmanfilter und dem Partikelfilter zwei unterschiedliche Umsetzungen eines Bayesischen Filters genauer vorgestellt.

Kalmanfilter

Der Kalmanfilter ermöglicht die Kombination zweier normalverteilter linearer Modelle. Dabei entsteht erneut ein normalverteiltes lineares Modell, dessen Varianz minimal ist und daher das optimale Ergebnis aus der Kombination der beiden Modelle darstellt (siehe Abbildung 2.7).

Die Optimalität der a posteriori Verteilung ist an die folgenden Bedingungen geknüpft [57]:

- $p(x_{k-1}|z_{k-1})$ ist normalverteilt,
- die Zufallsvariablen des Rauschens, n_k und v_k, sind normalverteilt und
- die beiden Funktionen f und h sind linear.

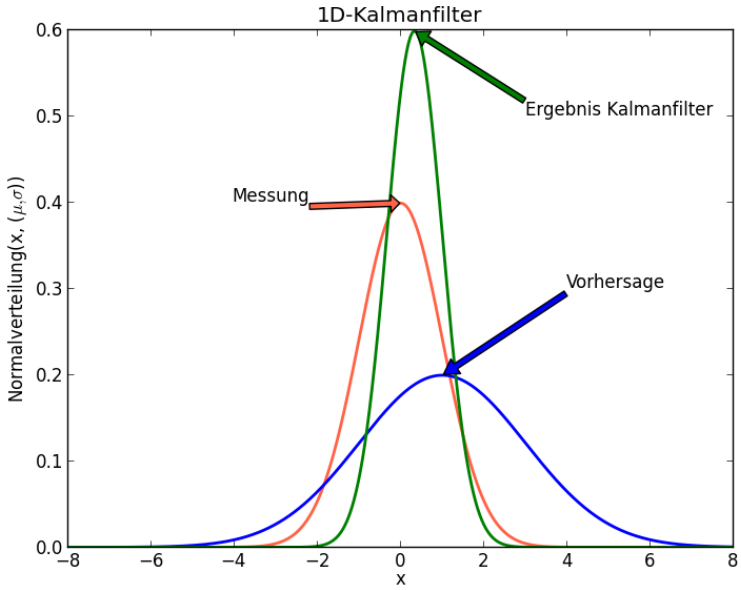


Abbildung 2.7: Ergebnis der Kalmanfilterung zweier eindimensionaler Normalverteilungen.

Unter diesen Voraussetzungen ist auch $p(x_k|z_k)$ normalverteilt. Durch die Linearität der Funktionen können diese durch eine Matrixmultiplikation mit Matrizen F_k bzw. H_k ausgedrückt werden.

Für die resultierende Normalverteilung (2.9) lassen sich die optimalen Argumente für die Parameter Mittelwert m und Kovarianz P nach [57] berechnen.

$$p(x_k|z_{k-1}) = \mathcal{N}(x_k; m_{k|k-1}; P_{k|k-1}) \tag{2.8}$$

$$p(x_k|z_k) = \mathcal{N}(x_k; m_{k|k}; P_{k|k}) \tag{2.9}$$

Die Argumente für $m_{k|k-1}$, $m_{k|k}$, $P_{k|k-1}$ und $P_{k|k}$ sind in diesem Fall durch die Lösung der folgenden Gleichungen gegeben (vgl. [4]):

$$m_{k|k-1} = F_k m_{k-1|k-1} \tag{2.10}$$

$$P_{k|k-1} = Q_{k-1} + F_k P_{k-1|k-1} F_k^T \tag{2.11}$$

$$m_{k|k} = m_{k|k-1} + K_k(z_k - H_k m_{k|k-1}) \tag{2.12}$$

$$P_{k|k} = P_{k|k-1} - K_k H_k P_{k|k-1} \tag{2.13}$$

Dabei bezeichnet Q_k die Kovarianzmatrix von n_k und K_k den optimale Kalman Gain, der wie folgt mithilfe der Matrix H des Messmodells, der Kovarianzmatrix $P_{k|k-1}$ und der Kovarianzmatrix R_k von v_k berechnet wird:

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{2.14}$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \tag{2.15}$$

Da die Voraussetzungen der Linearität von f und h zur Optimalität des Kalman Filter sehr restriktiv sind, wurden Algorithmen wie der Extended Kalman Filter entwickelt. Dabei werden nicht-lineare Funktionen durch Einsatz von Taylor-Reihen linearisiert und die Matrizen F und H anstelle des linearen Systems durch Jacobische Matrizen ersetzt. Dadurch verliert der Kalmanfilter jedoch die Eigenschaft der Optimalität. In solchen Fällen und im Falle von nicht-normalverteilten Wahrscheinlichkeiten kann es von Vorteil sein, die Wahrscheinlichkeitsverteilung zu diskretisieren und mit den diskreten Hypothese zu rechnen. Dies wird im folgenden Abschnitt anhand von Partikelfiltern vorgestellt.

Partikelfilter

Auch der Partikelfilter ist ein rekursiver Bayesischer Filter, der allerdings mit beliebigen Wahrscheinlichkeitsverteilungen und nicht-linearen Funktionen zurechtkommt. Bei Partikelfiltern wird die Dichtefunktion durch eine Menge N von gewichteten Hypothesen, den sogenannten Partikeln, diskretisiert und angenähert. Um eine Wahrscheinlichkeitsverteilung entsprechend gut anzunähern, ist eine genügend große Zahl an Partikeln notwendig. Diese Partikel $\{x_k^i\}$ und das zugehörige Gewicht $\{w_k^i\}$ werden sequentiell entsprechend der Gleichungen (2.5) und (2.6) aktualisiert. Das Gewicht w_k^i eines Partikels i steht dabei für die Wahrscheinlichkeit $p(x_k = x_k^i | z_{1:k})$ des Systemzustands, zum Zeitpunkt k genau x_k^i zu sein. Da die Gewichte ein diskretes Wahrscheinlichkeitsmaß darstellen, ist die Summe der Gewichte aller Partikel genau 1. Der Zustand x_k eines Systems lässt sich dementsprechend als gewichtete Summe der Zustände aller Partikel beschreiben:

$$x_k = \sum_i^N x_k^i \cdot w_k^i \tag{2.16}$$

Initialisiert werden die Partikel entsprechend dem Vorwissen über den Zustand eines Systems, indem sie zufällig aus einer passenden Verteilung gezogen werden. Um aus einem Zustand x_{k-1} bzw. der Partikelwolke $\{x_{k-1}^i\}$ zum Zeitpunkt $k-1$ den Nachfolgezustand x_k bzw. die nachfolgende Partikelwolke $\{x_k^i\}$ zum Zeitpunkt k zu berechnen, gibt es mehrere Möglichkeiten. Die gebräuchlichsten sind dabei Sequential Importance Sampling (SIS) und Sequential Importance Resampling (SIR).

SIS-Partikelfilter berechnen im Allgemeinen den Zustandsübergang je Partikel, indem die Partikel einen zufälligen neuen Zustand x_k^i entsprechend der Verteilung $p(x_k^i | x_{k-1}^i)$

ziehen. Die a priori Verteilung wird durch die nach Gleichung (2.17) modifizierten Gewichte $\bar{w}_k^i$ gegeben.

$$\bar{w}_k^i = p(x_k^i | x_{k-1}^i) \cdot w_{k-1}^i \tag{2.17}$$

Die a posteriori Verteilung wird anschließend gemäß Messmodell $p(x_k^i|z_k)$ mithilfe von Gleichung (2.18) und anschließender Normalisierung (2.19) bestimmt:

$$\hat{w}_k^i = p(x_k^i | z_k) \cdot \bar{w}_k^i \tag{2.18}$$

$$w_k^i = \frac{\hat{w}_k^i}{\sum_i^N x_k^i \cdot w_k^i} \tag{2.19}$$

Hierbei kann jedoch das Problem der Degenerierung auftreten, bei dem ein oder wenige Partikel den Großteil des Gewichts auf sich vereinen und somit die Eigenschaften des Partikelfilters zur Approximation einer Verteilung durch eine große Partikelzahl verloren gehen. Dazu arbeiten SIS-Partikelfilter häufig mit Grenzwerten, die Partikel mit niedrigen Gewichten entfernen, um Berechnungsarbeit für unwahrscheinliche Hypothesen zu vermeiden, und gleichzeitig Partikel mit hohen Gewichten in mehrere Partikel aufteilen, um eine möglichst hohe Auflösung in Bezug auf wahrscheinliche Zustände zu erreichen. Letzter Schritt geht bereits in die Richtung des SIR-Partikelfilters, der im Folgenden vorgestellt wird. [4]

Beim SIR-Partikelfilter wird die Menge $\{\bar{x}_k^i\}$ von Partikeln entsprechend der Verteilung der Partikel $\{x_{k-1}^i\}$ im Vorzustand nach Anwendung der Übergangswahrscheinlichkeit $p(x_k|x_{k-1})$ zufällig generiert. Damit ist eine Approximation der a priori Verteilung gegeben. Die zugehörigen Gewichte werden entsprechend der Messwahrscheinlichkeit $p(z_k|\bar{x}_k^i)$ für jedes Partikel einzeln berechnet. Im folgenden Resampling-Schritt wird durch N-maliges Ziehen mit Zurücklegen aus der Menge der Partikel entsprechend der jeweiligen Wahrscheinlichkeit eine neue Menge $\{x_k^i\}$ von Partikeln gezogen, die nun die a posteriori Verteilung nach der neuen Information der Messung approximieren. [128]

2.3 Gebräuchliche Technologien zur Positionsbestimmung in Gebäuden

Neben den allgemeinen Methoden zur Positionsbestimmung und der Sensorfusion werden nun existierende Systeme zur Positionsbestimmung aus der Forschung anhand der zugrundeliegenden Technologie klassifiziert und vorgestellt. Zudem wird die grundsätzliche Eignung der Technologie für die Bereitstellung von ortsbezogenen Diensten für Fußgänger in Gebäuden bewertet.

2.3.1 Funk-basierte Systeme

Die größte Verbreitung haben Systeme, die die Position mithilfe von Funkwellen bestimmen. Dementsprechend groß ist auch die Vielfältigkeit der Ansätze, die Nahbereichserkennung, den Musterabgleich oder alle Arten der Triangulation anwenden. Je nach Übertragungsfrequenz unterscheiden sich zudem die physikalischen Eigenschaften der Signale, wie Reichweite und Dämpfung.

Das wohl bekannteste Positionsbestimmungssystem auf Basis von Funkwellen ist GPS [101]. Hierbei werden Wellen mit Frequenzen im Bereich von 1200 MHz bis 1600 MHz eingesetzt, um aus der bekannten Position der Satelliten und der Signallaufzeit zum Empfänger dessen Position zu bestimmen. Dabei kommt der klassische Laterationsansatz Time-of-Arrival zum Einsatz. Die Genauigkeit unter freiem Himmel beträgt bei gebräuchlichen zivilen GPS-Empfängern ca. 10 Meter, während Empfänger mit zwei Frequenzen oder differenziellem GPS Genauigkeiten unter einem Meter erreichen können. Letztere sind jedoch noch nicht in Smartphones verfügbar. Der größte Nachteil von GPS ist die starke Abschwächung der Signale durch Bausubstanzen, wodurch ein Empfang von genügend GPS-Satelliten in ausreichender Qualität innerhalb von Gebäuden oftmals nicht möglich ist. Zudem ist selbst bei erfolgreicher Positionsbestimmung die Genauigkeit in Gebäuden durch Reflexion und Mehrwegeausbreitung deutlich geringer als in Außenbereichen. Ein handelsüblicher GPS-Empfängern erreicht eine 3D-Genauigkeit von knapp 30 Metern [28] und eignet sich damit nur für wenige ortsbezogene Dienste.

Um die Probleme von Satelliten-gestützter Positionsbestimmung zu beheben und auch in schwierigen Umgebungen noch eine Position bestimmen zu können, werden zuweilen zusätzliche Sender auf dem Boden eingesetzt. Solche Systeme werden Pseudoliten genannt und arbeiten entweder auf der gleichen Frequenz mit denselben Nachrichten wie die Satelliten, was eine Kompatibilität mit herkömmlichen Empfängern ermöglicht, oder erweitern die Protokolle und/oder Funkfrequenzen. Damit erfordern sie jedoch eine spezielle Hardware. Einsatzgebiete sind beispielsweise Fabrikhallen und Tagebau, wobei bei genügend Pseudoliten eine Genauigkeit im Bereich weniger Zentimeter erreicht werden kann [107]. Aufgrund der kostenintensiven Infrastruktur und erforderlicher Sichtlinie zur genauen Ortung kommen Pseudoliten für den flächendeckenden Einsatz für ortsbezogene Dienste nicht in Frage.

Abgesehen von GPS werden oftmals bestehende Kommunikationsnetze, wie Mobilfunknetze, WLAN, Bluetooth und RFID, aber auch Radio- und Fernsehsignale zur Positionsbestimmung genutzt.

Ein weiteres bekanntes Beispiel, das bereits auf vielen Smartphones zur groben Positionsbestimmung eingesetzt wird, ist die Ortung anhand des Mobilfunknetzes. Dabei lässt sich nicht nur bestimmen, in welcher Funkzelle ein Nutzer eingewählt ist und somit im Bereich welcher Zelle er sich befindet, sondern es werden oftmals auch die Signale der Nachbarzellen und deren Stärken genutzt, um den Bereich weiter einzuschränken. Es existieren Forschungsarbeiten, die auch in Innenbereichen eine Genauigkeit im Bereich weniger Meter erreichen, indem Techniken des Musterabgleichs genutzt werden [99]. Um diese Güte zu erreichen, müssen jedoch die Signale von fünf oder mehr Stationen empfangen werden,

was die Genauigkeit des Ansatzes in dünn besiedelten Gebieten mit wenigen Mobilfunkstationen stark verringert. In Bereichen, in denen viele Basisstationen verfügbar sind, bietet sich diese Technologie zur Positionsbestimmung auf dem Smartphone an.

In der Forschung genießt die Positionsbestimmung mithilfe von WLAN eine große Beliebtheit. In vielen Gebäuden ist WLAN flächendeckend verfügbar, und handelsübliche Smartphones sind damit ausgestattet. Zur Positionsbestimmung mit WLAN wird aufgrund der Ungenauigkeit der Uhren und Zeitstempel meist die empfangene Signalstärke verwendet. Dabei kommen neben Lateration und Nahbereichserkennung vor allem Techniken des Musterabgleichs zum Einsatz. RADAR [10] ist einer der ersten Ansätze zum WLAN-Fingerprinting und funktioniert sowohl mit gemessenen Trainingsdaten, als auch mit Daten, die aus einem Modell berechnet wurden. Die Genauigkeit liegt dabei bei dem empirischen Ansatz zwischen zwei und fünf Metern, während der Modell-basierte Ansatz schlechter abschneidet. Das 50%-Quantil unter Verwendung des Modells liegt bei $4,3$ Metern im Vergleich zu $2,9$ Metern beim empirischen Ansatz. Durch Einbeziehung von Aufenthaltswahrscheinlichkeiten und Bayes Regel konnte die Genauigkeit im Horus System weiter verbessert werden [149]. Experimente bestätigen, dass dadurch Genauigkeiten zwischen ein und zwei Metern möglich werden. Andere Systeme nutzen neurale Netze mit ähnlich hohen Genauigkeiten unter Verwendung von weniger Trainingsdaten [12, 114]. Damit bietet sich WLAN als Technologie zur Positionsbestimmung mit dem Smartphone an, wenn eine entsprechende Infrastruktur und Trainingsdaten vorliegen.

Bluetooth wird im Vergleich zu WLAN selten eingesetzt, da hier von der Architektur keine fest verbaute Infrastruktur mit bekannter Position vorgesehen und auch in kaum einem Gebäude verfügbar ist. Aufgrund der ungenauen Uhren wird mit handelsüblicher Bluetooth-Hardware auf Signalstärke-basierte Verfahren gesetzt. Hallberg et al. erreichen durch Triangulation in [45] eine durchschnittliche Genauigkeit von unter zwei Metern, allerdings kann es über 10 Sekunden dauern, bis die Position ermittelt werden kann. Solange keine feste Bluetooth-Infrastruktur in einem Großteil der Gebäude verfügbar ist, eignet sich die Technologie nur als Unterstützung anderer Verfahren zur Positionsbestimmung, wie in UbiSpot [118].

Radio Frequency Identification (RFID) erlaubt die Übertragung von Informationen über kurze Entfernungen von wenigen Metern hinweg. Damit eignet sich das Verfahren vor allem zur Nahbereichsentdeckung, wodurch aufgrund der geringen Reichweite bereits hohe Genauigkeiten erzielt werden. Auf der anderen Seite ist eine große Zahl an Tags notwendig, um eine flächendeckende Positionsbestimmung zu erlauben. Bei RFID werden entweder passive Tags durch den aktiven Leser unter Strom gesetzt, wodurch Signale wie die ID des Tags über Entfernungen bis zu 2 Metern übertragen werden, oder aktive Tags eingesetzt, die deutlich größere Entfernungen bewältigen. Damit unterstützen aktive RFID-Tags auch Signalstärke-basierte Ansätze. Aktives RFID wird beispielsweise in SpotON [51] und LANDMARC [92] zur Positionsbestimmung genutzt, wobei der maximale Fehler in der Positionsbestimmung unter zwei Metern liegt. In heutigen Smartphones ist RFID jedoch nicht direkt verfügbar, und die Erweiterung Near Field Communication besitzt eine zu geringe Reichweite, um ohne Nutzeraktion zur Positionsbestimmung eingesetzt werden zu können.

Wenige Forschungsarbeiten haben sich damit beschäftigt, die Position auch mithilfe von Radio- oder Fernsehsignalen zu ermitteln. Während Systeme zur Ortung mit Radiosignalen eine Genauigkeit von 20 bis 30 Metern erreichen können [30], liegen Systeme auf Grundlage von TV Signalen im Bereich von ca. 40 Metern [119]. Beide Technologien ähneln somit der Indoor-Genauigkeit von GPS und sind nur in wenigen ortsbezogenen Anwendungen für den Einsatz in Gebäuden geeignet.

Ein guter Überblick über weitere Funk-basierte Systeme zur Lokalisierung findet sich in [80].

2.3.2 Kamera-basierte Systeme

Kameras sind eine weitere Technologie, die sich großer Verbreitung im Bereich der Ortsbestimmung in Gebäuden erfreut. Die optische Positionsbestimmung entspricht dabei am ehesten der natürlichen Orientierung von Menschen. Gerade in komplexen Gebäuden, wie Flughäfen und Museen, werden mit Schildern und Wegweisern viele künstliche Landmarken geschaffen, die zur Bestimmung des Aufenthaltsorts oder des Wegs zum Ziel herangezogen werden. Grundsätzlich lassen sich Kamera-basierte Systeme in zwei Klassen einteilen: die erste Klasse besteht aus fest montierten Kameras, die in der Umgebung angebracht und zur Identifizierung und Ortung von Personen eingesetzt werden, und gehört somit zur Infrastruktur-basierten Ortung. Die zweite Klasse besteht aus beweglichen bzw. tragbaren Kameras, die ihre eigene Position ähnlich einem Menschen anhand der visuellen Charakteristiken des Ortes bestimmen [86] und somit zu den Endgeräte-basierten oder Endgeräte-unterstützten Ansätzen zählen. Im Folgenden werden beide Arten von Systemen genauer vorgestellt.

Stationäre Kameras sind in manchen sicherheitskritischen Gebäuden bereits vorhanden und werden für die Überwachung genutzt. Da die Position der Kamera genau bekannt ist, können Personen, die sich durch den Sichtbereich bewegen, per Nahbereichserkennung geortet werden. Voraussetzung hierfür ist die erfolgreiche Identifizierung der Person, um eine Verknüpfung zwischen der Person und dem Ort zu schaffen. In [130] verbinden Van dem Berghe et al. eine solche Positionsbestimmung mit statischen Kameras mit WLAN-Fingerprinting, um die Genauigkeit der Lokalisierung im Bereich des Sichtbereichs der Kameras deutlich zu erhöhen. Dabei wird allerdings keine Identifizierung eingesetzt, wodurch das Kamerasystem alleine nicht zur Ortung geeignet ist. Preis et al. nutzen in [104] die Tiefenkamera der Kinect von Microsoft, um Personen anhand ihrer Gangart zu identifizieren. Dabei wird bei einer Gruppe von neun Versuchspersonen eine Erfolgsquote von bis zu 91 % korrekter Identifizierung mit einem naiven Bayesischen Ansatz erreicht, was den Einsatz der Technologie in Szenarien mit wenigen Zielen, wie Ambient Assisted Living, möglich macht. Aufgrund des eingebauten Tiefensensors ist hier sogar eine Zentimetergenaue Ortung möglich, wenn die Position und Ausrichtung der Kamera bekannt sind. Es gibt jedoch zwei große Kritikpunkte beim Einsatz stationärer Kameras. Der erste betrifft den Schutz der Privatsphäre, der bei einer flächendeckenden Überwachung durch Kameras kombiniert mit der Identifikation von Personen nicht mehr gewährleistet wäre. Der zweite Kritikpunkt liegt in den Kosten, die für eine flächendeckende Ausstattung und Verkabe-

lung von Gebäuden mit stationären Kameras entstehen. Dementsprechend eignet sich der zweite Ansatz der Kamera-Selbstpositionierung besser für den Einsatz in ortsbezogenen Diensten.

Wie bereits angedeutet, verlässt sich die Positionsbestimmung mit mobilen Kameras meist auf den Musterabgleich. Dabei gibt es wiederum mehrere unterschiedliche Ansätze, die aus unterschiedlichen Forschungsbereichen stammen. Aus dem Bereich der Augmented Reality hat sich eine Methode zur Positionsbestimmung mithilfe von künstlichen Markern entwickelt. Wurden ursprünglich an der Position der Marker virtuelle Objekte in das Kamerabild eingebettet, so wurde der Mechanismus schon bald umgekehrt und die eigene Position im künstlichen Marker kodiert. Ist der Marker auf dem Kamerabild zu entdecken, so befindet sich der Nutzer, der die Kamera hält, in der Nähe des Markers. Während die Erkennung des Markers auf visuellem Musterabgleich basiert, liefert die Nahbereichserkennung die grobe Position. Aus der Größe und Verzerrung des Markers im Kamerabild kann man bei bekannter Auflösung und eingestelltem Zoom aus Referenzbildern zudem die Entfernung und den Blickwinkel zum Marker berechnen, um eine genauere Positionsschätzung im Bereich weniger Zentimeter zu erhalten [90]. Die Positionsbestimmung mithilfe von künstlichen Markern hat jedoch auch Nachteile. Die Marker müssen flächendeckend installiert werden, was zwar günstiger ist, als Referenzstationen in Radio-basierten Systemen auszubringen, aber dennoch einiges an Aufwand bedeutet. Zudem müssen die Marker eindeutig sein und im Kamerabild korrekt erkannt werden, was insbesondere bei schlechter Belichtung eine Herausforderung darstellen kann.

Aus der Objekterkennung stammt eine zweite Methode, die sich zur Positionsbestimmung mit Kameras auf natürliche, aber dennoch eindeutige, wiedererkennbare und unterscheidbare Features in der Umgebung verlässt. Diese Features können beispielsweise mithilfe bekannter Algorithmen wie SIFT [82] oder SURF [13] aus Bildern extrahiert werden und, wie in [138] gezeigt zur Positionsbestimmung innerhalb von Gebäuden genutzt werden. Dieses Vorgehen wird hier nur kurz angeschnitten und in Abschnitt 4.3.1 anhand eines selbst entwickelten Systems genauer vorgestellt. In diesem Fall wird analog zum WLAN-Fingerprinting eine Referenzdatenbank in einer initialen Kalibrierungsphase erstellt, in der mehrere Bilder von unterschiedlichen bekannten Positionen aufgenommen werden. Aus diesen Bildern werden wiedererkennbare natürliche Punkte mithilfe des SURF-Algorithmus extrahiert und zu den Bildern gespeichert. Zur Positionsbestimmung werden wiederum die interessanten Punkte mit SURF aus dem aktuellen Kamerabild extrahiert und mit den Referenzdaten verglichen. Das Referenzbild mit der größten Übereinstimmung wird als Grundlage für eine verbesserte Positionsbestimmung genommen, solange die Übereinstimmung über einem definierten Grenzwert liegt. Dazu wird nicht die Referenzposition des Bildes als eigene Position angenommen, sondern die wahrscheinliche Verschiebung zu dieser anhand der zueinander passenden Feature-Punkte berechnet, wodurch Genauigkeiten unter einem Meter erreicht werden. Allerdings ist die Extraktion und der Abgleich der Feature-Punkte sehr rechenintensiv, wodurch die Positionsbestimmung wenige Sekunden benötigen kann.

Ein letzter Ansatz arbeitet mit dem Musterabgleich auf Gebäudeplänen. Dazu werden bestimmte Punkte oder Formen der Raumgeometrie aus dem Kamerabild extrahiert, die

sich mit den Daten eines Gebäudeplans vergleichen lassen. Hile und Boriello zeigen in [52], dass sich Türstöcke durch Kantenerkennung aus Kamerabildern extrahieren lassen. Aus der Anordnung bezüglich des Fluchtpunktes im Bild werden Türstöcke identifiziert. Die Anzahl der Türstöcke und deren Entfernung zueinander liefern ein schematisches Bild des Ganges, in dem sich der Nutzer aufhält. Dieses Bild kann wiederum mit den eingezeichneten Türen in einem Gebäudeplan verglichen werden. So wird die Position des Nutzers bezüglich des Gebäudeplans auf unter einen Meter genau bestimmt. Aufgrund möglicher Symmetrien oder Ähnlichkeiten der Türanordnung in verschiedenen Gängen ist hierbei der Einsatz eines weiteren, zumindest groben Positionsbestimmungsverfahrens notwendig. Der Einsatz des Verfahrens zur Positionsbestimmung in größeren Räumen ist nur begrenzt möglich. Einen Schritt weiter gehen Kohoutek et al. in [70]. Hier wird ein komplettes 3D-Modell des Gebäudes im CityGML Standard [40] mit dem aktuellen Bild einer Kamera mit Entfernungsmesser abgeglichen. Durch die detaillierte Darstellung kann nicht nur der Raum ermittelt werden, in dem sich die Kamera befindet, sondern durch die Erkennung von Türen und Fenstern ist eine relative Positionsbestimmung bezüglich dieser Bauteile mit Genauigkeiten von unter einem Meter möglich.

Für die Positionsbestimmung von Fußgängern für ortsbezogene Dienste eignen sich vor allem die Methoden zur Selbstortung. Allerdings gibt es neben dem Musterabgleich zur absoluten Lokalisierung Ansätze, die aus aufeinanderfolgenden Kamerabildern die relative Bewegung der Kamera ermitteln. Muffert et al. stellen in [89] eine Methode vor, die aufeinanderfolgende omnidirektionale Kamerabilder nutzt, um die relative Fortbewegung der Kamera zu ermitteln. Dadurch kann Koppelnavigation zur Positionsbestimmung genutzt werden. Ein weiterer Ansatz zur Koppelnavigation mithilfe von Kamerabildern wird von Aubec et al. in [7] vorgestellt. Dazu wird die Kamera vor dem Nutzer so auf den Boden gerichtet, dass die Fußspitzen bei jedem Schritt in dem Kamerabild auftauchen. Durch Mustererkennung wird dies registriert und bei jedem Wechsel zwischen linker und rechter Fußspitze ein Schritt erkannt. Zudem fließt die Position der Fußspitze im Kamerabild in die Schrittlängenschätzung mit ein. Für die Schrittrichtungserkennung wird jedoch in [7] auf weitere Sensoren wie Gyroskop und Kompass zurückgegriffen.

Weitere optische Systeme zur Positionsbestimmung werden in [86] vorgestellt.

2.3.3 Systeme mit Beschleunigungs- und Richtungssensoren

Die meisten Systeme zur Positionsbestimmung, die auf Koppelnavigation basieren, nutzen Beschleunigungs- und Richtungssensoren, um anhand der relativen Bewegung auf die Position zu schließen. Dabei kann man verschiedene Arten von Systemen daran unterscheiden, wo am Körper des Nutzers die Sensoren angebracht sind. Fuß- und Hand-getragene Systeme sind hier am verbreitetsten, aber auch an der Hüfte, in Hosen- oder Handtaschen getragene Systeme sind denkbar. Für die Genauigkeit der Positionsbestimmung ist natürlich die Genauigkeit der Sensoren entscheidend. Der größte Vorteil dieser Systeme ist die Unabhängigkeit von bestehender Infrastruktur. Gerade die inertialen Sensoren, Beschleunigungssensor und Gyroskop, messen die eigene Beschleunigung, bzw. Winkelgeschwindigkeit der Drehung, wodurch sie unabhängig von äußeren Störeinflüssen sind, wie elektromagne-

tischen Interferenzen oder Hintergrundstrahlung/Rauschen. Um jedoch auf die Änderung des Ortes zu schließen, müssen die Messwerte ein- oder zweimal integriert werden, wodurch sich die Messfehler in einzelnen Messungen deutlich stärker bemerkbar machen und über die Zeit aufaddieren. Störanfälliger, aber dafür mit einem zeitunabhängigem Fehler sind Magnetfeldsensoren, wie der elektronische Kompass, zur Messung der Bewegungsrichtung oder Luftdrucksensoren, die zur Messung der vertikalen Bewegung eingesetzt werden.

Zu den Fuß-getragenen Systemen zählt beispielsweise die Arbeit von Woodman und Harle [145], in der ein 3-Achsen-Beschleunigungssensor und ein 3-Achsen-Gyroskop an einem Fuß des Nutzers angebracht werden. Die Bewegung des Fußes wird erkannt, einzelne Schritte werden registriert und die Schrittlänge berechnet. Dazu kommt eine Technik namens *Zero Velocity Updates* (ZVU) zum Einsatz. Bei ZVU wird der Moment erkannt, an dem der Fuß auf dem Boden aufsetzt und einen Augenblick lang ohne Beschleunigung verbleibt, bis er zum nächsten Schritt angehoben wird [33]. Dabei kann die Beschleunigung und die Geschwindigkeit auf 0 zurückgesetzt werden und der Fehler, der durch die Integration der Komponenten entsteht, reduziert werden. In [145] kommt zur Verbesserung der Genauigkeit zusätzlich eine Kartenkorrektur zum Einsatz. In [146] wird der Ansatz weiter verbessert und WLAN-Fingerprinting für eine initiale Positionsbestimmung verwendet. Weitere Beispiele für Fuß-getragene Systeme werden in [108, 35, 131] vorgestellt.

Hand-getragene Systeme verwenden in den meisten Fällen die Sensoren, die im Smartphone verbaut sind und verwirklichen so vielfach die Idee der *self-contained* oder selbstenthaltenen Positionsbestimmung ohne Infrastruktur und ohne Kommunikation mit anderen Komponenten. Dieses Vorgehen lässt sich der Endgeräte-basierten Positionsbestimmung zuordnen und bietet das höchste Maß an Privatsphäre. Dabei gibt es bei der Verwendung von in der Hand gehaltenen Smartphones einige Schwierigkeiten zu beachten: Zum einen sind die Sensoren nicht zur Positionsbestimmung bestimmt gewesen und bestehen daher oftmals aus sehr günstiger und ungenauer Hardware. Zum anderen geben die Sensoren die Messwerte bezüglich der Achsen des Gerätes wieder, welche dann erst in das verwendete Referenzsystem transformiert werden, um die relative Bewegung bezüglich eines Gebäudeplans oder einer Karte berechnen zu können. Zuletzt kann die ZVU-Technik nicht eingesetzt werden, da ein Smartphone in der Hand keine Ruhelage aufweist, sondern ständigen Beschleunigungen unterliegt. Meist wird daher eine Schritterkennung aufgrund der gemessenen Beschleunigungswerte durchgeführt und anhand anderer Parameter, wie Schrittfrequenz oder Körpergröße, auf die Schrittlänge geschlossen [42].

Da Koppelnavigationssysteme nur die relative Position ermitteln können, sind diese fast immer auf eine initiale Positionsschätzung angewiesen. Nur in manchen Fällen lässt sich über Hinzunahme von Kartendaten und Kartenfilterung nach einiger Zeit auf die tatsächliche Position schließen. Darauf wird in Abschnitt 4.2.3 genauer eingegangen. Ansonsten wird die Koppelnavigation auch in vielen Lokalisierungssystemen als Vorhersagemodell eingesetzt oder zur relativen Positionsbestimmung zwischen zwei absoluten Positions-Updates genutzt, um eine feingranularere Lokalisierung zu erlauben. Ein Beispiel hierfür lässt sich in [29] finden, in dem inertiale Sensoren die Vorhersage in einem Kalman- oder Partikelfilterbasierten System durchführen, während die absolute Positionsbestimmung mit WLAN erfolgt.

2.3.4 Schall-basierte Systeme

Unter den ersten Ortungssystemen in Gebäuden sind auch Schall-basierte Systeme vertreten, insbesondere ActiveBat [134] und Cricket [121]. Dabei handelt es sich oftmals um Infrastruktur-basierte Systeme, die eine Vielzahl an Ultraschall-Empfängern meist an der Decke platzieren und damit die Signale der mobilen Tags messen und mittels Lateration und/oder Angulation zu einer Ortsschätzung interpretieren. Ultraschall-basierte Systeme haben den Vorteil, dass die Schallwellen nicht vom menschlichen Gehör wahrgenommen werden und somit keine störenden Geräusche bei der Ortung entstehen. Da Ultraschall besonders gut von Hindernissen reflektiert wird, durchdringen die Signale keine Wände, und ein dichtes Netz von Empfängern ist notwendig, um flächendeckende Ortung zu erlauben. Das macht solche Systeme üblicherweise zu teuer, um ganze Gebäude damit auszustatten, obwohl die Technologie Genauigkeiten im Bereich weniger Zentimeter zulässt [134].

Einen ganz anderen Ansatz zur akustischen Lokalisierung liefern Tarzia et al. in [127]. Ganz ohne die Ausbringung von Infrastruktur wird gezeigt, dass mithilfe von Musterabgleich die Wiedererkennung von Räumen anhand des akustischen Hintergundrauschens möglich ist. In einem Testdatensatz von 33 verschiedenen Räumen, wurde mit einem handelsüblichen Smartphone eine Erfolgsquote von 70 % richtiger Wiedererkennung erreicht. Besondere Schwierigkeiten hat das System jedoch, wenn viele Personen sich im Raum gleichzeitig unterhalten oder sich mehrere Geräusche überlagern. Auch in Räumen ohne Geräuschquellen wurde eine geringere Genauigkeit festgestellt. Die Autoren empfehlen eine gemeinsame Nutzung der Technologie mit WLAN-Fingerprinting, da ein ähnlicher Kalibrierungsaufwand auftritt, aber die Kombination beider Ansätze aufgrund von unterschiedlichen Fehlerquellen ein größere Robustheit und höhere Genauigkeit bewirkt.

2.3.5 Sensorfusions-basierte Systeme

Ursprünglich aus der Robotik stammend, nutzen viele Systeme Informationen von mehreren unterschiedlichen Technologien, die mittels Ansätzen der Sensorfusion miteinander kombiniert werden. Dabei wird versucht, die Unzulänglichkeiten der einzelnen Technologien oder Sensoren durch den Einsatz weiterer Komponenten zu kompensieren. Hierbei werden entsprechend der Bayesischen Filter oftmals Komponenten zur direkten absoluten Positionsbestimmung mit Dead Reckoning Systemen kombiniert, die zwischen den einzelnen Messungen relative Positionsänderungen nachverfolgen. Im Folgenden wird ein kurzer Überblick über ausgewählte Systeme und die dabei kombinierten Technologien gegeben.

Evennou und Marx kombinieren in [29] WLAN mit inertialen Sensoren mit einem Kalman- und einem Partikelfilter. Dabei lässt die Auswertung den Schluss zu, dass der Partikelfilter zwar genauere Ergebnisse liefert, allerdings auch eine größere Rechenkomplexität verlangt. In [145] und [146] kombinieren Woodman und Harle ein Fuß-getragenes Dead Reckoning System mit WLAN-Fingerprinting in einem Partikelfilter, um eine schnellere Initialisierung bzw. Konvergenz der Partikel zu erreichen. In [81] schlagen Liu et al. ebenfalls einen Partikelfilter zur Kombination von inertialen Sensoren mit WLAN vor. Im Gegensatz dazu wählen Ruiz et al. in [111] einen Kalmanfilter zur Kombination von Dead

Reckoning mit inertialen Sensoren und Entfernungsmessungen anhand von RFID-Tags.

Es gibt auch Kombinationen von verschiedenen Funktechnologien, wie das Projekt MagicMap [20]. Dabei werden verschiedene Radiofrequenzverfahren in einem Laterationsalgorithmus kombiniert. Zudem gibt es die Möglichkeit, Bewegungsmodelle in einem Partikelfilter mit aus Entfernungsmessungen geschätzten Positionen zu kombinieren. Widyawan et al. stellen in [142] einen Ansatz vor, wie WLAN und ein Sensornetz mithilfe eines Bayesischen Ansatzes zur Positionsbestimmung kombiniert werden können. Dabei wird die Gesamtgenauigkeit des kombinierten Systems höher als die der Einzelsysteme.

Gerade bei den beiden letztgenannten Arbeiten sieht man schön den Unterschied der von Hall und McMullen in [44] vorgestellten zwei möglichen Varianten zur Sensorfusion. Bei der zentralistischen Sensorfusion werden die Messwerte aller Sensoren direkt zur Zustandsschätzung genutzt, also beispielsweise verschiedene Funksignale zur Lateration wie in [20]. Die dezentrale Sensorfusion bildet für jeden Sensor eine Zustandsschätzung und kombiniert diese auf einer höheren Ebene, meist mittels Bayesischen Filtern wie in [142].

Grundsätzlich lassen sich alle Technologien miteinander verbinden, um eine verbesserte Positionsbestimmung durchführen zu können. Beispiele hierfür sind Plattformen zur Positionsbestimmung über beliebige Sensoren, wie beispielsweise der Ansatz des Location Stacks [50], dessen reelle Umsetzung in [38] vorgestellt und diskutiert wird. Ein ähnlicher Ansatz wird von Najib et al. in [91] verfolgt.

2.4 Karten und Umgebungsmodelle

Was ist eine Koordinate ohne eine Referenzsystem, das deren Werte in einen Bezug zur realen Welt setzt? Zu diesem Zweck existieren Karten und Umgebungsmodelle, die einem Nutzer nicht nur die eigene Position in Relation zur Umgebung aufzeigen, sondern ebenfalls Informationen über mögliche Ziele, gangbare Wege oder Orientierungshilfen, wie beispielsweise Landmarken, liefern.

In Außenbereichen kann ein Nutzer aus einer Vielzahl von unterschiedlichen Kartenanbietern wählen und das für die eigenen Zwecke bestgeeignetste Kartenmaterial auswählen. Die Genauigkeit der Karten kann anhand von Satellitenbildern überprüft und verbessert werden, Testpersonen und Qualitätssicherungspersonal sorgen für eine ständige Aktualität. In Innenbereichen liegt eine ganz andere Situation vor. Das Kartenmaterial gehört dem Architekt, Bauträger oder der Gebäudeverwaltung und ist aus Sicherheitsgründen häufig höchstens analog in Form eines ausgehängten Fluchtplans für die Öffentlichkeit verfügbar. Für Nutzer verfügbare digitale und semantisch annotierte Karten und Modelle sind eine Seltenheit. In seiner Keynote auf der IPIN 2012 (3rd International Conference on Indoor Positioning and Indoor Navigation 2012) identifizierte Waleed Kadous, der Leiter der Abteilung für Indoor-Positionsbestimmung und Indoor-Kartenerstellung bei Google, die mangelnde Verfügbarkeit von Gebäudedaten als das vielleicht größte Hindernis von ortsbezogenen Diensten in Gebäuden. Hier werden Standards benötigt, um Gebäudedaten für Diensterbringer und Nutzer gleichermaßen verfügbar zu machen, aber auch Methoden, um bestehende Architekturpläne, Bauzeichnungen oder Fluchtpläne in eine maschinenles-

bare Form zu bringen, die für ortsbezogene Dienste in Gebäuden geeignet ist.

Die speziellen Gegebenheiten innerhalb von Gebäuden führen zudem zu anderen Anforderungen an Indoor-Kartenmaterial als an Straßenkarten. Die Bewegungsfreiheit eines Fußgängers im Gebäude wird nicht durch Straßen, Spuren und Kreuzungen eingeschränkt, so dass sich hier weniger ein Graph möglicher Wege, sondern vielmehr eine Vielzahl an begehbaren Freiflächen gegeben ist. Das beeinflusst nicht nur die Positionsbestimmung, sondern auch die Berechnung kürzester Wege oder die automatische Erzeugung von Navigationsanweisungen. Dementsprechend werden für ortsbezogene Dienste in Gebäuden weiterführende Modelle genutzt, die oftmals an die verfügbaren Daten, die Anwendung und die verfügbaren Positionsbestimmungssysteme angepasst sind.

Becker und Dürr unterscheiden in [14] verschiedene Arten von Umgebungsmodellen im Bereich des allgegenwärtigen oder ubiquitären Computings:

- **Geometrische Modelle** bieten eine Repräsentation des Gebäudes an, bei der Objekte durch geometrische Koordinaten in einem Koordinatensystem gegeben sind. Die Darstellung kann dabei 3D oder 2,5D sein, wobei in letzterem Fall die kontinuierliche Höhenangabe der Objekte durch eine diskrete Stockwerksangabe in der Koordinate ersetzt wird. Diese Darstellung wird beispielsweise für Triangulation und Koppelnavigation benötigt, um Entfernungen oder Winkel berechnen zu können oder beim Mapmatching, um Kollisionen mit Wänden zu ermitteln.

- **Symbolische Modelle** beschreiben das Gebäude durch textuelle und meist menschenverständliche Bezeichner, wie beispielsweise Raumnamen. Diese Darstellung bietet sich gerade bei der Mustererkennung an, wenn sich die Muster bezüglich der symbolischen Bezeichner unterscheiden. So kann die Ansicht oder das empfangene Signalstärkemuster über einem Raum hinweg recht einheitlich sein, aber sich im Nebenraum stark verändern. Symbolische Modelle lassen sich noch weiter unterteilen:

  - Eine Ausprägung stellen die **hierarchischen Modelle** dar, in denen die symbolischen Bezeichner einer strengen Ordnung des Enthaltenseins unterworfen sind. Ein Beispiel hierfür sind Gebäude, die aus Stockwerken bestehen, welche wiederum aus mehreren Räumen zusammengesetzt sind.
  - **Mengenbasierte Modelle** sind eine Verallgemeinerung der hierarchischen Modelle, in denen eine beliebige, möglicherweise auch überlappende Zusammenfassung von symbolischen Bezeichnern in übergeordneten Bezeichnern möglich ist.
  - Zudem existieren **Graph-basierte Modelle**, welche die topologische Verbundenheit zwischen symbolischen Bezeichnern durch Knoten und Kanten eines Graphen modellieren. So sind zwei benachbarte Räume durch eine Kante miteinander verbunden, wenn es eine Tür gibt, durch die ein Nutzer vom einen Raum direkt in den anderen gelangen kann.

Die meisten Modelle verbinden Elemente aus symbolischen und geometrischen Modellen und bieten somit als hybrides Modell eine detaillierte und flexible Beschreibung eines

Gebäudes. Dieses kann nicht nur symbolische und geometrische Koordinaten verschiedener Positionsbestimmungssysteme in Relation zueinander setzen, sondern unterstützt auch eine Vielzahl von ortsbezogenen Diensten, wie Navigations- und Informationsdienste.

2.5 Ortsbezogene Dienste in Gebäuden

Nachdem in den vorhergehenden Abschnitten verschiedene Methoden und Technologien zur Positionsbestimmung und unterschiedliche Ansätze zur Bereitstellung von Umgebungsmodellen vorgestellt wurden, wird in diesem Abschnitt genauer auf ortsbezogene Dienste eingegangen. Dazu werden zunächst ortsbezogene Dienste definiert und eine Klassifikation bestehender Dienste gegeben. Anschließend wird genauer auf den Fall der ortsbezogenen Dienste in Gebäuden eingegangen und zum Schluss werden die Anforderungen an Positionsbestimmung und Umgebungsmodelle aus Sicht der Dienste hergeleitet.

2.5.1 Einführung in ortsbezogene Dienste

So vielfältig wie die ortsbezogenen Dienste selbst, so vielfältig ist auch die Anzahl der unterschiedlichen Definitionen, welche sie charakterisieren. Die GSM Association beschreibt einen ortsbezogenen Dienst als "*Dienst, der die Umgebung des Ziels nutzt, um zusätzliche Funktionalität bereit zu stellen.*" [96, aus dem Englischen]. Axel Küpper definiert in seinem Buch [73] ortsbezogene Dienste folgendermaßen:

> "*Location-based Services (LBSs) are IT services for providing information that has been created, compiled, selected, or filtered taking into consideration the current locations of the users or those of other persons or mobile devices.*" [73]

Im Rahmen dieser Arbeit wird die Definition von Küpper verwendet, um ortsbezogene Dienste zu charakterisieren. Im Gegensatz zu anderen Definitionen umfasst diese die Tatsache, dass auch die Position anderer Personen oder Objekte zur Diensterbringung relevant sein kann, insbesondere natürlich auch die Position mehrerer Ziele. Dadurch werden gerade ortsbezogene Dienste der zweiten Generation erfasst, die sich unter anderem durch Proaktivität, gegenseitiges Referenzieren anderer Personen oder Objekte und mehrerer Ziele auszeichnen [16]. Unter Proaktivität wird insbesondere die Etablierung von sogenannten Push-Diensten verstanden, bei denen im Gegensatz zu reaktiven Pull-Diensten relevante Informationen aktiv auf das Endgerät gesendet werden und keine aktive Datenabfrage seitens des mobilen Gerätes notwendig ist.

Ortsbezogene Dienste lassen sich in verschiedene Anwendungsgebiete unterteilen. Unterschiedliche Klassifikationen teilen die Dienste nach ökonomischen Gesichtspunkten [73] oder Anwendungsgebieten [23, 93]. In [60] werden die verschiedenen Klassifikationen mit den in [125] genannten Anwendungsgebieten kombiniert und neun Klassen von Diensten identifiziert:

- Eine der gebräuchlichsten Klassen sind **Informationsdienste**, die einem Nutzer Informationen über seine Umgebung bieten.

- **Navigationsdienste** erweitern Informationsdienste um eine Komponente zur Wegewahl und Wegeführung zu einem oder mehreren Zielen.
- Anwendungen, die es dem Nutzer erlauben, seine Umgebung für andere sichtbar zu bewerten oder beschreiben, gehören zur Klasse der **Umgebungsannotation**.
- Eine besondere Ausprägung von ortsbezogenen Diensten sind **Spiele**, die auf die Einbeziehung von Ortsinformationen ausgerichtet sind.
- **Management-Dienste** bieten Funktionalität zur ortsabhängigen Verwaltung komplexer Prozesse. Logistik, Flottenmanagement und instrumentierte Umgebungen, die auf die Position des Nutzer reagieren, gehören zu dieser Klasse.
- Zur Klasse des **Trackings** gehören Dienste, welche die Position von Endgeräten oder Personen kontinuierlich überwachen.
- **Ortsbezogenes mobiles Marketing** zielt darauf ab, potentielle Kunden durch Werbung vor Ort zu gewinnen.
- Ortsabhängige Tarife oder Mautsysteme gehören zur Klasse der **Abrechnungsdienste**, bei denen die Kosten in Abhängigkeit vom Aufenthaltsort berechnet werden.
- **Notfalldienste** sind schließlich das ursprüngliche Anwendungsgebiet ortsbezogener Dienste, bei denen die Position in Notfällen zur schnellen Hilfeleistung genutzt wird.

Diese Klassifikation ist weder erschöpfend noch eindeutig, da ein bestimmter Dienst durchaus in mehrere Klassen fallen kann. So kann ein Notfalldienst mit dem Tracking einer Person einhergehen, allerdings existieren durchaus auch Beispiele, welche sich genau einer Klasse zuordnen lassen. Die Klassifikation ist insbesondere in Hinblick auf unterschiedliche Anforderungen an Ortungsverfahren und Umgebungsmodelle sowie passende Systemarchitekturen für skalierbare und effiziente Systeme interessant.

2.5.2 Ortsbezogene Dienste in Gebäuden

Grundsätzlich lassen sich für alle genannten Klassen auch Einsatzszenarien innerhalb von Gebäuden finden. Sei es die Stationen-genaue Abrechnung bei Fahrten mit der U-Bahn, mobiles Marketing in Kaufhäusern, Evakuierung von Gebäuden im Falle eines Feuers oder selbst-organisierende Warenlager, ortsbezogene Dienste bieten auch in Gebäuden zahlreiche Anwendungsmöglichkeiten. In diesem Abschnitt wird daher auf die Besonderheiten von ortsbezogenen Diensten eingegangen, die sich speziell auf den Einsatz in Gebäuden beziehen (Indoor-Location Based Services, I-LBS), und nicht auf die Nutzung herkömmlicher ortsbezogener Dienste aus einem Gebäude heraus. Hierbei gibt es einige Unterschiede zu beachten.

Ein wichtiger Unterschied ist die Bereitstellung von Kartenmaterial in Gebäuden. Wird Kartenmaterial in Außenbereichen flächendeckend von bestimmten Unternehmen, wie Google, Navteq oder OpenStreetMap, angeboten, so ist innerhalb eines Gebäudes die jeweilige

Gebäudeverwaltung für die Baupläne zuständig. Diese Baupläne sind nicht immer in digitaler Form vorhanden und meist mehr als Hilfe beim Bau zu verstehen, denn als semantisches Modell zur Unterstützung von ortsabhängigen Diensten. Selbst falls die Pläne in digitaler Form vorliegen, so müssen aus den geometrischen Informationen für viele Dienste auch Daten über die topologischen Zusammenhänge und symbolischen Bezeichner gewonnen werden. Sind auch diese Hürden genommen, so existiert ein verwendbares Umgebungsmodell für ein bestimmtes Gebäude. Daher ist eine der Herausforderungen für die flächendeckende Bereitstellung von I-LBS, dass der Vorgang der Kartenerstellung soweit wie möglich automatisiert wird.

Ein weiterer grundlegender Unterschied ist die Struktur von Wegen. Außerhalb von Gebäuden werden Wege üblicherweise als Wegpunktgraph in Form von Knoten und Kanten dargestellt, wodurch sich Wegstrecken anhand bekannter Algorithmen, wie Dijkstra oder A*, effizient berechnen lassen. Innerhalb von Gebäuden sind Wege durch Freiflächen gegeben, die zunächst einmal keine Einschränkung der Bewegung auf ein Wegenetz zulassen. Es existieren zwar Algorithmen zur Konstruktion von Wegenetzen aus Flächen, die oftmals gerade in Computerspielen eingesetzt werden [129], allerdings muss bei der Erstellung darauf geachtet werden, dass die resultierenden Wege dem natürlichen Verhalten der Menschen angepasst werden. Aufgrund von unterschiedlichen Fortbewegungsmitteln, wie Rolltreppen und Aufzügen, gibt es zudem Herausforderungen in Bezug auf effiziente Routingalgorithmen und Metriken, da der kürzeste Weg nicht zwingend der schnellste sein muss. Hinzu kommt, dass bestimmten Personengruppen, wie Rollstuhlfahrern, nicht jeder Weg zugänglich ist und in bestimmten Gebäuden, wie Flughäfen, manche Bereiche, wie Sicherheitszonen, nur von Personen mit bestimmten Rechten betreten werden dürfen.

Der letzte wichtige Unterschied ist das Fehlen eines flächendeckend verfügbaren und zudem genügend genauen Systems zur Positionsbestimmung. In Außenbereichen wird diese Funktion von Satellitennavigationssystemen wie GPS übernommen, welche mit wenigen Metern Genauigkeit ausreichende Qualität für die meisten Dienste bereitstellen. Innerhalb von Gebäuden gibt es, wie in den vorherigen Abschnitten bereits vorgestellt, eine Vielzahl von Methoden und Technologien zur Ortung, von denen sich bislang jedoch keine als global verfügbare Lösung etablieren konnte. Das liegt zum einen daran, dass viele Systeme eine fest installierte Infrastruktur benötigen, die zu großen Kosten für Inbetriebnahme und Unterhalt eines solchen Systems führen kann und nicht überall verfügbar ist. Zum Anderen erreichen Systeme ohne Infrastrukturkomponenten auf lange Zeit nicht die notwendigen hohen Genauigkeiten [66], benötigen eine gewisse Warmlaufzeit [145] oder eine zeitaufwändige Kalibrierung [138]. Zudem sind die einsetzbaren Methoden durch die komplexere Umgebung teilweise eingeschränkt. Aufgrund von Brechung und Reflexion lassen sich Angulationsverfahren nur eingeschränkt verwenden. Auch Laterationsverfahren stoßen aufgrund von Mehrwegeausbreitung und Reflexion an ihre Grenzen, wenn Genauigkeiten im Bereich von 1-2 Metern gefordert sind. Ein größerer Fehler kann eine fehlerhafte Lokalisierung auf dem falschen Stockwerk oder in angrenzenden Räumen bedeuten, wodurch die Diensterbringung nur nicht oder nur eingeschränkt erfolgen kann.

Die bestehenden Probleme lassen sich anhand eines Rollenmodells bestimmten Rollen zuordnen. Dazu wird das aus [73] bekannte und in [112] für den Bereich in Gebäuden

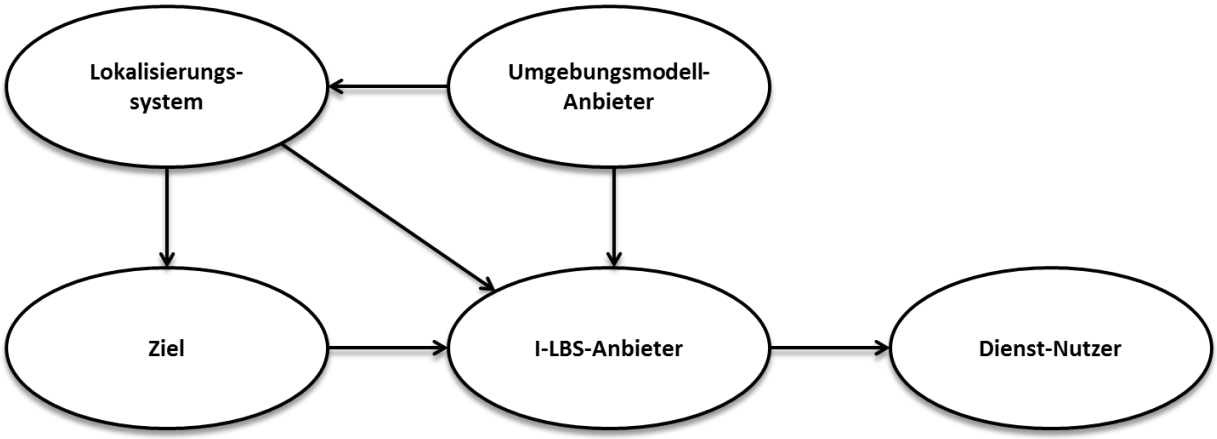


Abbildung 2.8: Einfaches Rollenmodell für I-LBS, adaptiert nach [113] und [112].

adaptierte Rollenmodell für ortsbezogene Dienste der zweiten Generation herangezogen. Abbildung 2.8 zeigt eine Vereinfachung des Modells, welches das Zusammenspiel der verschiedenen Rollen verdeutlicht.

- Der **Dienst-Nutzer** greift als Nutzer eines ortsbezogenen Dienstes auf die vom I-LBS-Anbieter bereitgestellten Funktionalitäten zu.

- Der **I-LBS-Anbieter** bietet dem Nutzer bestimmte Dienste basierend auf der Position der Ziele und der Umgebungsinformationen vom Umgebungsmodell-Anbieter. Dabei kann die Position direkt vom Lokalisierungssystem stammen, beispielsweise im Fall von Infrastruktur-basierter Lokalisierung, oder von den Zielen ermittelt und bereitgestellt werden.

- Der **Umgebungsmodell-Anbieter** bietet dem Lokalisierungssystem und dem I-LBS-Anbieter direkt die benötigten Modelle an oder erlaubt Anfragen an diese.

- Das **Lokalisierungssystem** bestimmt die Position des Ziels und bietet diese im Fall von Endgeräte-zentrischer Ortung dem Ziel an, im Fall von Infrastruktur-basierter oder Endgerät-unterstützter Positionsbestimmung möglicherweise auch direkt dem I-LBS-Anbieter.

- Das **Ziel** stellt seine Positionsdaten dem I-LBS-Anbieter zur Verfügung, wo diese zur Diensterbringung weiter verarbeitet werden.

Natürlich unterliegt das Lokalisierungssystem bestimmten Anforderungen von Seiten des Zieles und des I-LBS-Anbieters, wie auch der Umgebungsmodell-Anbieter Anforderungen seitens des Lokalisierungssystems und des I-LBS-Anbieters erfüllen muss. In den folgenden Abschnitten werden diese Anforderungen ermittelt und genauer vorgestellt.

2.5.3 Anforderungen an Umgebungsmodelle

Die Anforderungen an Gebäudemodelle aus Sicht der ortsbezogenen Anwendungen sind vielschichtig. Sie kommen sowohl von den eingesetzten Ortungssystemen als auch der Art des angebotenen Dienstes. Folgende Schlüsselanforderungen an ein Gebäudemodell wurden in [60] und [61] aus einer Vielzahl von ortsbezogenen Anwendungen innerhalb von Gebäuden (z.B. [5, 97, 47, 113]) identifiziert. Dabei orientiert sich die Aufzählung an den in [14] genannten Anforderungen und gruppiert diese nach den verschiedenen Anfragen, die an ein Umgebungsmodell gestellt werden:

- Es muss möglich sein, eine von einem Ortungssystem gemessene Position auf das Gebäudemodell zu mappen. Dabei kann die Position als symbolischer Bezeichner wie dem Raumnamen oder als geometrische Koordinate gegeben sein. Da viele Ortungssysteme mit lokalen Koordinatensystemen arbeiten, muss eine Koordinatentransformation zwischen Modell-spezifischen und anderen Referenzsystemen unterstützt werden, insbesondere auch eine Zuordnung zwischen symbolischen und geometrischen Koordinaten. Diese Anforderung wird mit **Transformation** bezeichnet.

- Es muss möglich sein, Anfragen zur Wegewahl und Weglänge zu beantworten. Dazu ist eine Graphstruktur notwendig, die zusätzlich in die Geometrie des Gebäudes eingebunden werden sollte, um geometrische Informationen mit topologischen zu verknüpfen. Nachdem jedoch die geforderte Genauigkeit und Auflösung des Graphen je nach Anwendung variieren kann, sollen Graphen austauschbar sein und dementsprechend Änderungen am Graphen weder die Struktur noch die Topologie des Gebäudes verändern. Diese Anforderung wird mit **Routing** bezeichnet.

- Anfragen nach den nächsten Nachbarn müssen unterstützt werden. Der Begriff Nachbar kann sich hierbei auf Objekte, Ort oder Personen von Interesse beziehen, wobei Einschränkungen nach bestimmten Typen erlaubt sein müssen. Die Nähe der Nachbarn kann sich auf die Euklidische Distanz oder die Wegstrecke beziehen. Diese Anforderung wird mit **Nachbarn** bezeichnet.

- Auch Anfragen nach Objekten in einem bestimmten Bereich müssen unterstützt werden, wobei Bereiche geometrischen Figuren wie Kreisen oder Polygonen entsprechen. Dazu gehören Enthaltenseinsanfragen nach Objekten in Räumen, aber ebenso die Zuordnung von Räumen zu Gebäuden und Stockwerken oder Knoten eines Graphen zu Räumen. Damit kann es auch Enthaltenseinsanfragen für symbolische Bezeichner geben, welche ebenfalls durch das Gebäudemodell beantwortet werden sollen. Diese Anforderung wird mit **Bereich** bezeichnet.

- Gebäudemodelle benötigen eine Beschreibung, die eine Visualisierung in Form einer Karte zulässt. Hierfür reicht meist eine 2,5D oder 3D geometrische Beschreibung der Objekte und Bauteile aus. Diese Anforderung wird mit **Visualisierung** bezeichnet.

- Kollisions- oder Sichtlinienanfragen müssen unterstützt werden, um Navigation nach Landmarken oder Mapmatching zu erlauben. Diese Anforderung wird mit **Sichtlinie** bezeichnet.

- Es muss im Modell möglich sein, explizit Orte oder Objekte von Interesse zu modellieren, da diese für viele Klassen von Diensten relevant sind. Diese Anforderung wird mit **POIs** bezeichnet.

- Aufgrund der großen Diversität der verschiedenen Anwendungen muss es möglich sein, beliebige weitere Attribute für Elemente des Gebäudemodells zu vergeben. Diese Anforderung wird mit **Tagging** bezeichnet.

2.5.4 Anforderungen an die Positionsbestimmung

Ähnlich vielfältig, wie die Anforderungen an die Umgebungsmodelle, sind die Anforderungen an die Lokalisierungssysteme aus Sicht der I-LBS-Anbieter. Dabei hängen die Anforderungen vor allem von der Art des Dienstes ab. In diesem Abschnitt wird daher von Anwendungen auf dem Smartphone des Nutzers ausgegangen, die mit aktuellen LBS aus Außenbereichen vergleichbar sind. Die Aufzählung orientiert sich dabei an den Eigenschaften von Ortungssystemen, wobei einige der Anforderungen, wie Kosten und Flächendeckung, durchaus als komplementär anzusehen sind.

- Die **Genauigkeit**, im Englischen *accuracy*, der Positionsbestimmung soll nicht schlechter als 1-2 Meter sein, damit Stockwerke und Räume in den meisten Fällen korrekt erkannt werden. Aus demselben Grund sollen symbolische Orte mit mindestens 90 % Wahrscheinlichkeit korrekt identifiziert werden.

- Die **Präzision**, im Englischen *precision*, der Positionsbestimmung soll in einem ähnlichen Bereich liegen, so dass im Fehlerfall die Positionsbestimmung nicht um mehr als einen Raum oder ein Stockwerk falsch liegt.

- Für viele Dienste ist zudem eine **Abschätzung von Genauigkeit und Präzision** in Form einer Wahrscheinlichkeitsverteilung sinnvoll, damit eine Qualitätskontrolle und -sicherung von Dienstseite aus erfolgen kann.

- Die Positionsbestimmung soll **flächendeckend** und in möglichst vielen Gebäuden möglich sein, ohne Einschränkungen an bestimmte Infrastrukturen oder Umgebungen.

- Die Positionsbestimmung soll **Energie-effizient** sein und den Akku des mobilen Endgerätes schonen.

- Das Lokalisierungssystem soll möglichst günstig sein und keine dedizierte **Hardware** erfordern.

- Die **Kosten** für die Inbetriebnahme und den Betrieb sollen möglichst gering sein, dazu gehört auch der Zeitbedarf zur Einrichtung und Instandhaltung.
- Das Lokalisierungssystem soll die **Privatsphäre** des Nutzers schützen, indem Ortsinformationen nur mit Wissen und Willen des Nutzers an dritte weitergegeben werden. Dazu muss Endgeräte-zentrische Positionsbestimmung erfolgen.
- Das Lokalisierungssystem soll mit der Anzahl der Ziele **skalieren**.
- Das Lokalisierungssystem soll **ausfallsicher** sein und flexibel auf Änderungen in der Umgebung reagieren.
- Das Lokalisierungssystem soll Positionsinformationen in einer genügend hohen **Frequenz** zur Verfügung stellen. Bei einer angenommenen Bewegungsgeschwindigkeit von $0 - 8$ m/s muss die Update-Rate bei mindestens 4 Hz liegen, um die Anforderungen an Genauigkeit und Präzision noch zu erfüllen.

Kapitel 3

Ein hybrides Umgebungsmodell für I-LBS

Ein wichtiger Bestandteil von ortsbezogenen Diensten sind Umgebungsmodelle. Diese dienen nicht nur als Referenz für Positionsangaben, sondern ebenfalls als Informationssystem über die Umgebung. Zudem werden sie zur Unterstützung bei der Positionsbestimmung, z.B. für Mapmatching oder Signalausbreitungsmodelle, oder als Grundlage für bestimmte Anfragen, wie Sichtbarkeits-, Routing- oder Enthaltenseinsanfragen (siehe auch Abschnitt 2.5.3) eingesetzt.

Im Bereich ortsbezogener Dienste sind insbesondere zwei Arten von Gebäudemodellen verbreitet: Vektor- und Raster-basierte Modelle. Vektor-basierte Modelle repräsentieren Gebäude durch Ecken, Kanten und Flächen, also durch eine geometrische Beschreibung von Objekten und Bauteilen. Raster-basierte Modelle hingegen beschreiben Gebäude durch gefärbte Zellen in einem Raster, wobei im Normalfall weiße Farbe begehbare Flächen symbolisiert und demgegenüber Objekte oder Bauteile mit Schwarz markiert sind. Im Folgenden werden zunächst verwandte Arbeiten auf dem Gebiet der Gebäudemodelle vorgestellt und anschließend jeweils ein selbst entwickeltes Modell erläutert und dessen Vor- und Nachteile diskutiert. Zuletzt wird ein hybrides Modell eingeführt, welches die jeweiligen Vorteile miteinander kombiniert und die Nachteile vermeidet.

3.1 Verwandte Arbeiten

Es existieren bereits einige Ansätze zur Modellierung von Gebäuden für unterschiedliche Zwecke. Dabei gibt es sowohl Standards aus dem Bereich der Bauzeichnungen, wie DXF (Drawing Interchange Format) [9] oder IFC (Industry Foundation Classes) [78], als auch aus dem Geoinformationssystem (GIS) Umfeld, wie CityGML [40], oder ursprünglichen Outdoor-Kartenanbietern für ortsbezogene Dienste, wie Google Maps mit KML (Keyhole Markup Language) [143] oder OpenStreetMap [2].

Dabei sind die Bauzeichnungen aufgrund der Komplexität und dem Schwerpunkt auf visuelle und strukturelle Darstellung des Gebäudes für Bauzwecke kaum als semantisches

Gebäudemodell geeignet, bietet wohl aber eine detaillierte Grundlage für die Extraktion von geometrischen Informationen. Aus diesem Grund wird in diesem Abschnitt vor allem auf eng verwandte Arbeiten aus dem Umfeld der kontext- und ortsbezogenen Dienste und dem Bereich GIS eingegangen.

Im Folgenden werden die bekanntesten Ansätze vorgestellt und auf die Erfüllung der Anforderungen aus Abschnitt 2.5.3 eingegangen. Dabei orientiert sich die Auswahl an [60], wobei aufgrund der Weiterentwicklung der einzelnen Ansätze neue Mechanismen vor allem im Indoor-Bereich aufgenommen und diskutiert werden.

OpenStreetMap und Indoor-OSM

Das im Jahr 2004 gegründete OpenStreetMap Projekt stellt durch Nutzer selbst zusammengetragene Kartendaten für jedermann frei zur Verfügung [2]. Die Daten werden dabei durch GPS-Aufzeichnungen gewonnen, die mithilfe von Satellitenbildern auf die Erdoberfläche projiziert und manuell korrigiert werden und zudem zusätzlich mit bestimmten Eigenschaften, wie Einbahnstraßen und Abbiegeverboten, annotiert werden. Durch die Vielzahl von aktiven Nutzern weist OpenStreetMap in manchen Bereichen gar eine deutlich höhere Auflösung auf, als vergleichbare kommerzielle Systeme. In Gebieten mit wenigen Einwohnern und kaum aktiven Nutzern können aber auch Informationen fehlen oder nur unvollständig erfasst worden sein. OpenStreetMap weist oftmals insbesondere qualitativ hochwertige Informationen über Wander- und Radwege auf. Die Daten werden in einem eigenen XML-basierten Format gespeichert und stehen in verschieden großen Ausschnitten bis hin zum sogenannten *planet file* für die ganze Welt online zur Verfügung. Sie lassen sich auch herunterladen und offline nutzen. [2]

OpenStreetMap kommt mit drei Basisklassen aus, die durch bestimmte Eigenschaften in Form von Tags, also Schlüssel/Wert-Paaren, genauer spezifiziert werden können [2]:

- Ein *Node* definiert einen Punkt im Raum durch die Angabe einer WGS84-Koordinate und der optionalen Höhe über dem Meeresspiegel. Ein Node kann dabei noch beliebige weitere Eigenschaften in Form von Tags beinhalten, z.B. visuelle Informationen, wie die Zugehörigkeit zu bestimmen Layern, oder anwendungsspezifische Informationen, wie der Typ einer Kreuzung.

- Ein *Way* definiert einen Weg über einen Linienzug zwischen mehreren Punkten. Dabei kann dieser entweder ein offenes Polygon darstellen, d.h. Start- und Endpunkt stimmen nicht überein, was einem Wegabschnitt oder Straßenzug entspricht, oder ein geschlossenes Polygon, was einer bestimmten Fläche oder einem Umriss, wie einem Gebäude oder einem Platz, entspricht. Auch hier stehen bestimmte Tags zur Verfügung, um die Art des Ways genauer festzulegen. Die vordefinierten Möglichkeiten reichen von Autobahn über Einbahnstraßen zu visuellen Informationen, wie dem Bewuchs einer Freifläche.

- Eine *Relation* besteht aus einer geordneten Liste von Node- und/oder Way-Elementen und definiert einen logischen oder geographischen Zusammenhang der Elemente. Ty-

pischerweise beschreibt eine Relation Routen und längere Straßenzüge, eine Sammlung von Polygonen oder Einschränkungen, wie Abbiegeverbote. Auch Relationen können wieder zusätzliche Tags haben, und die Elemente einer Relation können innerhalb dieser eine bestimmte Rolle einnehmen.

Bei OpenStreetMap wird die große Freiheit klar, die durch den Einsatz der Tags erreicht wird. Durch die drei einfachen Basisklassen lassen sich alle Elemente und die für viele Dienste notwendigen Informationen abbilden, indem bei Bedarf die zusätzlich benötigten Informationen durch eine standardisierte Beschreibung von Schlüsselelementen hinzugenommen werden können. Dadurch werden neben den geometrischen Informationen der Nodes und den topologischen Informationen der Ways auch symbolische Bezeichner, wie Straßennamen, ermöglicht. So können beispielsweise auch öffentliche Verkehrsmittel modelliert oder das Innere von Gebäuden dargestellt werden.

Zur Modellierung von Gebäudeplänen wurde OpenStreetMap in einem Forschungsprojekt Indoor-OSM der Universität Heidelberg erweitert [37]. Dabei wurde eine Ontologie zur Abbildung eines Gebäudes verwendet, die anschließend durch die Hinzunahme von fest definierten Tags in OSM integriert werden kann. Die Ontologie modelliert dabei ein Gebäude sowohl durch die äußere Erscheinungsform, wie Art und Farbe des Daches oder der Fassade, als auch durch den inneren Aufbau in mehrere Stockwerke, die aus sogenannten Gebäudeteilen bestehen. Diese bilden Räume und Gänge genauso ab, wie auch Türen, Treppen oder Fenster. Zudem ist es möglich, explizit Points of Interest anzugeben. Da Bauteile durch passende Basisklassen abgebildet werden, ist es möglich, weitere Tags zu vergeben.

Evaluiert wurde Indoor-OSM anhand der beispielhaften Modellierung eines Gebäudes. Dazu wurden die Daten mithilfe eines Editors manuell in das OpenStreetMap-Format gebracht. Die Arbeit zeigt, dass es zwar durchaus möglich ist, Gebäudemodelle mit OpenStreetMap darzustellen. Ob sich jedoch die manuelle Bereitstellung von Gebäudedaten durch eine breite Masse an Nutzern durchsetzen lässt, ist aufgrund des großen Aufwands und der notwendigen Komplexität des Modells fraglich.

Indoor-OpenStreetMap erfüllt dabei grundlegend alle Anforderungen an Umgebungsmodelle aus Abschnitt 2.5.3. Aufgrund des Ursprungs aus einer globalen Kartenbasis werden allerdings alle Koordinaten einheitlich im WGS84 Koordinatensystem angegeben. Dadurch müssen auch alle Positionsangaben von Ortungssystemen erst einmal in das entsprechende Koordinatensystem transferiert werden, wodurch die komplexe Aufgabe der Koordinatentransformation vom Umgebungsmodell Anbieter an die Positionsbestimmung ausgelagert wird. Zudem leidet die Lesbarkeit der Daten an der großen Anzahl an unterschiedlichen Tags, was durch die Einführung von neuen Elementen hätte vermieden werden können.

Google Indoor Maps

Das Unternehmen Google stellt mit Google Maps Kartenmaterial zur nicht-kommerziellen Nutzung online zur Verfügung. Die Karten basieren auf der Keyhole Markup Language

(KML) und haben ihren Schwerpunkt in der Visualisierung der Daten. KML ist mit der aktuellen Version 2.2 seit 2008 ein Standard des Open Geospatial Consortium (OGC) [143]. Ähnlich zu OpenStreetMap werden verschiedene Objekte explizit modelliert, wobei KML deutlich mehr Objekttypen besitzt und somit für die meisten Menschen leichter verständlich ist. Ein Herzstück für geographische Informationen sind die Placemark-Elemente, die sowohl einem geometrischen Punkt als auch einer Fläche entsprechen können, textuelle Beschreibungen erlauben und zur Visualisierung eine bestimmte Perspektive angeben können. Die geometrischen Informationen werden dabei durch Punkte, Linien oder Polygone mit WGS84 Koordinaten angegeben. Zudem kann man Grafiken einbetten und die Darstellung von Objekten durch Stilinformationen verändern.

Seit kurzem bietet Google auch Indoor-Karten unter dem Namen Google Indoor Maps an, diese können jedoch im Augenblick in Deutschland nur mit der mobilen Version von Google Maps genutzt werden. Dazu werden Nutzer aufgefordert, selbstständig Gebäudepläne in Form von Bilddaten hochzuladen und semantische Informationen, wie Stockwerksangaben, Adressen und Namen des Gebäudes anzugeben. Google kümmert sich dann um die Erzeugung des semantischen Modells im KML-Format. Über den genauen Vorgang liegen aktuell leider keine öffentlichen Informationen vor. Zudem bietet Google mit dem Google Maps Floor Plan Marker[1] eine mobile Anwendung zu Verbesserung der Positionsbestimmung in Gebäuden.

Auch Google Indoor Maps erfüllt grundlegend die Anforderungen aus Sicht der ortsbezogenen Dienste aus Abschnitt 2.5.3. Allerdings gilt Ähnliches wie für OpenStreetMap, nämlich dass durch die Herkunft aus einer globalen Kartenbasis lokale Koordinatensysteme erst umgerechnet werden müssen.

Nexus

Nexus ist ein Projekt der Universität Stuttgart, das seit 2003 in einem eigenen Sonderforschungsbereich bearbeitet wird [110]. Das Ziel des Projekts ist die Bereitstellung einer Plattform als Grundlage für eine Vielzahl von kontextbezogenen Systemen, die auf das sogenannte *Augmented World Model* zurückgreifen. Dieses bildet ein Umgebungsmodell für die ganze Welt, in dem Objekte in einer flachen Hierarchie angeordnet sind und über Referenzen miteinander verknüpft werden. Im Folgenden wird auf die in der Arbeit von Nicklas [93] genannten grundlegenden Komponenten näher eingegangen und die Modellierung von Gebäuden vorgestellt.

Nexus unterscheidet bei der Modellierung zwischen dem *NexusStandardAttributeSchema* für verschiedene Eigenschaften von Objekten, die wiederum im *NexusStandardClassSchema* beschrieben werden. Die Attribute erlauben beispielsweise geometrische Beschreibungen von Objekten, wie die *NexusDirectionType* und *NexusGeometryType*, welche sowohl eine Repräsentation im standardisierten Well-Known-Text-Format (WKT) als auch in GML zulässt. Alle Datenobjekte verfügen über eine im Augmented World Model eindeutige ID und können reale oder virtuelle Objekte der realen Welt darstellen. Objekte mit Ortsbezug

[1] zu finden unter https://play.google.com/

sind die *SpatialObjects*, welche durch eine Position und ein geometrisches Ausmaß festgelegt werden.

Für die Modellierung von Gebäuden mit deren Innenbereichen stellt Nexus ein *BuildingObject* zur Verfügung, welches von *SpatialObject* erbt. Realisierungen von *BuildingObjects* sind beispielsweise Gebäude, Stockwerke oder Räume, die mithilfe einer Relation *PartOf* miteinander in Verbindung gesetzt werden. Während Nexus in Außenbereichen zwar die Topologie von Straßen von den geometrischen Informationen durch sogenannte *NavigationalObjects*, nämlich *NavigationNodes* und *NavigationEdges*, die einen Wegegraphen aufspannen, trennt, wurde dieser Ansatz noch nicht für Bereiche innerhalb von Gebäuden umgesetzt. Allerdings wird in [94] über die Integration von CityGML in Nexus für die Modellierung von Bereichen in Gebäuden gesprochen.

Grundsätzlich eignet sich das Augmented World Model als Basis für viele ortsbezogene Dienste in Gebäuden, allerdings ist ein auf diesen Bereich spezialisiertes Modell, gerade im Bereich von topologischen Beziehungen in Gebäuden, besser für dieses Einsatzgebiet geeignet.

CityGML

Mit CityGML existiert ein Ansatz zur Modellierung von Gebäuden, der zwar auf die detaillierte Darstellung von Stadtmodellen spezialisiert ist, aber auch eine Detailstufe für Innenbereiche besitzt. CityGML ist seit 2008 wie KML ein Standard der OGC und seit Anfang 2012 in der Version 2.0 veröffentlicht [40]. Dabei greift CityGML auf den bekannten OGC Standard GML zurück und ist konform zu den Richtlinien der OGC als Anwendungsschema von GML 3.1.1 ausspezifiziert worden. CityGML ist aufgrund der verschiedenen Detailstufen in verschiedene Module aufgeteilt, wobei an dieser Stelle nur auf die höchste Detailstufe LOD4 eingegangen wird, welche der Modellierung von Bereichen innerhalb von Gebäuden entspricht. Dazu wird zunächst das CityGML Core Modul vorgestellt und anschließend das Building Modul für Gebäude erörtert.

Das CityGML Core Modul definiert grundlegende Objekttypen wie das *CityObject*, das von dem GML Typ *Feature* abgeleitet ist. Das *CityObject* ist die Basisklasse für weitere Objekte, wie Gebäudeobjekte, und bietet die Möglichkeit externe Referenzen beispielsweise in das topographische Informationssystem ATKIS anzugeben.

Das Modul Building beschreibt ein Gebäude in der höchsten Detailstufe LOD4. Als grundlegender abstrakter Objekttyp wird hier das *AbstractBuilding* eingeführt, welches als *CityObject* ebenfalls von GML Feature erbt. *AbstractBuilding* beinhaltet Informationen zur Funktion oder Nutzung eines Gebäudes, Jahresangaben, das äußere Erscheinungsbild und strukturelle Informationen, wie die Anzahl der Stockwerke und die Höhe des Gebäudes, aber auch geometrische Informationen in Form von 2- oder 3-dimensionaler GML Geometrie. Diese erlauben zudem die Angabe eines Referenzsystems zu den einzelnen Koordinaten. Realisiert wird ein *AbstractBuilding* durch die Typen *Building* oder *BuildingPart*. Zu jedem *AbstactBuilding* werden zudem die enthaltenen Räume durch den Typ *Room* angegeben. Diese können ebenfalls bestimmte Klassen oder eine bestimmte Nutzung aufweisen und können durch Objekte von Typ *BuildingFurniture* möbliert sein. Die geometrischen

Informationen zum Ausmaß des Raumes werden in 3D durch die den Raum umgebenden Oberflächen, also Boden, Wände und Decken, modelliert. Auch Türen und Fenster werden explizit modelliert, und Treppen oder Aufzüge können als *IntBuildingInstallation* angegeben werden.

CityGML eignet sich vorzüglich zur Visualisierung von Gebäuden in hohem Detailgrad, unter Einbezug der geometrischen 3D Struktur der Wände, von Möbelstücken im Raum bis hin zu Texturen für alle möglichen Oberflächen. Bei hohem Detailgrad eignet sich CityGML sogar als Grundlage für die Positionsbestimmung mithilfe von Tiefenbildern [70]. CityGML eignet sich grundsätzlich für die Modellierungen von Indoor Umgebungen für ortsbezogene Dienste nach den Anforderungen aus Abschnitt 2.5.3, allerdings ist das Modell unter den genannten Beispielen bei weitem das komplexeste. Bislang ist jedoch noch keine explizite Modellierung von Wegenetzen oder Begehbarkeitsinformationen im Standard vorgesehen. Dementsprechend lassen sich topologische Informationen hautsächlich implizit ermitteln, indem beispielsweise zwei Räume, die durch eine gemeinsame Tür miteinander verbunden sind als topologisch verbunden betrachtet werden.

Zusammenfassung

Nachdem verschiedene bestehende Ansätze zur Modellierung von Gebäuden untersucht wurden, lässt sich feststellen, dass diese zwar bereits die grundlegenden Anforderungen erfüllen (siehe Tabelle 3.1), es allerdings durchaus noch Verbesserungsbedarf in Bezug auf Darstellung und explizit modellierte Informationen gibt. Dementsprechend beschäftigt sich diese Arbeit im Folgenden mit zwei verschiedenen Arten von Umgebungsmodellen, die die Anforderungen aus Sicht der Dienstanbieter und der Positionsbestimmung optimal erfüllen und sich in einem hybriden Modell kombinieren lassen.

	Transf.	Routing	Nachbarn	Bereich	Visual.	Sichtl.	POIs	Tagging
I-OSM	-	x	o	o	x	o	x	x
I-Maps	-	x	o	o	x	o	x	x
Nexus	x	-	o	o	x	o	x	x
CityGML	x	-	o	o	x	o	o	x

Tabelle 3.1: Bewertung bestehender Umgebungsmodelle anhand der Anforderungen.

3.2 BIGML - ein Vektor-basiertes Gebäudemodell für I-LBS

Umgebungsmodelle wurden bereits in Abschnitt 2.4 angesprochen. Allerdings sind die meisten Umgebungsmodelle für bestimmte Einsatzszenarien entworfen und optimiert worden und eignen sich nur bedingt für die ganze Bandbreite von ortsbezogenen Diensten in

Gebäuden. Building Information GML (BIGML) [61] bietet ein Vektor-basiertes Umgebungsmodell mit topologischen Informationen, welches für eine Vielzahl an Diensten und Einsatzszenarien eignet. Das Modell ist in Form eines GML Applikationsschemas [103] gegeben und eignet sich somit ebenfalls als Austauschformat zwischen LBS Anbieter und Umgebungsmodell-Anbieter oder kann einem Positionsbestimmungssystem zur Verfügung gestellt werden. Eine Besonderheit von BIGML ist die Unterstützung der automatischen Erzeugung von Wegegraphen aus dem Modell, welche in Abschnitt 3.2.3 genauer vorgestellt wird und die einfache Austauschbarkeit der Wegegraphen, welche durch die Trennung der geometrischen und topologischen Informationen von dem Graphen erreicht wird.

3.2.1 Aufbau von BIGML

Entsprechend der Klassifikation von Becker und Dürr [14] handelt es sich bei BIGML um ein hybrides Umgebungsmodell, das hierarchische und Graph-basierte Elemente vereint und sowohl geometrische als auch symbolische Koordinaten zulässt. Die Vorstellung der einzelnen Typen orientiert sich dabei an [61].

Die hierarchische Grundstruktur wird durch die Aufteilung eines Gebäudes bereits vorgegeben. Alle in der folgenden Aufzählung genannten Elemente repräsentieren Objekte in der realen Welt und erben von der Klasse **AbstractBuildingInformation**. Diese Klasse kann eine beliebige Anzahl von zusätzlichen Eigenschaften in Form von *PropertyTags* aufweisen. Diese bestehen aus Schlüssel/Wert-Paaren und lassen somit die Vergabe von beliebigen Attributen für die Elemente eines Gebäudes zu.

- Das Wurzelelement der Hierarchie stellt ein **Site**-Objekt dar, welches ein Grundstück in der realen Welt repräsentiert. Üblicherweise ist damit auch die Zugehörigkeit zu einer bestimmten Verwaltungseinheit verbunden. Globale Kartendatenanbieter modellieren in diesem Bereich meist keine Wege mehr, so dass dies im Umgebungsmodell selbst übernommen wird. Aus diesem Grund ist die Modellierung einer Site notwendig, um Übergänge zwischen globalen Kartendaten und lokalen Daten zu ermöglichen.

- Auf einer Site können wiederum mehrere **Building**-Objekte platziert sein, welche den eigentlichen Gebäuden entsprechen. Hierbei ist es auch möglich, ein komplexes Gebäude in mehrere Teilgebäude zu zerlegen. Wichtig sind in letzterem Fall vor allem die Übergänge zwischen den Gebäudeteilen und zwischen der Site und dem Gebäude.

- Jedes Gebäude besteht aus mindestens einem **Floor**-Objekt, welches einzelne Stockwerke beschreibt. Auch hier muss wiederum Wert auf die topologische Verknüpfung von den einzelnen Stockwerken gelegt werden.

- In jedem Stockwerk gibt es mindestens ein **Room**-Objekt, das für einen einzelnen Raum steht.

- Jeder Raum hat mindestens einen Eingang in Form einer Tür, Treppe oder eines Aufzugs, welche als Konkretisierung des abstrakten Typs **Portal**, beispielsweise als

DoorPortal-Objekt dargestellt sind. Die Portale bilden die topologische Verbindung zwischen zwei Räumen explizit ab.

- Zudem kann ein Raum auf beliebig viele **POI**- oder **AOI**-Objekte verweisen. Erstere stellen ein Objekt oder einen Ort von Interesse dar, während letztere dazu dienen, ganze Gegenden von Interesse zu gruppieren.

Innerhalb der Hierarchie haben alle Elemente sowohl Referenzen auf die Kind-Objekte, als auch auf das jeweilige Eltern-Objekt um Enthaltenseinsbeziehungen effizient abfragen zu können (vgl. Abbildung 3.1).

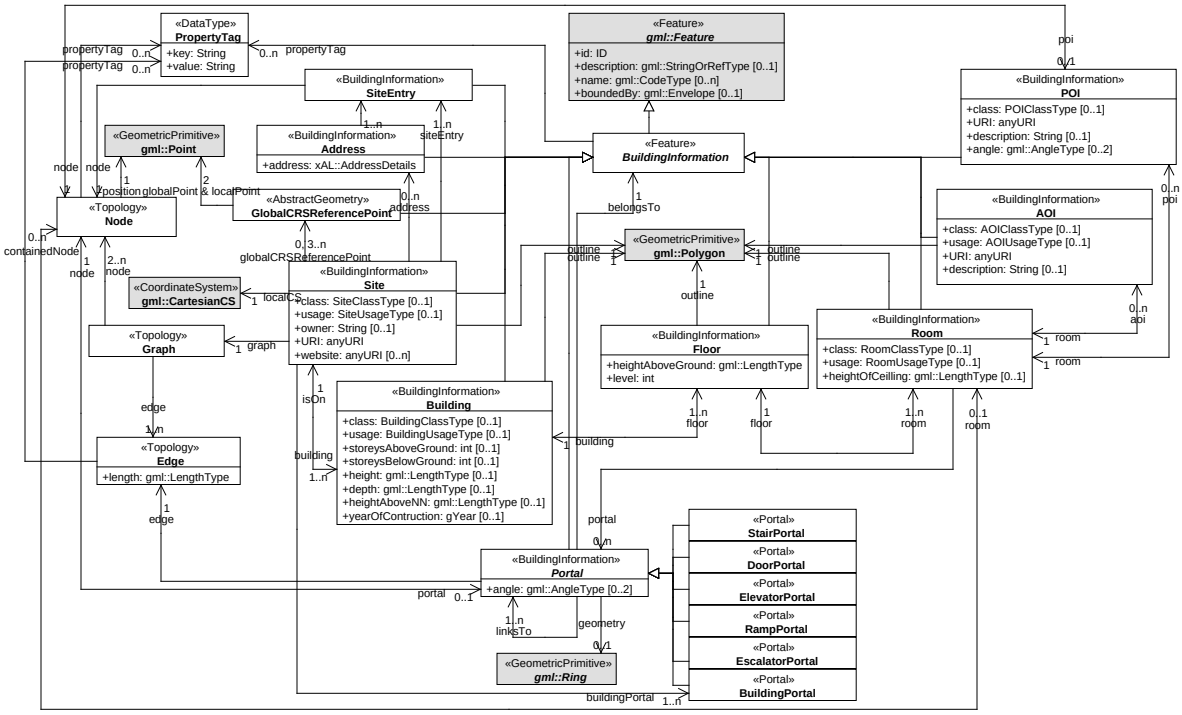


Abbildung 3.1: BIGML UML-Diagram mit GML-Klassen (grau). [61]

Durch die *Portal*-Objekte wird die Topologie des Gebäudes explizit abgebildet, da hierdurch bereits alle Verbindungen zwischen Räumen, auch über mehrere Stockwerke hinweg, als auch Verbindungen zwischen Räumen und Außenbereichen modelliert werden. Aus dieser expliziten Topologie zusammen mit der Raumgeometrie kann die automatische Grapherzeugung unterstützt werden.

Neben der expliziten Hierarchie und den Tags gibt es noch weitere Elemente und Attribute, die im Folgenden detailliert vorgestellt werden:

Zu jeder **Site** gehört eine Anzahl von Eingängen zum Grundstück vom Typ *SiteEntry* mit zugehörigen Adressen vom Typ *Address* und eine Referenz zu allen Eingängen zu den

Gebäuden vom Typ *BuildingPortal*. Als Wurzelknoten legt die *Site* ebenfalls das lokale Koordinatensystem vom Typ *gml::CartesianCS* sowie drei linear unabhängige Referenzpunkte in einem globalen Koordinatensystem fest. Letztere sind von Typ **GlobalCRSReferencePoint** und erlauben die Berechnung der Parameter zur Koordinatentransformation zwischen dem lokalen System und einem globalen System wie WGS84. Für die geometrische Repräsentation verweist eine *Site* auf ein *gml::Polygon* und für die Topologie auf den Wegegraphen vom Typ *Graph*. Zudem besitzt eine *Site* noch einige Attribute, wie eine optionale Website, den Besitzer, einen eindeutigen Identifier und die von CityGML [40] adaptierten optionalen Attribute *usage* und *class*, die die Art und die Nutzung des Gebäudes abbilden.

Auch ein **Building**-Objekt verweist auf seine geometrische Beschreibung in Form eines *gml::Polygons* und hat ebenfalls die optionalen Attribute *usage* und *class*. Zudem kann die Höhe und Tiefe des Gebäudes, die Höhe des Erdgeschosses über dem Meeresspiegel, die Anzahl der Stockwerke über und unter der Erde und das Baujahr angegeben werden.

Ein **Floor**-Objekt hat die Nummer des Stockwerks und die Höhe des Stockwerks über dem Boden als Attribute und verweist ebenfalls auf seine geometrischen Ausmaße in Form eines *gml::Polygons*.

Ein **Room**-Objekt verweist neben seinem Umriss auch auf alle enthaltenen Knoten des Graphen vom Typ *Node*, sowie eine beliebige Zahl an *POI*- und *AOI*-Objekten. Zudem hat jeder Raum die optionalen Attribute der Höhe des Raumes, *usage* und *class*.

Ein **Portal**-Objekt verweist ebenfalls auf seine geometrischen Ausmaße, das zugehörige Portal im angrenzenden Raum, die zugehörige Kante und den zugehörigen Knoten im Graphen. Zudem können zwei Winkel angegeben werden, welche den Sektor definieren, in dem das Portal im Elternraum sichtbar ist. Ein *Portal* hat dabei immer eine konkrete Realisierung durch einen der folgenden Typen: *StairPortal*, *DoorPortal*, *ElevatorPortal*, *RampPortal*, *EscalatorPortal* oder *BuildingPortal*.

Auch für ein **POI**-Objekt können zwei Winkel angegeben werden, unter denen dieses im Elternraum sichtbar ist. Zudem hat jeder *POI* einen eindeutigen Bezeichner, sowie optional eine Klasse und Beschreibung. Ein *POI* referenziert ebenfalls den zugehörigen Knoten im Graphen.

Ein **AOI**-Objekt hat auch eine Bezeichnung, jedoch ist diese im Unterschied zum *POI* nicht eindeutig. Unter diesem Bezeichner werden mehrere *AOIs* zusammengefasst und erlauben die Beschreibung des semantischen Zusammenhangs größerer Flächen außerhalb der strikten Hierarchie. Da ein einzelnes *AOI*-Objekt auch nur einen Teil des Elternraumes einnehmen kann, hat es zudem ein *gml::Polygon* zur Beschreibung des Umrisses, sowie die optionalen Attribute einer Beschreibung, sowie *usage* und *class*.

SiteEntries stellen nicht nur den Eingang zum Grundstück dar, sondern bieten ebenfalls einen Verweis auf einen Knoten im Graphen und somit die Möglichkeit der Anbindung des Gebäudemodells an Straßenkarten. Die zugehörigen Adressen werden durch **Address**-Objekte modelliert, welche in der Syntax der Extensible Address Language [26] angegeben werden.

Neben den bereits genannten Typen gibt es noch die topologischen Typen *Graph*, *Edge* und *Node*, die den Graphen und seine Knoten und Kanten beschreiben.

Das einzige **Graph**-Objekt für eine Site enthält alle zugehörigen Knoten und Kanten

des Wegegraphen des Grundstücks, auch wenn diese nicht zwingendermaßen eine einzelne Zusammenhangskomponente bilden.

Die **Node**-Objekte sind in das lokale Koordinatensystem eingebettet und enthalten neben den ausgehenden Kanten auch die geometrische Position des Knotens. Weitere Verweise zu einem Raum, POI oder Portal oder die Angabe von beliebig vielen *PropertyTags* sind zudem möglich.

Die **Edge**-Objekte haben Verweise auf die beiden angrenzenden Knoten und zudem aus Effizienzgründen die Länge der Kante gespeichert. Auch hier ist die Angabe von beliebig vielen *PropertyTags* möglich.

3.2.2 Vorteile und Einschränkungen

Die Vorteile von Vektor-basierten Modellen im Allgemeinen und BIGML im Besonderen liegen auf der Hand und werden im Folgenden in Anlehnung an die aufgestellten Anforderungen aufgeführt. Im Folgenden werden die Stärken und Schwächen des Modells angesprochen:

- Eine große Stärke ist die Visualisierung, da Vektor-basierte Modelle verlustfrei skalierbar sind und somit eine beliebig detaillierte Ansicht zur Verfügung stellen.
- Auch die Angabe von beliebigen weiteren Attributen für bestehende Elemente und die explizite Modellierung von interessanten Orten und Objekten ist eine Stärke von BIGML.
- Durch die geometrische und symbolische Beschreibung von Orten in BIGML ist es möglich, eine Position entsprechend beiden Repräsentationen anzugeben.
- Ein großer Vorteil von BIGML ist zudem die explizite Modellierung von Wegegraphen, die Routing, Navigation und Weglängenberechnungen effizient unterstützen.
- Anfragen zu k-nächsten Nachbarn lassen sich in BIGML sowohl in euklidischer Distanz, als auch bezüglich der Weglänge effizient berechnen. Letztere allerdings nur bezüglich der Knoten des Graphen.
- Bereichsanfragen bezüglich vorgegebener Elemente (POI oder AOI in Räumen, Stockwerken, etc.) lassen sich effizient beantworten, da hierbei die Enthaltenseinsbeziehung explizit modelliert ist. Anders sieht es mit Anfragen in beliebigen geometrischen Bereichen oder bezüglich nicht modellierter Ziele, wie Personen, aus. Hier muss das Enthaltensein aus den geometrischen Koordinaten berechnet werden.
- Eine Schwäche von Vektor-basierten Modellen ist es, dass Sichtlinien- oder Kollisionsanfragen, sowie Punkt-in-Polygon Tests nicht effizient unterstützt werden. Zur Sichtlinienüberprüfung ist im schlimmsten Fall die Schnittberechnung der Sichtlinie mit jeder Linie des Gebäudeplans, zumindest aber jeder Linie des aktuellen Raums notwendig. Die Anzahl der notwendigen Berechnungen hängt dabei vom Detailgrad

der Modellierung und der Komplexität des Raumes ab. Für einen Punkt-in-Polygon Test muss im schlimmsten Fall auch wieder jedes Raumpolygon getestet werden. Der Aufwand kann zwar durch Zuhilfenahme bestimmter Speicherstrukturen, wie Quadtrees oder spatialer Bäume, auf einen Teil der Räume gesenkt werden, solche Strukturen benötigen allerdings meist zusätzlichen Speicherplatz.

3.2.3 Automatische Erzeugung von BIGML-Modellen aus CAD-Daten

Eine im obigen Abschnitt nicht genannte nicht-funktionale Anforderung ist die einfache und möglichst automatische Erzeugung eines Umgebungsmodells. Gerade bei großen Bauplänen ist eine manuelle Erzeugung, beispielsweise durch Nachzeichnen von Kanten im Gebäudemodell [123, 124], nicht effizient und fehleranfällig. In [115] stellen Schäfer et al. einen Ansatz vor, wie aus CAD-Daten mit einer reinen Liniendarstellung der Gebäudegeometrie Räume und Türen erkannt werden. Zudem werden Treppen und Aufzüge auf einem bestimmten Layer gesucht und erkannt. Für die Erzeugung von BIGML-konformen Umgebungsmodellen wurde in [60] eine eigene Methodik entwickelt, um die CAD-Daten eines Gebäudes automatisch einzulesen, semantisch anzureichern und in BIGML zu speichern. Dazu werden die folgenden Schritte in der festgelegten Reihenfolge ausgeführt:

1. Ausgehend von einer Anzahl an DXF-Dateien, die gewisse Anforderungen erfüllen, werden zunächst die Informationen über die Hierarchie des Gebäudes und deren Elemente inklusive der symbolischen Bezeichner gesammelt. Die folgenden Anforderungen müssen erfüllt sein:

   - für jedes Gebäude und jedes Stockwerk existiert eine eigene DXF-Datei, deren Dateiname den Gebäudenamen und die Bezeichnung des Stockwerks als regulären Ausdruck darstellt.
   - Die symbolischen Bezeichner für Räume sind entweder als Label zu einem Raum oder als Text innerhalb des Raumpolygons vorhanden.
   - Türen, Treppen und Aufzüge sind jeweils einheitlich dargestellt.

2. Anschließend werden Portale aufgrund ihrer geometrischen Eigenschaften erkannt und mit den zugehörigen Elementen der Hierarchie verbunden.

3. Danach werden aus weiteren Quellen POIs, AOIs sowie die Eingänge zum Grundstück und deren Adressen hinzugefügt.

4. Schließlich können aus den bestehenden Informationen unterschiedliche Wegpunktgraphen erzeugt werden:

   (a) Erzeuge einen Wegpunkt für jedes Portal und eine Kante zwischen den Wegpunkten zugehöriger Portale.

(b) Erzeuge einen Wegpunkt für jeden POI und jeden Eingang zum Grundstück.

(c) Berechne einen Wegpunktgraphen für jeden Raum und den Teil des Grundstücks, der nicht von Gebäuden bedeckt ist.

(d) Verbinde die Teilgraphen mit den zuvor erzeugten Wegpunkten für Eingänge, Übergänge und POIs

Das Ergebnis ist ein Wegpunktgraph für das ganze Umgebungsmodell, welcher die Topologie des Gebäudes korrekt abbildet.

5. Das erzeugte Modell kann nun als BIGML-Datei abgespeichert werden. Möglicherweise zusätzlich benötigte Metainformationen, wie die Angabe der Referenzpunkte für das globale Koordinatensystem oder die Raumnutzung müssen manuell eingetragen werden.

Das Vorgehen erlaubt die automatische Erzeugung eines Großteils des Gebäudemodells und erfordert nur einen minimalen Anteil an manueller Interaktion. In Folgenden wird genauer auf die geometrische Erkennung von Portalen eingegangen und anschließend die Erzeugung des Wegegraphen diskutiert.

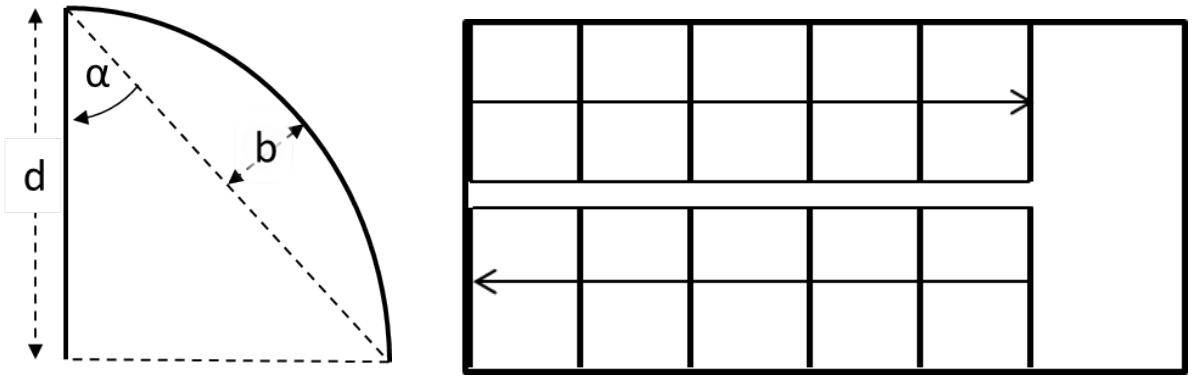
Geometrische Portalerkennung

Die Erkennung von Portalen anhand der geometrischen Linien einer CAD-Zeichnung ist essentiell, um ein topologisches Indoor-Umgebungsmodell bereitstellen zu können. In [88] wird bereits ein System vorgestellt, das CAD-Daten in Bilder umwandelt und eine graphische Türerkennung auf den Bilddaten durchgeführt. Zudem wird vorgeschlagen, die semantischen Informationen bereits in die Modellierung der CAD-Daten aufzunehmen. Shafer stellt in [120] ein System vor, welches bereits die automatische Erzeugung aus CAD-Daten ermöglicht, allerdings wird in der Arbeit nicht genauer auf den Erzeugungsprozess eingegangen. Ausgehend von [60] verwenden Schäfer et al. in [115] eine ähnliche Methodik, wobei die Portale ebenfalls aufgrund der geometrischen Eigenschaften der zugehörigen Linien identifiziert werden.

Eine normale **Tür** ist üblicherweise durch zwei Linien dargestellt, die einen Punkt gemeinsam haben und im Winkel von ca. 45° zueinander stehen, wobei genau eine der beiden Linien gebogen ist (siehe Abbildung 3.2a). Die weitere Umgebung der Tür, sprich Türstöcke und angrenzende Wände unterscheiden sich hingegen in verschiedenen Zeichnungen teilweise deutlich. Dementsprechend werden zwei geometrische Eigenschaften von Linienpaaren überprüft, um festzustellen, ob es sich um eine Tür handelt.

- Die zwei Linien haben einen Punkt gemeinsam und formen mit den beiden nicht gemeinsamen Endpunkten ein gleichschenkliges Dreieck.

- Die zwei Linien haben einen Punkt gemeinsam und liegen ungefähr in einem Winkel $\alpha = 45°$ zueinander.



(a) Darstellung einer Türe (b) Darstellung einer Treppe

Abbildung 3.2: Geometrische Erkennung verschiedener Portalobjekte.

Ist genau eine der beiden Linien gekrümmt, also $|b| > 0$, so entsprechen die Linien mit hoher Wahrscheinlichkeit einer Tür. Ist zudem die Länge d der Linien im Rahmen der Länge einer normalen Tür, also zwischen 70 cm und 180 cm lang, so ist es zudem sehr unwahrscheinlich, dass die beiden Linien zu einer Wand oder Säule gehören. Allein mithilfe dieser geometrischen Eigenschaften wurden in dem großen Datensatz des Flughafen Münchens über 95 % der normalen Türen erkannt. Allerdings ist die Erkennung spezieller Türen, wie Schiebetüren, Drehtüren oder großer Portale nicht ohne weiteres möglich, da hierfür keine standardisierten Formen vorliegen und sie dementsprechend in unterschiedlichen Zeichnungen anders dargestellt werden.

Auch für **Treppen** existiert eine große Anzahl verschiedener Darstellungen, wobei all diese Darstellungen aus einer großen Zahl von Linien gleicher Länge bestehen, die in annähernd gleichem Winkel zueinander stehen, den gleichen Abstand haben und entsprechend die Treppenstufen darstellen (siehe Abbildung 3.2b). Im Fall einer geraden Treppe sind diese Linien quasi parallel, es muss allerdings immer mit einer gewissen Ungenauigkeit in der Zeichnung gerechnet werden. Damit nicht auch Gitterstrukturen fälschlicherweise als Treppen erkannt werden, wird auch hier von einem bestimmten Abstand zwischen zwei Treppenstufen ausgegangen, die im Normalfall zwischen 10 cm für sehr steile Treppen und 50 cm für flache Treppen liegt. Da es auch Treppenstufen gibt, die keine benachbarten Stockwerke verbinden, sondern beispielsweise lediglich kleine Höhenunterschiede innerhalb eines Stockwerks ausgleichen, werden Treppen erst dann zu Portalen zwischen Stockwerken, wenn auf dem benachbarten Stockwerk ebenfalls eine Treppe ähnlichen Ausmaßes dargestellt ist und zudem mindestens 13 Treppenstufen zur Treppe gehören. Dazu ist es jedoch notwendig, dass beispielsweise in Treppenhäusern zusammengehörige Treppensegmente erkannt werde, wozu im Umkreis von 2 m zur Treppengeometrie liegende Treppen, die keine Trennwand zwischen sich beinhalten, als Treppenverbund angesehen werden. In diesem Fall wird die Anzahl der Stufen für den Stockwerkswechsel zusammengezählt. In Grenzfällen werden Treppenkandidaten für manuelle Bearbeitung markiert. Da Treppen üblicherweise einen Richtungspfeil aufweisen, kann sogar ermittelt werden, von welcher

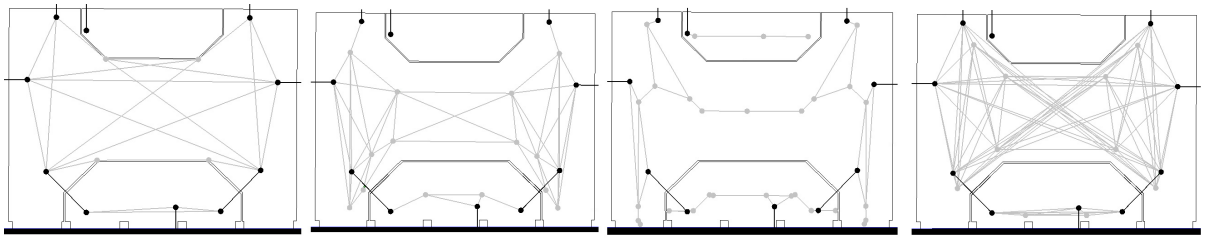
Seite die Treppe in jedem Stockwerk betreten werden kann.

Aufzüge werden wiederum in Form eines Rechtecks dargestellt, in dessen Inneren die Diagonalen eingezeichnet sind. Wird eine solche Geometrie gefunden und liegt die Fläche im Bereich von $1 - 10$ m^2 so kann von einem Aufzug ausgegangen werden. Hierbei muss jedoch aus der umgebenden Geometrie ermittelt werden, auf welcher Seite der Aufzug betreten werden kann und auf welchen Seiten sich Mauerwerk und Wände befinden.

Aufgrund der teilweise starken Unterschiede in der Darstellung von speziellen Portalen, kann eine manuelle Nachbearbeitung des erzeugten Modells nicht ausgeschlossen werden. Mithilfe der vorgestellten Algorithmen konnte ein Großteil der Portale in unterschiedlichen Bauzeichnungen automatisch erkannt werden. Für eine vollautomatische fehlerfreie Lösung sind jedoch weitere Anforderungen an die Modellierung zu stellen. Beispielsweise können die Objekte in den CAD-Daten auf einem Layer mit international einheitlicher Bezeichnung platziert werden und die Erkennung so erheblich vereinfacht werden.

Automatische Grapherzeugung

Gerade die automatische Erzeugung von Wegpunktgraphen bietet eine sehr flexible Handhabung unterschiedlicher Anforderungen an den Graphen, die von verschiedenen Anwendungsfällen her rühren. So benötigen Anwendungen zur Fußgängernavigation Wege, die das menschliche Verhalten realistisch abbilden und leicht nachzuvollziehen sind, während in der Logistik oder Roboternavigation vor allem die Kürze der Wege entscheidend ist. Die Erzeugung des Graphen kann von den restlichen Berechnungsschritten losgekoppelt auch auf bestehenden Modellen durchgeführt werden.

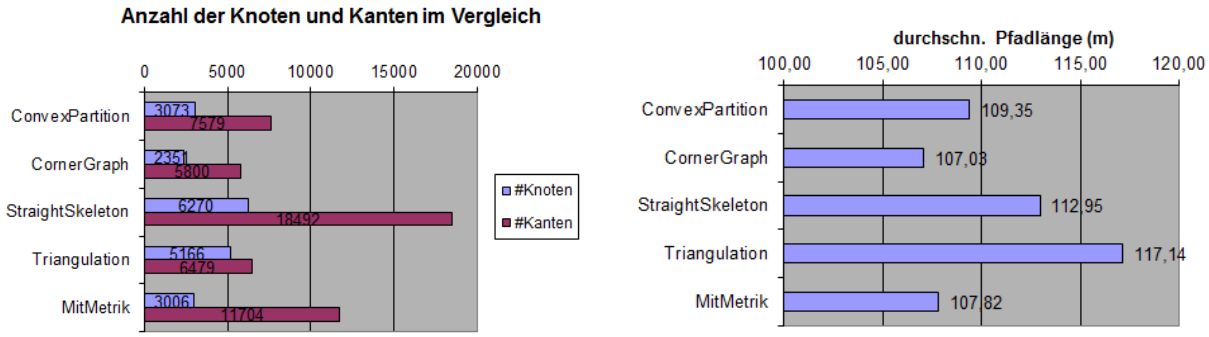


(a) Corner Graph (b) Konvexe Partitionierung (c) Triangulation (d) Modifizierter Corner Graph

Abbildung 3.3: Durch unterschiedliche Algorithmen erzeugte Wegpunktgraphen. Portalknoten (schwarz) und austauschbare Knoten und Kanten (grau). [61]

In [62] werden einige unterschiedliche Algorithmen vorgestellt und nach den Kriterien der Begehbarkeit, der Anzahl an Richtungsänderungen, der Länge der Wege, der Komplexität und der Abdeckung bewertet. Hierbei wird der Fokus auf sogenannte Rand-basierten Verfahren gelegt und keine Bewertung von Innenbereich-basierten Verfahren wie Grids gegeben. Untersucht wurden der Corner Graph, die Delaunay Triangulation, die Konvexe Partitionierung, das Straight Skeleton und ein hybrider Ansatz, der je nach Raumform entweder eine Variante des Corner Graphs oder die Konvexe Partitionierung verwendet.

Ein Ausschnitt des Modells mit unterschiedlich erzeugten Graphen findet sich in Abbildung 3.3. Für weitere Informationen zu den einzelnen Algorithmen sei auf [98, 3, 129, 60, 112] verwiesen.



(a) Komplexität des Modells

(b) Länge der Wege

Abbildung 3.4: Komplexität und Weglänge von Routinganfragen im BIGML-Modell.

Die Ergebnisse aus [62] zeigen, dass der hybride Ansatz die beste Begehbarkeit aufweist. Das bedeutet, dass die Wege ermittelt werden, die Fußgänger intuitiv am ehesten zurücklegen würden. Dabei muss sich der Ansatz bei der Gesamtlänge der einzelnen Pfade nur dem Corner Graph geschlagen geben (vgl. Abbildung 3.4b). Dieser hat allerdings eine sehr geringe Begehbarkeit, da die Knoten auf den Eckpunkten der Räume liegen, Fußgänger jedoch üblicherweise einen gewissen Mindestabstand zu Wänden einhalten. Die Komplexität ist bei fast allen Algorithmen ähnlich (vgl. Abbildung 3.4a), genauso wie eine 100%-ige Abdeckung erlangt wurde. Das bedeutet, dass von jedem Punkt in einem Raum eine direkte Sichtverbindung zu mindestens einem Knoten im selben Raum gegeben ist. Die Komplexität konnte durch eine Optimierung, nämlich das Entfernen von Kanten und Knoten, die nicht auf dem kürzesten Weg zwischen Raumportalen und/oder POIs lagen, reduziert werden, wobei allerdings auch die 100%-tige Abdeckung nicht mehr gewährleistet werden konnte.

3.3 Ein Raster-basiertes Gebäudemodell auf Basis einer Bitmap

Eines der einfachsten Beispiele für Raster-basierte Modelle sind Bitmaps, also Pixelgrafiken. Allerdings werden in dieser grundsätzlichen Form viele Anfragen, wie Routen- oder Distanzanfragen, genauso wenig unterstützt wie symbolische Bezeichner. Dennoch bieten sich Bitmaps an, wenn eine einfache und schnelle Visualisierung benötigt wird oder beispielsweise Mapmatching zur Positionsbestimmung durchgeführt werden soll. In [137] wird aus einer einfachen Bitmap mithilfe von gebräuchlichen Bildmanipulationen ein semantisches Gebäudemodell erzeugt.

Zunächst einmal werden in [137] ein paar Begriffe definiert, die für die spätere Erzeugung des Modells benötigt werden:

- Ein **Gebiet** ist eine Teilmenge von Pixeln des Bildes.
- Eine **4-Nachbarn-Flutung** ist eine Operation auf einem Pixel, die rekursiv alle Nachbar-Pixel derselben Ursprungsfarbe mit einer gemeinsamen neuen Farbe füllt.
- Der **Flutungsabschluss** eines Pixels ist das *Gebiet*, das als Ergebnis einer *4-Nachbar-Flutung* dieses Pixels mit einer in der Bitmap nicht vorkommenden Farbe gefüllt wird.
- Ein *Gebiet* ist **flutungsverbunden**, wenn es der *Flutungsabschluss* jedes seiner Pixel ist.
- Ein **Raum** ist ein nicht-schwarzes *flutungsverbundenes Gebiet*.
- Der **Außenbereich** des Modells ist der *Flutungsabschluss* der Pixelkoordinate (0,0).
- Ein *Raum* ist in einem anderen *Raum* genau dann enthalten, wenn er in dessen konvexen Abschluss liegt.

Zudem existieren drei Anforderungen an das Modell: Das Gebäude ist nach außen hin vollständig von schwarzen Linien umschlossen und von weißem Hintergrund umgeben. Türen entsprechen der obigen Definition von Räumen, insbesondere sind auch Türen komplett von Linien umgeben. Begehbare Flächen sind weiß dargestellt, Kollisionsgeometrie in schwarz.

Erfüllt eine Bitmap diese Anforderungen, so ist es möglich, einen Wegpunktgraphen darauf zu erzeugen. Zunächst einmal müssen alle Räume ermittelt werden, anschließend werden aus diesen Räumen Türen erkannt und die jeweils zugehörigen Nachbarräume ermittelt. Auf diesen Daten kann bereits ein Adjazenzgraph die Topologie des Plans abbilden, jedoch ist eine Weglängenangabe oder gar Navigation nur eingeschränkt möglich. Für die Bearbeitung einer Routenanfrage wird daher ein feinerer Wegpunktgraph erzeugt. Der Adjazenzgraph wird benutzt, um auf Raumebene kürzeste Wege zwischen Start- und Zielkoordinate zu berechnen, woraufhin alle Türsymbole auf dem Weg entfernt werden. Das Ergebnis ist ein flutungsverbundenes Gebiet, das beide Koordinaten enthält. Auf diesem Gebiet lässt sich mittels Pixel-basiertem A^*-Algorithmus der kürzeste Weg berechnen. Dieser wird sich allerdings wie ein Corner Graph verhalten, da der kürzeste Weg über Ecken und entlang von Wänden führt, und eignet sich nur bedingt zu Anzeige als Navigationshilfe. Dafür wurde in [137] ein Algorithmus ähnlich den Kreis-basierten Wegpunktgraphen entwickelt (siehe [129]), welcher den Weg von Kollisionsgeometrie wegschiebt. Im Folgenden wird genauer auf die Erkennung von Türen und Nachbarräumen sowie die Erzeugung und Nachbearbeitung des Graphen eingegangen.

3.3.1 Erkennung von Türen und Nachbarräumen

In [137] wird das Symbol einer Tür in der Bitmap anhand von verschiedenen Eigenschaften klassifiziert. Zunächst einmal ist jede Tür ein Raum entsprechend den Anforderungen. Zudem wird die Fläche des Raumes, gegeben durch die Anzahl der flutungsverbundenen Pixel, sowie die Höhe und Breite eines umschließenden achsenparallelen Rechtecks genutzt, um Türen zu erkennen. Dabei sind die jeweiligen Referenzwerte der Klasse Tür für jedes Bild separat anzugeben, da diese Maße nicht skaleninvariant sind. Weitere wichtige Eigenschaften werden zudem von einer Methode namens Zentralradar abgeleitet. Hier werden aus dem Schwerpunkt eines Raums Strahlen in 360° geworfen und der Pixel-Abstand zur Kollisionsgeometrie ermittelt. Der größte Abstand, der kleinste Abstand und ein Maß für die Abweichung von einer quadratischen Form bilden die aus dem Zentralradar gewonnenen Eigenschaften eines Raumes. Mithilfe einer Menge an vorgegebenen Trainingsdaten können mit verschiedenen Algorithmen des maschinellen Lernens die Bitmap-abhängigen Eigenschaften der Klasse Tür ermittelt und alle weiteren Türen automatisch erkannt werden. Mit bekannten Algorithmen wie Bayes, Entscheidungsbäumen oder RIPPER wurden im Testgebäude Erkennungsraten von durchgehend über 97% erreicht.

Die erkannten Türen werden genutzt, um die angrenzenden Räume zu ermitteln. Dazu wurde eine Variante des Zentralradars genutzt, die sequentiell für jede gefundene Tür durchgeführt wird: Zuerst wird der Raum, der die Tür beschreibt, in einer Markerfarbe gefüllt. Anschließend wird mit dem zentralen Radar der erste weiße Pixel entlang des Strahls gesucht. Dieser gehört zu einem Nachbarraum, welcher durch eine 4-Nachbarn-Flutung des gefundenen Pixels in einer anderen Markerfarbe für Nachbarn gefüllt wird. Das Vorgehen wird mit der Änderung wiederholt, dass der Strahl beim Auftreffen auf die Nachbarmarkerfarbe rotiert wird. Somit werden die beiden Nachbarn einer Tür gefunden und die Markerfarben anschließend wieder entfernt.

3.3.2 Erzeugung und Nachbearbeitung des Graphen

Zu diesem Zeitpunkt sind alle Türen und deren Nachbarräume bekannt. Diese Beziehung reicht zur Erstellung eines Adjazenzgraphen. Für die Erzeugung des Navigationsgraphen wird eine weitere Bearbeitung notwendig. Bei einer Routenanfrage zwischen zwei Pixel-Koordinaten werden zunächst die zu den Pixeln zugehörigen Räume ermittelt. Anschließend wird auf Basis des Adjazenzgraphen die kürzeste Reihenfolge von Räumen errechnet. Jeder Raum in dieser Reihenfolge, der einer Tür entspricht, wird insofern vergrößert (die angrenzenden schwarzen Pixel weiß gefärbt), bis ein flutungsverbundenes Gebiet entstanden ist. Dieses beinhaltet die Pixel-Koordinaten der Routenanfrage und lässt sich daher nutzen, um auf den weißen Pixeln mithilfe des A^*-Algorithmus den kürzesten Weg zu berechnen. Dabei haben benachbarte Pixel eine Entfernung von 1 und diagonal zueinander liegende Pixel eine Entfernung von $\sqrt{2}$. Als Heuristik kann die Euklidische Distanz in Pixeln angenommen werden.

Wie bereits angedeutet, entspricht der kürzeste Weg keinem realistischen und visuell einleuchtenden Weg und weist daher eine niedrige Begehbarkeit auf. Dieses Problem kann

durch eine Nachbearbeitung der Wege behoben werden. Hierfür wird der Weg entsprechend einer minimalen Entfernung tesseliert und an jedem Punkt p_i der Tesselierung ein Kreis wachsen gelassen. Der Kreis beinhaltet nur weiße Pixel und hat P_i als Mittelpunkt. Stößt der Kreis auf ein Hindernis, so wird der Mittelpunkt vom Hindernis weg verschoben, so dass der Kreis weiter wachsen kann, bis er einen festgelegten maximalen Radius erreicht oder festklemmt. Dadurch wird der Weg von Kollisionsgeometrie weggeschoben und erreicht eine höhere Begehbarkeit. Dieser Effekt wird dadurch verstärkt, dass die solcherart nachbearbeiteten Wege meist Richtungsänderungen von annähernd 90° aufweisen.

3.3.3 Vorteile und Einschränkungen

Die Vorteile von Raster-basierten Modellen im Allgemeinen und dem Bitmap-basierten Modell im Besonderen werden im Folgenden in Anlehnung an die aufgestellten Anforderungen aufgeführt. Auch hier gibt es Schwächen, die ebenfalls angesprochen werden:

- Eine Schwäche ist die Visualisierung, da Raster-basierte Modelle auf eine bei der Erzeugung gewählte Auflösung festgelegt sind und daher nur einen bestimmten Detailgrad zur Verfügung stellen.
- Auch ist keine Angabe von beliebigen weiteren Attributen für bestehende Elemente möglich, auch wenn die explizite Modellierung von interessanten Orten und Objekten durch bestimmte Farben möglich ist.
- Eine geometrische Beschreibung von Orten ist direkt durch die Pixel-Koordinaten gegeben, während eine symbolische Beschreibung durch Farbinformationen und eine separate Schlüsseldatei gespeichert werden muss.
- Ein Vorteil des vorgestellten Bitmap-basierten Modells ist die Möglichkeit der Berechnung von Wegegraphen, die Routing, Navigation und Weglängenberechnungen unterstützen.
- Anfragen zu k nächsten Nachbarn lassen sich im Bitmap-basierten Modell sowohl in euklidischer Distanz, als auch bezüglich der Weglänge berechnen. Dabei kann durch die notwendige Erzeugung des Graphen im letzteren Fall Zeit verloren gehen.
- Bereichsanfragen für Räume lassen sich bei vorliegenden Farbtabellen mithilfe von 4-Nachbarn-Flutung effizient beantworten. Bei Anfragen bezüglich beliebiger geometrischer Bereiche kann auch durch Einfärben des Bereichs und vorherige Überprüfung der Farbe das Enthaltensein effizient berechnet werden.
- Die größte Stärke von Raster-basierten Modellen ist es, Sichtlinien- oder Kollisionsanfragen extrem effizient zu unterstützen. Zur Sichtlinienüberprüfung reicht es, eine Linie zu malen und jedes Pixel auf die Farbe schwarz zu überprüfen. Somit ist lediglich der Test einiger Pixel notwendig und unabhängig vom Detailgrad der Modellierung und der Komplexität des Raumes. Für einen Punkt-in-Polygon Test reicht das

Einfärben des Punktes mit einer Markerfarbe und das anschließende Füllen des Polygons mit einer anderen Farbe und der abschließende Test, welche Farbe der Punkt nun hat. Dies lässt sich effizient berechnen.

Bei der Erzeugung des Graphen und der Türerkennung war darauf eingegangen worden, dass dazu Bitmap-spezifische Parameter notwendig sind. Diese können zusammen mit anderen semantischen Informationen beispielweise in die Exif-Tags des Bildes gespeichert werden. Somit eignet sich das platzsparende Modell auch zum Einsatz auf mobilen Geräten wie Smartphones.

3.4 Ein hybrides Vektor- und Raster-basiertes Umgebungsmodell

Wie in den letzten Abschnitten dargestllt wurde, sind manche Anforderungen an Umgebungsmodelle besser von Raster-basierten Modellen zu lösen, andere wiederum durch ein Vektor-basiertes Modell. In diesem Abschnitt wird daher ein hybrides Umgebungsmodell vorgestellt, das sich aus Vektor-basierten Elementen von BIGML und Raster-basierten Elementen des Bitmap-basierten Umgebungsmodells zusammensetzt. Das hybride Modell wurde dabei in Zusammenarbeit mit Eva Nießner in [95] entwickelt und eignet sich besonders für die serverseitige effiziente Beantwortung verschiedenster Anfragen von ortsbezogenen Diensten oder auch Positionsbestimmungssystemen. Zunächst wird auf den Aufbau und die verschiedenen Ebenen im Modell eingegangen, anschließend folgen die detaillierte Beschreibung der verschiedenen Anfragen und deren Beantwortung. Zum Schluss wird die Performanz des hybriden Umgebungsmodells im Vergleich zu den jeweiligen Einzelmodellen untersucht.

3.4.1 Aufbau eines Umgebungsmodells aus Anfragesicht

Bereits in [136] wird von Werner und Kessel der Aufbau eines hybriden Umgebungsmodells am Beispiel eines Indoor-Navigationssystems vorgeschlagen, welches die in Abschnitt 2.5.3 genannten Anforderungen besonders gut erfüllt. Demnach lassen sich die drei wohl verbreitetsten Aufgaben von Umgebungsmodellen aus Sicht eines Navigationssystems in die folgenden drei Kategorien einteilen: *Unterstützung der Positionsbestimmung*, *Auswahl von Wegen/Berechnung von Weglängen* und *Führung entlang eines Weges* aufteilen. Jede der Aufgaben benötigt zur erfolgreichen Ausführung die Beantwortung von bestimmten Anfragen. Für die erste Aufgabe ist beispielsweise die Modellierung von Freiflächen als möglicher Aufenthaltsort ebenso notwendig, wie die Modellierung von Hindernissen, wie Wänden, welche einen Aufenthalt innerhalb der Geometrie des Hindernisses unmöglich machen. Für die Wegewahl ist die Modellierung von topologischen Verbindungen zwischen verschiedenen Räumen durch Türen, Treppen und Aufzügen notwendig, wobei bestimmte Randbedingungen, wie Zugangsbeschränkungen, einen Einfluss auf den Weg haben können. So kann es sein, dass für einen bestimmten Nutzerkreis, wie Rollstuhlfahrer, Treppen nicht

zugänglich sind und für die Wegewahl vermieden werden müssen. Ähnliches gilt für Sicherheitsbereiche an Flughäfen, die nur für bestimmte Personenkreise zugänglich sind. Die Wegeführung wiederum erfordert eine geeignete Darstellung des Routingergebnisses, sei es in Form einer Karte oder in Form von textuellen Anweisungen.

Grundsätzlich lassen sich Anfragen in die Teilbereiche Positionsanfragen, Bereichsanfragen, Nächste-Nachbarn-Anfragen, Routinganfragen, Visualisierung und Bearbeitungsanfragen aufteilen.

- **Positionsanfragen** beinhalten die Umwandlung von Koordinaten verschiedener Art in ein anderes Referenzsystem.
- **Bereichsanfragen** beinhalten die Suche nach Objekten in einem bestimmten Bereich.
  - **Enthaltenseinstests** liefern alle Objekte in einem bestimmten Bereich.
  - **Zugehörigkeitstests** ergeben, zu welchem Bereich bestimmte Objekte gehören.
  - **Sichtbarkeitsanfragen** überprüfen mögliche Kollisionen auf der Sichtlinie zwischen zwei Punkten.
- **Nächste-Nachbarn-Anfragen** ermitteln zu einem Objekt oder einer Position die nächstgelegenen Objekte eines bestimmten Typs.
- **Routinganfragen** beinhalten das Routing zwischen verschiedenen Koordinaten, aber auch die Erzeugung von textuellen Navigationsanweisungen zur Darstellung der Route.
- Die Aufgabe der **Visualisierung** besteht vor allem aus der Übertragung von visuellen Objekten, die eine graphische Darstellung des Modells erlauben.
- **Bearbeitungsanfragen** beinhalten alle Arten von Änderungen am Modell.

Zur Erfüllung der Anforderungen und zur optimalen Beantwortung der Anfragen wird in [136] eine Datenstruktur vorgeschlagen, die als hybrid zwischen Raster- und Vektorbasierten Modellen angesehen werden kann und aus mehreren Blöcken besteht. Im ersten Block werden globale topologische Informationen in Form von mehreren Graphen modelliert. Der zweite Block unterstützt verschiedene Visualisierungen, während ein dritter Block lokale topologische Informationen in Form von mit speziellen Farben gekennzeichneten Bitmaps verwaltet. Der letzte Block ist für die semantischen Informationen zuständig. Jeder der Blöcke wird nun detaillierter vorgestellt.

Die globale Topologie entspricht den Verbindungen zwischen Räumen und Routen. Hier können mehrere Graphen zu unterschiedlichen Zwecken verwaltet werden, wobei jeder Knoten in den Navigationsraum eingebettet ist und jede Kante sowohl ein Gewicht entsprechend der Länge, als auch der dafür benötigten Zeit, inklusive benötigter Wartezeiten, hat. Beispiele für verschiedene vorberechnete Graphen werden im Folgenden genannt:

- Besonders wichtig sind Zugangsgraphen, die den Zugang zu Freiflächen in einer bestimmten Diskretisierung modellieren und für die Berechnung detaillierter Routen genutzt werden.

- Um potentielle Ziele zu modellieren und Wege zwischen diesen effizient zu berechnen, gibt es einen POI-Verbindungsgraphen, der aus dem Zugangsgraphen erzeugt werden kann und für jede Verbindung zwischen POIs das Gesamtgewicht der Kanten auf dem kürzesten Weg beinhaltet.

- Ein Stockwerksverbindungsgraph erlaubt die Anfrage nach Verbindungen über Stockwerke hinweg. Jede Kante verweist auf die Art der Verbindung und kommt auch im Zugangsgraphen vor. Somit lässt sich effizient eine Einschränkung der Wegewahl auf bestimmte Verbindungen, wie Aufzüge, erreichen.

- Ein reduzierter Navigationsgraph, der beispielsweise nur Abzweigungen beinhaltet, eignet sich besonders gut zur Erzeugung textueller Navigationsanweisungen.

Der Block zur Visualisierung unterstützt ebenfalls mehrere Datenstrukturen, die je nach Bedarf und Anfrage gewechselt werden können. Eine Datenstruktur entspricht einem 2,5D oder dreidimensionalen Vektor-basierten Modell, dass die Visualisierung aus Linien erzeugt, während ein anderes Modell aus einer einfachen Bitmap je Ebene besteht.

Die lokale Topologie besteht im Gegenzug zur Globalen aus mehreren Bitmaps. Diese weisen zusätzliche Parameter auf, um eine Transformation aus und in andere Modelle aus den vorhergehenden Blöcken zu erlauben. Objekte mit spezieller Farbe entsprechen Landmarken zur Erzeugung textueller Navigationsanweisungen oder POIs als Navigationsziel. Die Semantik der einzelnen Farben wird in Form von Farbtabellen verwaltet. Zudem ist es möglich, solche Linien auf der Bitmap durch eine spezielle Markerfarbe zu überlagern, welche keinem Hindernis entsprechen. Die Beschreibung der lokalen Topologie orientiert sich an dem Bitmap-basierten Gebäudemodell aus [137], wodurch gerade Kollisions- oder Sichtlinienanfragen, wie auch Enthaltenseinsanfragen sehr effizient beantwortet werden können.

Semantische Informationen, wie Raumnamen oder Objektbezeichnungen, werden mit der geometrischen Darstellung des Gebäudes verknüpft. Das bedeutet, dass hier die Zuordnungen zwischen Raumnamen, Knoten im Graphen, Gebäudegeometrie und visueller Darstellung, beispielsweise in Form eines Piktogramms, erreicht wird.

Ein großer Vorteil dieser Aufteilung liegt darin, dass für jede Art von Anfrage die Daten für eine effiziente Beantwortung bereits explizit modelliert sind. Als Beispiel soll die Sichtlinienüberprüfung genannt werden. In einem als Quadtree organisierten Vektor-basiertem Modell und in dem hybriden Modell wurden 100.000 zufällige Sichtlinienanfragen gestellt und die Antwortzeiten gemessen. Dabei wurde beim hybriden Modell die Anfrage durch das lokale topologische Modell in Form einer Bitmap mit Pixel-Größe von 0.15 m x 0.15 m beantwortet. Die Antwortzeit beim hybriden Modell war rund 30 mal schneller als beim reinen Vektor-basierten Modell, wobei allerdings alleine der Speicherplatz der Bitmap ungefähr 60 mal dem Speicherplatz der Kollisionslinien im Vektor-basierten Modell entsprach.

3.4.2 Ebenen des hybriden Umgebungsmodells

Ausgehend von den Ergebnissen aus [136] wurde ein Umgebungsmodell definiert, welches die verschiedensten Anfragen auf dem hybriden Modell effizient beantworten kann und dazu eine Kombination aus dem BIGML und dem Bitmap-basierten Umgebungsmodell verwendet. Das Modell lässt sich in verschiedene Ebenen der Wissensrepräsentation einteilen, die teilweise der Vektor-basierten Darstellung und teilweise der Raster-basierten Darstellung zuzuordnen sind. Zudem gibt es noch eine symbolische Ebene, in der menschenlesbare Bezeichner den räumlichen Strukturen zugewiesen werden.

Vektorebenen

Eine Vektorebene des hybriden Umgebungsmodells, die **BIGML-Ebene**, besteht aus der Geometrie des BIGML-Modells. Dieses beinhaltet für alle in BIGML vorkommenden räumlichen Objekte deren genaue geometrische Informationen und eignet sich somit unter anderem für die Funktion der visuellen Darstellung. Dies ist bei Räumen, Stockwerken oder Gebäuden beispielsweise der Umriss, bei Treppen der Bereich, der auf einem Stockwerk liegt, bei Türen die von der geschlossenen Tür eingenommene Fläche. Zudem existiert ein **Graphebene** mit einem Wegegraphen, dessen Knoten und Kanten in die Vektorebene eingebettet sind und Informationen über die begehbaren Flächen im Gebäude beinhalten. Zudem verweisen manche der Knoten auf spezielle Objekttypen, wie Portale eines bestimmten Typs oder POIs. Die Graphebene vereint somit die globalen topologischen Informationen, ohne einen POI-Verbindungsgraphen oder einen Stockwerkverbindungsgraphen explizit anzugeben. Diese lassen sich bei Bedarf jedoch schnell aus dem angegebenen Graphen in Zusammenhang mit den genannten Objekttypen berechnen.

Rasterebenen

Die Rasterebenen bestehen aus mehreren farbigen Bitmaps, die aus Effizienzgründen zusätzlich bestimmte Informationen beinhalten und aus dem BIGML-Modell erzeugt werden. Zum einen gibt es die **Pixelrasterebene**, welche die lokalen topologischen Informationen abbildet und aus einem Pixelraster besteht, welches begehbare Flächen durch weiße Pixel, Kollisionsgeometrie durch schwarze Pixel und Portale durch Pixel einer bestimmten von der Art des Portals abhängigen Farbe darstellt. Diese Ebene dient vor allem der Kollisionsabfrage, erlaubt jedoch zudem ein Routing auf Pixelebene. Eine weitere Rasterebene beinhaltet die farbliche Kodierung der Raumzugehörigkeit einzelner Pixel und wird im Folgenden als **Raumidentifizierungsebene** bezeichnet. Dazu wird jeder Raum aus der Pixelrasterebene durch eine 4-Nachbarn-Flutung in einer eindeutigen Farbe gekennzeichnet und diese dem entsprechenden symbolischen Bezeichner in einer der Raumidentifizierungsebene beigefügten Farbtabelle zugeordnet. Diese enthält für jede Farbe einen Verweis auf das zugehörige Objekt der symbolischen Ebene. Analog zur Raumidentifizierungsebene wird eine **AOI-Identifizierungsebene** erzeugt, welche die Zuordnung einzelner Pixel zu einer AOI beinhaltet. Überschneiden sich mehrere AOIs, so wird neben dem Farbraum zusätzlich der Alpha-Kanal genutzt. Ist dieser ungleich null, so verweist der Eintrag auf eine separate

Tabelle, in welcher bei 8 Bit Farbtiefe bis zu 256 Kombinationen von Überschneidungen aufgelöst werden können. Alternativ kann man auch Überschneidungen von verschiedenen AOIs als separate Farbe speichern, wodurch bei Verwendung des Alpha-Kanals mehr als 4 Milliarden Kombinationen von AOIs kodiert werden können.

Symbolische Ebene

Die **symbolische Ebene** beinhaltet alle Namen und Bezeichner von Objekten im BIGML-Modell. Dies reicht von Gebäude-, Stockwerks- und Raumnamen über Bezeichner von POIs und AOIs bis hin zu Identifikationsnummern von einzelnen Portalen. Jeder dieser Bezeichner verfügt zudem über eine Zuordnung zu einer geometrischen Darstellung auf der BIGML-Ebene und im Fall von POIs oder Portalen zudem über einen Verweis auf den zugehörigen Knoten in der Graphebene. Zudem gibt es für jedes Objekt räumlicher Ausdehnung einen Verweis auf die zugehörige Farbtabelle der entsprechenden Identifizierungsebene. Damit erfüllt diese Ebene die Anforderung nach der Darstellung semantischer Informationen und deren Verknüpfung mit den restlichen Ebenen.

3.4.3 Anfragebearbeitung

Die Anfragebearbeitung gliedert sich in die bereits genannten sechs verschiedenen Anfragetypen: Positionsanfragen, Bereichsanfragen, Nächste-Nachbarn-Anfragen, Routinganfragen, Visualisierung und Bearbeitungsanfragen. Dazu wird jeweils auf die entsprechenden Komponenten im System eingegangen, welche zur Beantwortung der Anfrage genutzt werden.

Positionsanfragen

Positionsanfragen beinhalten eine Koordinate beliebiger Art, die beispielsweise von einem Positionsbestimmungssystem erzeugt wurde und somit symbolisch oder geometrisch sowie deterministisch oder probabilistisch sein kann. Zudem kann eine bestimmte Repräsentation als erwartete Antwort angegeben werden. Ist dies der Fall, so beinhaltet die Antwort die Position in der geforderten Repräsentation mit optionalen weiteren Informationen, wie der Begehbarkeit des Modells am Ort der Position. Ist die geforderte Repräsentation nicht weiter spezifiziert, so wird eine Standard-Repräsentation des Modells gewählt, die üblicherweise einem oder mehreren symbolischen Bezeichnern mit einer zugehörigen Wahrscheinlichkeitsangabe entspricht. Der Grund für dieses Vorgehen liegt darin, dass die meisten Systeme zur Positionsbestimmung mit geometrischen Koordinaten arbeiten, der Nutzer jedoch symbolische Koordinaten als verständlicher empfindet.

Grundsätzlich unterscheidet das hybride Umgebungsmodell mindestens vier verschiedene Koordinatensysteme, namentlich das symbolische System der menschenverständlichen Bezeichner, das lokale metrische Koordinatensystem des Vektor-basierten Modells, das lokale metrische Koordinatensystem des Bitmap-basierten Modells, wobei die Koordinaten

den Pixeln auf dem Bild entsprechen, und ein globales Koordinatensystem, wie WGS84. Abbildung 3.5 zeigt die Transformationsmöglichkeiten zwischen den einzelnen Modellen.

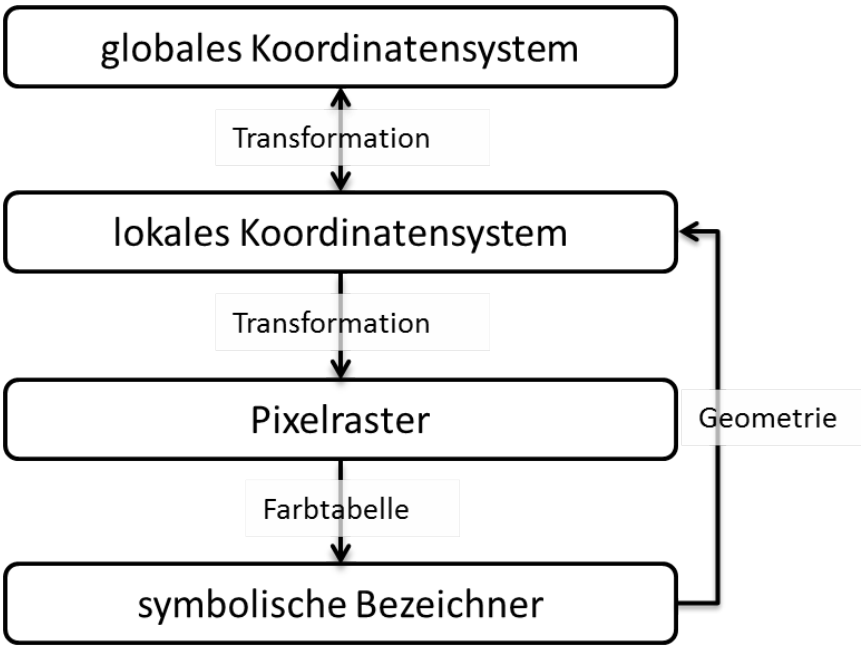


Abbildung 3.5: Transformationsmöglichkeiten zwischen den verschiedenen Koordinatensystemen des hybriden Umgebungsmodells.

Die Transformation einer globalen Koordinate kann mittels einer Koordinatentransformation in eine lokale Koordinate des Vektor-basierten Modells erfolgen, dazu erfolgt üblicherweise eine Transformation zwischen dem globalen Ellipsoiden Koordinatensystem und einem lokalen Kartesischen Koordinatensystem. Die genaue Funktionsweise kann beispielsweise in [53, Seite 278 ff.] nachgelesen werden. In den meisten Fällen ist die Ausdehnung des Gebäudes im Vergleich zur Erdoberfläche jedoch relativ gering, wodurch vereinfacht davon ausgegangen werden kann, dass der vom Gebäude überdeckte Teil der Erdoberfläche in einer Ebene liegt. In diesem Fall kann man die Transformation durch eine einfachere Helmert-Transformation zwischen Kartesischen Koordinatensystemen abbilden (siehe Gleichung 3.1):

$$\begin{pmatrix} x_l \\ y_l \end{pmatrix} = \begin{pmatrix} T_x \\ T_y \end{pmatrix} + \begin{pmatrix} cos(\alpha) & -sin(\alpha) \\ sin(\alpha) & cos(\alpha) \end{pmatrix} \times \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x_g \\ y_g \end{pmatrix} \tag{3.1}$$

Dabei ist $(x_l, y_l)^T$ der Ortsvektor im lokalen Koordinatensystem, $(x_g, y_g)^T$ selbiger im globalen Koordinatensystem, T die Verschiebung der Ursprünge, α die Drehung und S der Skalierungsfaktor der Modelle zueinander. Als Approximation lässt sich hier für die globale Koordinate auch direkt der Längen- und Breitengrad angeben.

Die Transformation zwischen dem lokalen Koordinatensystem und dem Pixelraster erfolgt ebenfalls mit einer Helmert-Transformation, wobei in diesem Fall die Rotationsmatrix

aus Gleichung (3.1) der Einheitsmatrix entspricht und der Skalierungsfaktor bezüglich beider Achsen gleich ist. Problematisch ist hierbei, dass die Abbildung nicht bijektiv ist, da unendlich viele Kartesische Koordinaten auf jeweils ein und dieselbe Pixelkoordinate abgebildet werden. Aus diesem Grund wird im Modell von einer direkten Rücktransformation abgesehen, wobei diese bei der Umsetzung der Anfragen auf Modellseite auch nicht benötigt wird.

Um von einer Pixelkoordinate auf einen symbolischen Bezeichner räumlicher Ausdehnung zu schließen wird die zugehörige Identifizierungsebene und deren Farbtabelle für die Zuordnung von Pixeln zu Räumen oder Areas of Interest genutzt. Dies ist deutlich effizienter als die geometrische Berechnung eines Point-in-Polygon Tests für alle möglichen symbolischen Bezeichner. Handelt es sich um einen symbolischen Bezeichner, der durch einen Punkt repräsentiert wird, also beispielsweise einen Point of Interest, so wird dessen Koordinate in eine Pixelkoordinate umgewandelt und anschließend ein Vergleich der Koordinaten durchgeführt.

Ein symbolischer Bezeichner kann in jedes andere System dadurch übertragen werden, dass die explizit dazu angegebene Geometrie ausgelesen wird, die im lokalen Koordinatensystem angegeben ist. Ausgehend von dieser Angabe lässt sich ein Bezeichner auch in alle anderen Systeme übertragen. Dasselbe gilt für die Transformation von globalen Koordinaten in symbolische oder Pixelkoordinaten.

Bisher wurde für die Zwecke der Transformationen von deterministischen Koordinaten ausgegangen. Die Verarbeitung von probabilistischen Koordinaten ist dabei jedoch für viele Anwendungsfälle zwingend notwendig, da viele Positionsbestimmungssysteme aufgrund der Messungenauigkeit der Sensoren nur fehlerbehaftete Positionsdaten liefern. Kann eine Aussage über den Fehler getroffen werden, so wird die Position üblicherweise in Form einer Verteilung angegeben. Dabei sind diskrete Verteilungen, beispielsweise in Form von Partikelwolken verbreitet, aber auch Konfidenzellipsen von zweidimensionalen Gleich- oder Normalverteilungen sind üblich. Im Falle der Partikel lässt sich eine Transformation zwar generell auf alle Partikel einzeln anwenden, jedoch führt dies bei hoher Partikeldichte zu möglicherweise ungewollter Prozessorlast. Hier schafft das Clustern der Partikel und die Transformation der Clusterschwerpunkte Abhilfe. Konfidenzellipsen lassen sich dadurch abbilden, dass die Ellipsenparameter transformiert werden und so eine Ellipse im Zielkoordinatensystem vorliegt.

Ist die Position in Form einer probabilistischen Koordinate gegeben, so lässt sich meist kein eindeutiger symbolischer Bezeichner als Ergebnis zurückgeben, da meist mehrere symbolische Bezeichner eine positive Wahrscheinlichkeit aufweisen. In diesem Fall wird eine Liste der möglichen symbolischen Bezeichner - geordnet nach der jeweiligen Wahrscheinlichkeit - zurück gegeben, wobei ein Grenzwert für eine Mindestwahrscheinlichkeit angegeben werden kann. Die Wahrscheinlichkeit berechnet sich im Fall einer diskreten Wahrscheinlichkeitsverteilung über die Summe der Gewichte der einzelnen Partikel je Bezeichner, im Fall von kontinuierlichen Verteilungen durch die Integration über das geometrische Ausmaß des symbolischen Bezeichners. Dies lässt sich im Fall einer Gleichverteilung durch die prozentuale Schnittfläche der Ellipse mit den einzelnen Räumen vereinfachen.

Bereichsanfragen

Bereichsanfragen beinhalten einen geometrischen Bereich oder symbolischen Bezeichner und einen oder mehrere BIGML-Typangaben oder geometrische Koordinaten der Zielobjekte. Als Antwort werden alle Objekte des angeforderten Typs im angegebenen Bereich oder eine Liste der bei der Anfrage angegebenen Koordinaten, die im Bereich enthalten sind, zurückgegeben. Dabei unterscheidet sich die Bearbeitung je Parametern der Anfrage und Objekttyp.

Entspricht der gesuchte Objekttyp einem BuildingInformation-Element aus BIGML und wurde ein symbolischer Bezeichner als Bereich angegeben, so kann die Bereichsanfrage direkt auf der hierarchischen Repräsentation des Grundstücks im BIGML-Modell bearbeitet werden. Ein solcher Fall tritt beispielsweise ein, wenn alle Räume eines Gebäudes oder Stockwerks gesucht werden, aber auch wenn POIs eines bestimmten Typs innerhalb eines symbolischen Bezeichners gesucht sind.

Ist allerdings auch nur einer der Anfrageparameter als geometrische Form angegeben, so muss die Bereichsanfrage rechnerisch gelöst werden. Üblicherweise erfolgt in diesem Fall eine aufwändige Auswertung von geometrischen Beziehungen, die bei einer großen Anzahl an Objekten schnell sehr komplex werden kann. In manchen Fällen lässt sich die Anfrage jedoch deutlich schneller und effizienter durch Algorithmen der Bildbearbeitung auf der Bitmap-Ebene lösen. Im Folgenden werden die verschiedenen Fälle vorgestellt.

Werden geometrische Koordinaten in einem symbolischen Bezeichner gesucht, so wird die geometrische Koordinate in eine Pixelkoordinate umgerechnet und anschließend in der Farbtabelle der zum symbolischen Bezeichner passenden Indentifizierungsebene eine Übereinstimmung überprüft. Damit lässt sich mit effizienter Indexierung die Komplexität auf $O(log(n))$ reduzieren, wobei n hier die Anzahl der symbolischen Bezeichner in der selben Ebene ist, wie die geometrische Koordinate, und die Komplexität einem einmaligen Aufruf der Farbtabelle entspricht. Dies ist in den meisten Fällen deutlich schneller, als ein geometrischer Punkt-in-Polygon-Test für jede Koordinate mit dem Polygon, das die geometrische Form des Bezeichners im BIGML-Modell darstellt.

Werden ein geometrischer Bereich und geometrische Koordinaten übergeben, so kann das Vorgehen auf den vorangehenden Fall abgebildet werden. Dazu wird der geometrische Bereich in einer temporären Kopie der Rasterebene des Umgebungsmodells in einer bestimmten Farbe eingefärbt, was zwar Speicherplatz, aber kaum Prozessorkapazität erfordert. Anschließend kann die Farbe jeder Pixeldarstellung der geometrischen Koordinate in konstanter Zeit überprüft werden. Falls sie der vorher gewählten Farbe entspricht, so befindet sie sich im geometrischen Bereich.

Der letzte Fall tritt genau dann ein, wenn die Anfrage einen geometrischen Bereich und Objekttypen beinhaltet. In diesem Fall muss zu jedem Objekttyp die geometrische Koordinate aller Objekte dieses Typs bestimmt werden und die Anfrage anschließend auf den vorangehenden Fall nur geometrischer Koordinaten abgebildet werden. Hierbei ist zu erwähnen, dass durch diese Anfrage auch das Enthaltensein von zweidimensionalen Objekten in einer anderen Fläche überprüft werden kann, beispielsweise durch einen Aufruf mit Objekttypen einer zweidimensionalen Geometrie. Dieser Fall wird auf die aus bestehenden

Geoinformationssystemen bekannte geometrische Berechnungen abgebildet (vgl. [73]).

Wie die Evaluation in Abschnitt 3.4.4 zeigt, kann durch die Abbildung geometrischer Operationen auf Mechanismen der Bildbearbeitung die Beantwortung von Anfragen effizient gestaltet werden. Da einer Bitmap für die Darstellung eines Pixels üblicherweise 255 Bits je Grundfarbe Rot, Grün und Blau und zudem noch einmal 255 Bit für die Transparenz (auch als Alpha bezeichnet) zur Verfügung stehen, können auf diese Weise mehr als 4 Milliarden symbolische Bezeichner kodiert werden. Wählt man den Alpha-Kanal jedoch dazu aus, um die Überlagerung von AOIs im selben Gebiet zu kodieren, so stehen immer noch mehr als 16 Millionen verschiedene AOIs je Umgebungsmodell zur Verfügung, was auch für große komplexe Gebäude ausreicht.

Zu der oben bereits genannten Bereichsanfrage gibt es noch die Anfrage zur Begehbarkeit eines Bereiches oder eines bestimmten Punktes. Dies kann vor allem für Positionsbestimmungssysteme oder Navigationsanfragen wichtig sein. Zu diesem Zweck sind begehbare Flächen in der Pixelrasterebene in weiß gekennzeichnet, um effizient über einen einzelnen Zugriff auf das Modell die Begehbarkeit eines Punktes zu überprüfen. Um Portale als spezielle begehbare Bereiche zu kennzeichnen, die im BIGML-Modell durch den geometrischen Bereich gegeben und durch Linien gekennzeichnet sind, wurde für jeden Portaltyp eine spezielle Farbe gewählt. Diese Farbe charakterisiert den Portaltyp und markiert den vom Portal eingenommenen Bereich auf der Pixelrasterebene in der entsprechenden Farbe. Wird beim Begehbarkeitstest eines Punktes eine der Portal-Farben oder Weiß zurückgegeben, so war der Test erfolgreich. Wird ein ganzer Bereich abgefragt, so wird der begehbare Anteil daran prozentual als Ergebnis zurückgegeben. Dies wird ebenfalls auf Basis der Farbgebung jedes Pixels im Bereich überprüft.

Eine der wichtigsten Begehbarkeitsanfragen ist jedoch die Kollisionsanfrage, die ebenfalls zu den Bereichsanfragen gezählt werden kann. Hier wird für zwei geometrische Koordinaten die Sichtbarkeit untereinander geprüft, wobei als Ergebnis wahr oder falsch zurückgegeben wird. Dazu wird auf der Pixelrasterebene Bresenhams Algorithmus zum Zeichnen einer Linie verwendet, wobei anstelle des Zeichnens jeder getroffene Pixel auf seine Farbe überprüft wird. Ist dieser schwarz, so kam es zu einer Kollision, und der Test war nicht erfolgreich. Genauso lässt sich eine Kollisionsanfrage auf BIGML-Ebene beantworten, allerdings muss in diesem Fall ein Test auf Überschneidung mit allen Linien desselben Stockwerks durchgeführt werden. Am effizientesten ist hierbei eine hybride Methode, bei der zuerst der zugehörige Raum auf der Raumidentifizierungsebene gesucht wird und dessen Geometrie auf Schnitt mit der Verbindunglinie der beiden Koordinaten untersucht wird. Allerdings müssen in diesem Fall bei einem Schnitt zusätzlich überprüft werden, ob es sich um begehbare Portalgeometrie handelt und die Kollision möglicherweise erlaubt ist.

Nächste-Nachbarn-Anfragen

Nächste-Nachbarn-Anfragen beinhalten eine geometrische oder symbolische Koordinate oder ein bestimmtes BIGML-Objekt als Ziel, zu dem die nächsten Nachbarn bestimmt werden sollen. Zusätzlich wird üblicherweise ein Objekttyp angegeben, der den Typ der gesuchten Nachbarobjekte angibt, aber auch ein Aufruf mit einer Menge an geometri-

schen oder symbolischen Koordinaten ist möglich. Zudem kann angegeben werden, wie viele Nachbarn zurückgegeben werden sollen, welcher maximale Abstand bestehen soll und bezüglich welcher Distanzberechnung die Nähe bestimmt werden soll. Hier stehen die Wegstrecke oder die Euklidische Distanz zur Verfügung. Die Antwort enthält eine Liste der Objekte, die vom angegebenen Typ sind und eine niedriger Distanz haben, als der maximal angegebene Abstand zum Ziel. Wurden anstelle des Objekttyps geometrische oder symbolische Koordinaten in der Anfrage verwendet, dann wird die gesuchte Zahl dieser Koordinaten, die einen geringeren Abstand als den Maximalabstand haben, zusammen mit der jeweiligen Distanz als geordnete Liste zurückgegeben.

An dieser Stelle soll zuerst der Abstand bezüglich Wegstrecke und Euklidischer Distanz zwischen Punkten und Flächen definiert werden, bevor auf die Beantwortung der Anfrage eingegangen wird. Die Euklidische Distanz zwischen Punkten ist wohldefiniert, der Abstand zwischen zwei Flächen wird als minimaler Abstand zweier Punkte dieser Flächen definiert. Entsprechend ist die Euklidische Distanz zwischen einem Punkt und einer Fläche der minimale Abstand zwischen dem Punkt und einem Punkt der Fläche. Die Wegstrecke zwischen zwei Flächen entspricht der kürzesten Wegstrecke zwischen zwei Wegpunkten, die sich innerhalb der Flächen befinden. Die Wegstrecke zwischen zwei Punkten lässt sich hingegen in den meisten Fällen nicht allein auf Basis des Wegegraphen berechnen, da die meisten Punkte nicht mit Knoten des Graphen übereinstimmen. Hierzu muss ein zweigeteiltes Routing auf dem Wegegraph und der Pixelrasterebene durchgeführt werden, welches im nachfolgenden Abschnitt 3.4.3 im Detail vorgestellt wird.

Wird die nächste-Nachbarn-Anfrage bezüglich der Wegstrecke mit einer symbolischen Koordinate eines Punktobjektes durchgeführt, beispielsweise eines POIs oder Wegpunktes, so kann die Anfrage direkt auf der Graphebene des Vektor-basierten Modells durchgeführt werden. Sind die gesuchten Nachbarn in Form von Objekttypen angegeben, so werden die Objekte des BIGML-Modells nach dem gesuchten Typ, der auch in Form eines Tags oder eines Attributs vorliegen kann, gefiltert. Dies geschieht dadurch, dass ein Dijkstra-Algorithmus mit dem Punktobjekt gestartet und bei jedem erreichten Knoten ein Test auf den Objekttyp durchgeführt wird. Da die Knoten in der Reihenfolge der Entfernung zum Startpunkt abgearbeitet werden, kann der Algorithmus beendet werden, wenn der als nächstes betrachtete Wegpunkt entweder eine größere Entfernung aufweist, als bei der Anfrage angegeben wurde oder bereits die angegebene Anzahl an nächsten Nachbarn gefunden wurde.

Auf diesen Fall lassen sich alle anderen Fälle abbilden. Sind die Nachbarn in Form von geometrischen Koordinaten angegeben, so wird für jeden Nachbarn mithilfe der Bereichsanfrage auf Raumidentifizierungsebene zunächst der umschließende Raum ermittelt. Anschließend wird analog zum vorangehenden Fall mithilfe von Dijkstra die Entfernungstabelle zu den Räumen aufgebaut. Entspricht ein Raum den ermittelten umschließenden Räumen, so wird anschließend auf Pixelrasterebene der Weg vom ermittelten ersten Wegpunkt im Raum zur geometrischen Koordinate des Nachbarn fortgesetzt und das Ergebnis entsprechend Dijkstra als offen gespeichert und aktualisiert, sobald ein kürzerer Weg zum selben Punkt zur Verfügung steht.

Ist das Ziel ebenfalls in Form einer geometrischen Koordinate angegeben, so wird das

Verfahren auf jeden Türknoten des das Ziel umschließenden Raumes gestartet und zu jedem Nachbarn die jeweils kürzeste zusammengesetzte Route zurückgegeben, die aus der Route zum Türknoten und der Route von der Tür zum Nachbarn besteht. Im Falle einer Anfrage mit einer Fläche wird zunächst eine Menge an Räumen ermittelt, welche die Fläche schneiden und das Verfahren anschließend auf jeden Raum einzeln angewendet.

Routinganfragen

Routinganfragen stellen eine wichtige Anfrageart dar, die nicht nur zu Navigationszwecken, sondern auch für Nächste-Nachbarn-Anfragen, Zeit- und Entfernungsschätzungen genutzt wird. Als Eingang dient eine partiell geordnete Liste von Koordinaten, wobei die erste Koordinate der Startpunkt ist. Zudem wird angegeben, ob die kürzeste oder die schnellste Route gesucht ist und ob Einschränkungen bestehen, wie die Forderung nach behindertengerechtem Zugang. Als Antwort wird neben der Zeit oder Entfernung eine vollständig geordnete Liste zurückgegeben, welche die partielle Ordnung erhält und beim sequentiellen Ablaufen der Koordinaten die kürzeste Strecke unter Einbeziehung aller Ziele darstellt. In dieser Arbeit liegt der Fokus der Anfragebearbeitung im Vordergrund, weswegen an dieser Stelle nur auf die Lösung des Routings zwischen zwei Koordinaten eingegangen wird. Für die Lösung des partiell geordneten Traveling-Salesman-Problems sei auf die Arbeit von Werner [135] verwiesen. In Abhängigkeit von den Koordinaten, mit denen die Routinganfrage durchgeführt wird, werden verschiedene Algorithmen ausgeführt. Es werden die folgenden Fälle vorgestellt, wobei auch Mischformen auftreten können, welche die Mechanismen entsprechend kombinieren: Routing zwischen Wegpunkten, zwischen symbolischen Bezeichnern, zwischen geometrischen Koordinaten und zwischen probabilistischen geometrischen Koordinaten.

Beim **Routing zwischen Wegpunkten** wird die kürzeste Route zwischen zwei Wegpunkten des BIGML-Wegegraphen berechnet. Dazu wird der Dijkstra-Algorithmus auf den Kantenlängen eingesetzt, wobei je nach gewünschtem Ergebnis die Entfernung oder die benötigte Zeit als Kantenlänge verwendet wird. Aufgrund der großen Unterschiede zwischen topologischer und Euklidischer Entfernung bei Stockwerksunterschieden ist die Verwendung des A*-Algorithmus zum Routing nicht zu empfehlen. Grundsätzlich lässt sich auch auf der Pixelrasterebene ein Routing zwischen Wegpunkten realisieren, jedoch steigt die Komplexität aufgrund der größeren Knotenzahl deutlich, und das Ergebnis erfordert möglicherweise eine Nachbearbeitung, um begehbare Wege anzugeben (vergleiche [137]). Alle im Folgenden beschriebenen Routingfälle lassen sich auf das Routing zwischen zwei Wegpunkten abbilden, da in allen Fällen zuerst von der Startkoordinate zu einem nahegelegenen Wegpunkt geroutet wird, von dort zu einem Wegpunkt, der nahe zur Zielkoordinate liegt, und anschließend von dort weiter zur Zielkoordinate.

Das **Routing zwischen symbolischen Bezeichnern** entspricht entweder dem Routing zwischen Wegpunkten, da alle symbolischen Bezeichner mit einer Punktkoordinate, wie POIs oder Portale, direkt auf einen Wegpunkt verweisen, oder dem Routing zwischen Flächen. Im letzterem Fall wird die kürzeste Strecke zwischen zwei Wegpunkten der beiden Flächen zurückgegeben. Sind die symbolischen Bezeichner Räume, so reicht es, alle

Kombinationen von Portalknoten in den Räumen auszuwerten, da diese die ersten Knoten in jedem Raum sind und so die alle anderen inneren Knoten in diesem Fall ignoriert werden können. Sind Ebenen gegeben, so ist sogar die Einschränkung auf Portale, die einen Stockwerkswechsel zulassen, möglich. Im Fall von Gebäuden müssen nur noch die *BuildingPortals* überprüft werden.

Zum **Routing zwischen geometrischen Koordinaten** werden die Koordinaten zunächst in die Pixelrasterebene transformiert, und anschließend wird ein dreistufiges Routing durchgeführt. Die Suche nach dem jeweils kürzesten Weg von der geometrischen Koordinate im Start- und Zielraum zu jedem Portalknoten im selben Raum und dem kürzesten Weg zwischen den Portalknoten in beiden Räumen. Die Kombination mit der kürzesten Gesamtwegstrecke wird als Ergebnis zurückgegeben. Mithilfe der Raumidentifizierungsebene werden zunächst Start- und Zielraum identifiziert. Anschließend wird auf Pixelrasterebene mit dem A*-Algorithmus die kürzeste Route zu jedem Portalknoten im Raum ermittelt. Im Gegensatz zum Routing auf Wegpunkteben lässt sich innerhalb eines Raumes mithilfe der Euklidischen Distanz eine valide Metrik für den A*-Algorithmus angeben und so eine schnellere Antwort als bei Dijkstra erzielen. Zwischen den identifizierten Wegpunkten wird anschließend die kürzeste Strecke ermittelt. Die Summe der Strecke der geometrischen Koordinaten zum jeweiligen Portalknoten und der Strecke zwischen diesen ergibt die Gesamtlänge des Weges. Aus allen möglichen Kombinationen wird nun die kürzeste Stecke als Ergebnis zurückgegeben. Die Suche nach der schnellsten Strecke funktioniert analog.

Des Weiteren wird der Fall des **Routings zwischen einer probabilistischen Koordinate und einem Wegpunkt** betrachtet, der beispielsweise dann eintritt, wenn ein System zur Positionsbestimmung eine probabilistische Koordinate als Ergebnis zurückgibt und dennoch die Navigation zu einem POI durchgeführt werden soll. In diesem Fall ist der Einbezug der Unsicherheit in das Routing sinnvoll. Wird nur die wahrscheinlichste Route zurückgegeben, ist der Nutzer im Fehlerfall vollständig unwissend, während bei Angabe mehrerer wahrscheinlicher Wege eine dedizierte Entscheidung auf Nutzerebene möglich ist. Dementsprechend wird je nach Art der probabilistischen Koordinate die Aufenthaltswahrscheinlichkeit für jeden Raum bestimmt. Im Falle einer Gleich- oder Normalverteilung geschieht dies durch die Integration über die Fläche des Raums, bei einer Partikelwolke durch das Aufsummieren des Gewichts jedes einzelnen Partikels pro Raum. Die Auflösung Partikel zu Raum geschieht wieder auf der Raumidentifikationsebene. Anschließend wird der Schwerpunkt der Wahrscheinlichkeitsverteilung je Raum bestimmt und dieser Punkt als Startpunkt eines Routings von einer geometrischen Koordinate zu einem Wegpunkt hergenommen. So wird schlussendlich für jeden Raum mit positiver Aufenthaltswahrscheinlichkeit eine Route zum Ziel erzeugt. Die Liste der Routen wird nach Wahrscheinlichkeit absteigend sortiert zurückgegeben, wobei sowohl eine maximale Anzahl an alternativen Routen, wie auch eine minimal geforderte Wahrscheinlichkeit einzelner Routen die Ergebnisliste beschränken kann. Ein Beispiel für probabilistisches Routing ist in Abbildung 3.6 dargestellt.

Zuletzt wird noch die **Generierung von Weganweisungen** behandelt, die zu den Routinganfragen gezählt werden. Hierzu kommt oftmals das Eight-Corner-System zum Einsatz, das eine Weganweisung entsprechend der Winkels zwischen aufeinanderfolgenden

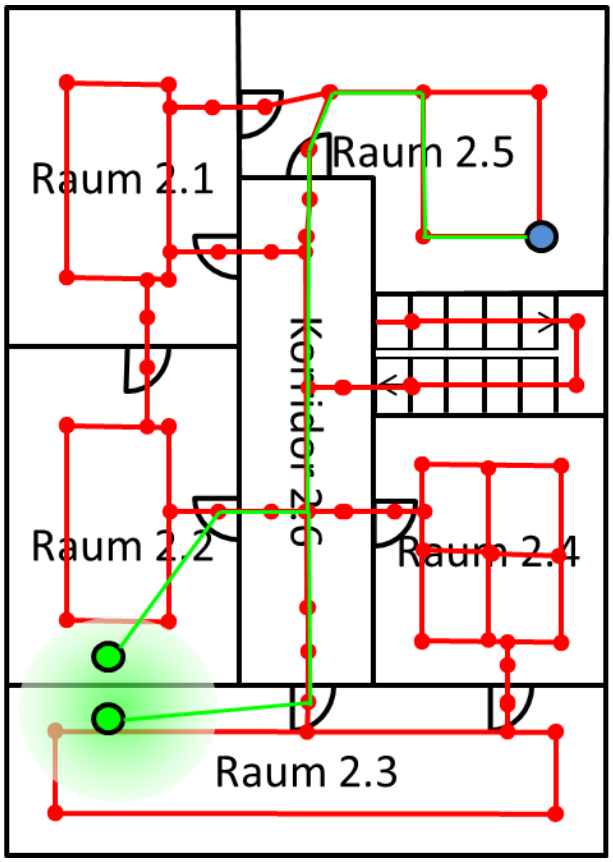


Abbildung 3.6: Probabilistisches Routing auf Basis einer normalverteilten Positionsschätzung mit zwei Clustern (hellgründe Punkte) zu einem Ziel (blauer Punkt). Es werden für jeden möglichen Aufenthaltsort alternative Routen (grüne Linien) im Wegenetz (rote Linien) angezeigt.

Streckenabschnitten erzeugt. Dabei werden acht verschiedene Anweisungen unterschieden, die von *geradeaus* über *schräg rechts*, *rechts* und *scharf rechts* bis zu *zurück* bestehen. Jeder Anweisung ist dabei ein Winkel von 45° zugewiesen. Eine detaillierte Diskussion zur Generierung von Weganweisungen auf Vektor-basierten Modellen findet sich in [112]. Auf Basis der Bitmap ist es ebenso denkbar Navigationsanweisungen zu erzeugen, aufgrund der 8-Nachbarn-Struktur einzelner Pixel eignet sich diese Darstellung sogar besonders gut für das 8-Corner-System. Um jedoch nicht zu viele Navigationsanweisungen zu erzeugen, wird in [137] ein radialer Sweepline-Algorithmus vorgestellt, mithilfe dessen das nächste nicht mehr sichtbare Pixel auf der Route und damit eine Navigationsanweisung ausgehend von der aktuellen Position berechnet werden kann.

Anfragen bezüglich Visualisierung

Auf Anfragen bezüglich der Visualisierung wird üblicherweise ein Teil des Modells angefragt, beispielsweise um eine Kartenansicht auf einem mobilen Endgerät zu ermöglichen.

Dazu steht dem hybriden Umgebungsmodell sowohl die BIGML-Ebene mit der Vektorbasierten Darstellung der Objekte, als auch die Pixelrasterebene zur Verfügung. In diesem Fall kann die Plattform die Entscheidung, welche Repräsentation übertragen werden soll, dem Empfänger überlassen werden. Benötigt beispielsweise ein System zur Positionsbestimmung eine Karte zur lokalen Begehbarkeitsberechnung, um die Privatsphäre zu schützen, so sollte die Pixelrasterebene angefragt und übertragen werden. Ist eine hochauflösende Visualisierung des 2,5D Modells gefragt, so wird die BIGML-Ebene die Anforderungen erfüllen. Damit hängt die Wahl des übertragenen Modells ganz klar von den Anforderungen auf Empfängerseite ab. Es sollte jedoch immer berücksichtigt werden, dass das BIGML-Umgebungsmodell möglicherweise geheime oder unter Schutz stehende Informationen enthält und so in vielen Fällen nur Teile davon übertragen werden dürfen. Da ein Fluchtplan meist in jedem Gebäude frei verfügbar aushängt, ist eine Verbreitung der Pixelrasterebene weniger kritisch. Es ist auch möglich, Visualisierungsobjekte eines beliebigen Formats, beispielsweise als OpenGL Vektoren, aus den vorhandenen Daten des hybriden Umgebungsmodells zu erzeugen und zu übertragen. Zudem sei noch angemerkt, dass sich das hybride Umgebungsmodell durch Übertragung der BIGML-Ebene auch auf anderen Geräten replizieren lässt.

Bearbeitungsanfragen

Für einige I-LBS ist es erforderlich, dass Daten im Umgebungsmodell geändert werden können. So kann es beispielsweise bei Rettungsdiensten im Falle eines Brandes dazu kommen, dass bestimmte Gebiete nicht mehr durchquert werden können, oder bei interaktiven Diensten ändern sich Informationen zu bestimmten Objekten. Besonders häufig ist dabei die Änderung semantischer Informationen von Objekten, was am geschicktesten durch das Ändern der *PropertyTags* in der BIGML-Ebene geschieht. Ändern sich aber die geometrischen Eigenschaften, wie durch das Entfernen von Wänden oder Vermauern von Türen, so muss neben der BIGML-Ebene auch der zugehörige Teil der anderen Ebenen aktualisiert werden. Die Änderungen sind glücklicherweise meist nur lokal durchzuführen, betreffen sie doch in den seltensten Fällen mehr als zwei Räume auf einmal. Änderungen an den Identifizierungsebenen erfordern dennoch eine Prüfung auf Überschneidung in Farbzuordnungen, um Inkonsistenzen beim Hinzufügen neuer Objekte zu vermeiden. Änderungen betreffen auch bei dem hybriden Modell immer die BIGML-Ebene, da alle weiteren Informationen aus dieser Ebene abgeleitet werden.

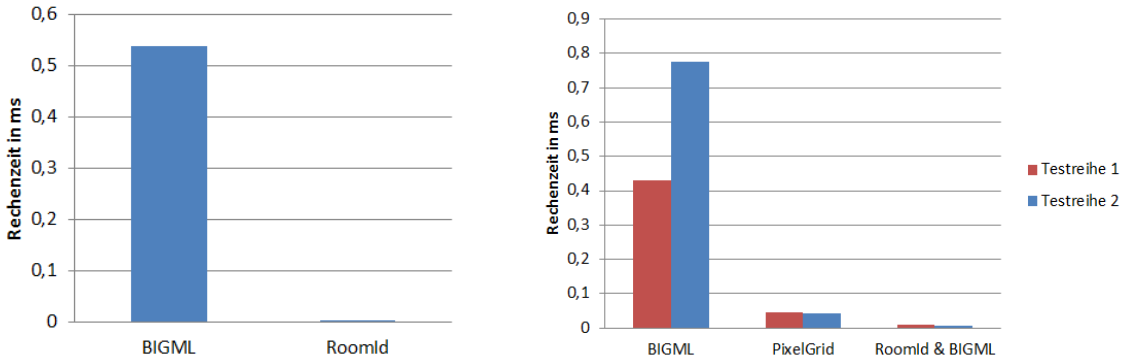
Bearbeitungsanfragen kann man auch dadurch ermöglichen, indem nur das BIGML-Modell geändert wird und die anderen Ebenen aus dem geänderten Modell neu erzeugt werden. Damit dies nicht zu temporären Dienstausfällen führt, sollten die Änderungen auf einer lokalen Kopie durchgeführt werden und erst bei vollständiger Übernahme der Änderungen auf allen Ebenen ein Umschalten auf das neue Modell erfolgen. Währenddessen sind keine weiteren Bearbeitungsanfragen möglich.

3.4.4 Evaluation

Nach der Vorstellung der Anfragebearbeitung wird die Effizienz der Bearbeitung in Bezug auf mögliche Alternativen in einem nicht hybriden Modell untersucht. Dazu wurde das hybride Umgebungsmodell für ein Gebäude der Ludwig-Maximilians-Universität München erstellt und als Grundlage für die Anfragebearbeitung genommen. Das Modell besteht aus fast 1.000 Räumen auf fünf Stockwerken, mehr als 1.000 unterschiedlichen Portalen, ca. 3.000 Knoten und 11.500 Kanten, verteilt über die Fläche des Grundstücks. Bei der Erzeugung der Rasterebenen aus dem Vektor-basierten Modell wurde dabei ein Skalierungsfaktor von 10 gewählt, wodurch ein einzelnes Pixel 10 cm^2 belegt. Die Größe der BIGML Datei, die zur Erzeugung des hybriden Modells genutzt wurde, war $8,6$ MB, der Speicherbedarf der Rasterebenen ca. 300 KB, allerdings ohne AOI-Identifizierungsebene.

Punkt-in-Polygon vs. Raumidentifizierungsebene

Einer der Nachteile des Vektor-basierten BIGML-Modells ist die aufwändige Berechnung des Raums, in dem eine geometrische Koordinate liegt. Dies geschieht durch mehrere Punkt-in-Polygon Tests, da hierbei das Enthaltensein für jeden Raum überprüft werden muss. Es reicht zwar in den meisten Fällen eine Überprüfung des Enthaltenseins im achsenparallelen umschließenden Rechteck eines Raums, das in BIGML ebenfalls angegeben wird, dennoch muss die Anfrage für jeden Raum einzeln durchgeführt werden, bis der richtige Raum gefunden wird. Daher wurde im hybriden Modell eine Raumidentifizierungsebene durch eine farbige Bitmap realisiert, in der jeder Raum eine eindeutige Farbe zugewiesen bekommt und alle Pixel des Raums in dieser Farbe eingefärbt werden. Wird nun für eine geometrische Koordinate das Enthaltensein abgefragt, so wird die Koordinate in das Pixelkoordinatensystem übertragen, die Farbe des entsprechenden Pixels ausgelesen und in der Zuordnungstabelle der zugehörige Raum bestimmt.



(a) Performanz der Raumidentifizierung

(b) Performanz der Kollisionsanfragen

Abbildung 3.7: Effizienz von verschiedenen Anfragen im Vergleich. [95]

Die Effizienz der beiden Verfahren wurde durch 1.000 Abfragen mit zufälligen begehbaren geometrischen Koordinaten miteinander verglichen. Das Ergebnis ist in Abbildung 3.7a

zu sehen. Während die Beantwortung der Anfrage in der BIGML-Ebene durchschnittlich $0,6$ ms dauert, benötigt die Beantwortung derselben Anfrage auf der Raumidentifizierungsebene mit $0,004$ ms weniger als ein 1 % der Zeit. Es ist allerdings anzumerken, dass es in 18 der 1.000 Fälle zu einer unterschiedlichen Zuordnung gekommen ist. Dies liegt an der beschränkten Auflösung der Bitmap, in allen Fällen wurde fälschlicherweise das Enthaltensein im Nachbarraum angenommen.

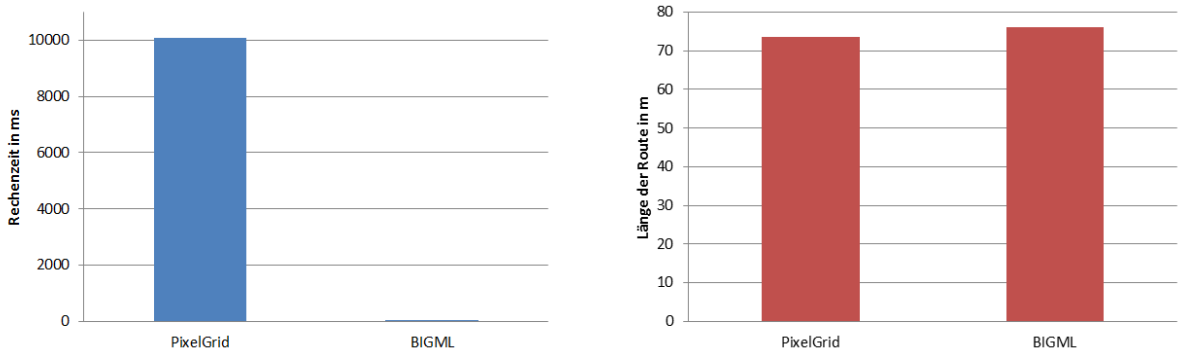
Kollisionsanfragen

Kollisionsanfragen können auf drei verschiedene Möglichkeiten beantwortet werden. Wird die Kollisionsanfrage direkt auf der BIGML-Ebene beantwortet, so wird solange auf Schnittpunkte zwischen der Kollisionslinie und allen anderen Linien im selben Stockwerk überprüft, bis entweder alle Linien getestet wurden oder es zu einer Kollision kam. Soll die Kollisionsanfrage auf der Pixelrasterebene durchgeführt werden, so wird mittels des Bresenham-Algorithmus überprüft, ob sich auf der Kollisionslinie ein schwarzes Pixel befindet. Zuletzt kann zudem zuerst der Raum mithilfe der Raumidentifizierungsebene ermittelt werden und anschließend ein Schnitt aller Raumkanten mit der Kollisionslinie durchgeführt werden. Im ersten und im letzten Fall muss allerdings bei einer Kollision noch getestet werden, ob die Kollisionslinie begehbarer Portalgeometrie entspricht.

Die Algorithmen wurden mithilfe von zwei unterschiedlichen Testreihen evaluiert. In der ersten Testreihe wurden 1.000 Paare zufälliger Koordinaten innerhalb eines Stockwerks überprüft, in der zweiten Testreihe 1.000 Paare zufälliger Koordinaten im selben Raum. Da ein Großteil der Räume im Modell konvex ist, sind in der ersten Testreihe eher negative Antworten, in der zweiten eher positive Antworten zu erwarten. Die Ergebnisse der Auswertung sind in Abbildung 3.7b dargestellt. Man kann deutlich erkennen, dass die durchschnittliche Ausführungszeit bei der BIGML-Ebene im Erfolgsfall steigt, da hierbei tatsächlich alle Linien überprüft werden müssen. Die Dauer der beiden anderen Algorithmen verhält sich in beiden Testreihen ähnlich. Tendenziell wird jedoch die Anfrage innerhalb eines Raums schneller beantwortet. In Testreihe 1 dauerten die Antworten bei der BIGML-Ebene im Schnitt $0,43$ ms, beim Pixelraster $0,05$ ms und bei der hybriden Methode $0,01$ ms. In Testreihe 2 betrugen die Zeiten im Schnitt $0,78$ ms bei der BIGML-Ebene, beim Pixelraster $0,04$ ms und bei der hybriden Methode $0,006$ ms. Hier erkennt man zudem einen großen Vorteil des hybriden Umgebungsmodells, das in diesem Fall nicht nur die Effizienz des für die Anfrage besser geeigneten Modells erreicht, sondern diese sogar noch deutlich übertreffen kann. Allerdings ist auch hier anzumerken, dass es zu unterschiedlichen Ergebnissen kommen kann. Im Fall der zweiten Testreihe wurden in 153 Fällen unterschiedliche Ergebnisse zurückgeliefert, wobei in 138 Fällen Bresenhams Algorithmus keine Sichtbarkeit ermitteln kann. Dies liegt daran, dass durch die Rasterisierung Diskretisierungsfehler bei den Randpixeln eines Raums in Kauf genommen werden müssen, sprich das Innere des Raums wird in der Pixelrasterebene verkleinert. Fällt eine der Koordinaten oder ein Pixel der gezeichneten Kollisionslinie auf so einen Randpixel, dann ist nach Bresenham keine Sichtbarkeit gegeben.

Routinganfragen

Zuletzt wird die Performanz von Routinganfragen auf dem hybriden Umgebungsmodell evaluiert. Dazu wird der Vergleich von Dauer der Anfragebearbeitung und Länge der resultierenden Wege miteinander verglichen. Während das Routing auf der Pixelrasterebene durch den A*-Algorithmus auf Pixelebene durchgeführt wird, wird in der Wegegraphebene auf den Knoten und Kanten des Graphen mittels Dijkstra geroutet. Um eine Vergleichbarkeit zu schaffen, wurden 1.000 mal die kürzesten Wege zwischen zwei zufälligen Graphknoten im selben Stockwerk ermittelt.



(a) Zeitaufwand des Routings

(b) Länge der Wege

Abbildung 3.8: Zeitdauer und Weglänge von Routinganfragen auf BIGML und Pixelrasterebene im Vergleich. [95]

Das Ergebnis der Auswertung ist in Abbildung 3.8 zu sehen. Dabei ist klar zu erkennen, dass das Routing auf Graphebene mit durchschnittlich 17 ms deutlich effizienter ist, als das Routing auf Pixelrasterebene, welches pro Anfrage durchschnittlich 10 s gedauert hat. Die Gründe liegen hierbei vor allem in der Anzahl an Knoten, die auf dem Weg untersucht werden. Die Zahl ist bei der Pixelrasterebene um ein Vielfaches größer, wodurch die Bearbeitungszeit steigt. In Bezug auf die Länge der zurückgegebenen Wege ist allerdings die Pixelrasterebene effizienter: mit durchschnittlich ungefähr $73,5$ m je Weg sind die Wege ca. $2,5$ m kürzer als die auf der Graphebene ermittelten Wege mit durchschnittlich ungefähr 76 m Länge. Allerdings liegt dies auch daran, dass die Wege auf Pixelrasterebene durch die Ecken in konkaven Räumen und an Türstöcken führen, während in der Graphebene Wert auf die Natürlichkeit und Begehbarkeit von Wegen gelegt wurde [62, 60] und die Wege daher einen Sicherheitsabstand zu Wänden einhalten. Dementsprechend sind die Wege zwar geringfügig kürzer, eignen sich aber nicht so gut für eine Vielzahl von I-LBS, die Navigation oder Routing beinhalten.

3.4.5 Bewertung des hybriden Umgebungsmodells

Das hybride Umgebungsmodell vereint die Vorteile des Vektor-basierten und des Raster-basierten Modells unter Vermeidung der einzelnen Schwächen. Es erfüllt die in Abschnitt

2.5.3 genannten Anforderungen (vgl. Tabelle 3.2).

	Transf.	Routing	Nachbarn	Bereich	Visual.	Sichtl.	POIs	Tagging
BIGML	x	x	x	o	x	o	x	x
Bitmap	x	o	o	x	x	x	x	o
Hybrid	x	x	x	x	x	x	x	x

Tabelle 3.2: Bewertung der vorgestellten Umgebungsmodelle anhand der Anforderungen.

Wie in Abschnitt 3.4.4 gezeigt wurde, werden die einzelnen Anfragen immer so effizient wie möglich beantwortet. Aufgrund der Hybridität ergeben sich dabei sogar Möglichkeiten, schneller zu antworten als es in beiden zugrundeliegenden Einzelmodellen möglich ist. Durch die Kombination der Modelle ergibt sich zwar ein höherer Speicherbedarf im Vergleich zu den jeweiligen Einzelmodellen. Da sich das hybride Umgebungsmodell jedoch vollständig aus dem BIGML-Modell erzeugen lässt, entfällt der erhöhte Speicherbedarf bei Übertragungen des Modells. In diesem Fall reicht eine Übertragung des BIGML-Modells vollständig aus.

3.5 Zusammenfassung

In diesem Kapitel wurden zunächst zwei unterschiedliche Umgebungsmodelle für ortsbezogene Dienste in Gebäuden vorgestellt. Das eine Modell namens BIGML ist als GML Anwendungsschema definiert und basiert auf Vektordaten und semantischen Informationen. Es lässt sich entsprechend der Klassifizierung von [14] als hierarchisches und graphbasiertes Umgebungsmodell einordnen, da es explizite topologische Informationen über die Begehbarkeit und Verknüpfung von räumlichen Strukturen in Form eines Wegpunktgraphen enthält. Zudem wurde gezeigt, wie ein solches Modell automatisch aus bestehenden CAD-Daten erzeugt werden kann, und es wurden die Vor- und Nachteile diskutiert.

Das zweite Modell besteht aus mehreren Bitmaps und modelliert semantische Informationen durch farbliche Kennzeichnung von bestimmten Gebieten. Es wurde gezeigt, wie ein solches Modell je Stockwerk eines Gebäudes aus einer einfachen Bitmap erzeugt wird. Dazu wurden Algorithmen, wie Objekterkennung und Routing, mithilfe von Bildbearbeitungsmethoden umgesetzt. Es wurde zudem festgestellt, dass die Vorteile des Bitmap-basierten Modells gerade den Nachteilen des BIGML-Modells entsprechen.

Aus diesem Grund wurde zuletzt ein hybrides Umgebungsmodell aus den beiden Modellen entwickelt, welches die üblichen Anfragen, wie Positionsanfragen, Bereichsanfragen und Routing, aus Sicht der ortsbezogenen Dienste effizient beantwortet. Da das Modell direkt aus dem BIGML-Modell erzeugt wird, lässt es sich ebenso direkt automatisch aus CAD-Daten erzeugen. Somit öffnet das hybride Umgebungsmodell neue Möglichkeiten, die Verbreitung von ortbezogenen Diensten zu fördern und bestehende Hindernisse bezüglich der Verfügbarkeit von Umgebungsdaten zu überwinden.

Kapitel 4

Infrastrukturlose und Infrastruktur-basierte Positionsbestimmung

Nachdem mit den Umgebungsmodellen aus Kapitel 3 bereits eine wichtige Grundlage für ortsbezogene Dienste innerhalb von Gebäuden geschaffen wurde, wird in diesem Kapitel auf eine Positionsbestimmung in Gebäuden eingegangen, welche die Anforderungen seitens der Dienste optimal erfüllt. Nach einer kurzen Vorstellung der Eigenschaften von Ortungssystemen wird zunächst eine infrastrukturlose Positionsbestimmung auf dem Smartphone mithilfe von Beschleunigungssensor, Kompass und Gyroskop vorgestellt. Die Sensoren werden in einem Partikelfilter kombiniert, und Mapmatching eingesetzt, um kumulative Fehler beim gewählten Koppelnavigationsansatz zu verringern. Anschließend werden zwei Infrastruktur-basierte Fingerprinting Systeme zur Positionsbestimmung auf Basis von Kamerabildern oder WLAN vorgestellt. Als Alternative zur infrastrukturlosen Positionsbestimmung wird eine oftmals in Gebäuden bereits vorhandene Infrastruktur genutzt, die absolute Positionsbestimmung erlaubt, dazu jedoch eine initiale Kalibrierung erfordert. Gerade bei WLAN-Fingerprinting werden einige Erweiterungen vorgestellt, wie die Hinzunahme eines Richtungsmessers zur Blickrichtungserkennung, Beschleunigungssensoren zur Vorhersage der Bewegung und ein online Fehlerschätzverfahren. Letzteres spielt bei WLAN-Fingerprinting eine besondere Rolle, um ein geeignetes Messmodell für Bayesische Filter angeben zu können. Abschnitt 4.4 schließlich fasst die gewonnenen Erkenntnisse und die wichtigsten Ergebnisse zusammen.

4.1 Eigenschaften von Ortungssystemen

Die Anforderungen an die Positionsbestimmung aus Sicht der ortsbezogenen Dienste sind vielfältig und kommen ebenso wie die Anforderungen an die Umgebungsmodelle aus den vielen verschiedenen Anwendungsgebieten innerhalb von Gebäuden. Die Anforderungen wurden in Abschnitt 2.5.4 bereits vorgestellt und beziehen sich auf die folgenden Eigen-

schaften von Ortungssystemen:

- **Genauigkeit**, im Englischen *accuracy*, bezeichnet die Abweichung von der tatsächlichen Position und wird durch die mittlere Abweichung gemessen.
- **Präzision**, im Englischen *precision*, bezeichnet die Streuung der Positionsschätzung und wird durch die Standardabweichung der mittleren Abweichung gemessen oder durch die kumulative Verteilungsfunktion angegeben.
- Die **Abdeckung** wird durch den Anteil der Fläche, in dem eine Positionsbestimmung möglich ist, in Abhängigkeit der Gesamtfläche eines Gebäudes gemessen.
- Die **Anschaffungskosten** werden durch die Investitionskosten zur Inbetriebnahme des Systems pro Gebäude gemessen.
- Die **Betriebskosten** werden durch die laufenden Kosten des Systems pro Gebäude gemessen.
- Es wird die **Infrastruktur** angegeben, die zur Bereitstellung des Systems notwendig ist.
- Die **unterstützte Nutzerzahl** wird am Mehraufwand je zusätzlichem Nutzer pro Gebäude gemessen.
- Der **Ergebnistyp** wird als relative oder absolute, symbolische oder geometrische und deterministische oder probabilistische Positionsangabe angegeben.
- Die **Verfügbarkeit** wird durch die Verfügbarkeit der benötigten Technologie in Gebäuden und/oder auf Smartphones dargestellt.
- Die **Zuverlässigkeit** wird anhand der Fähigkeit, mit unbekannten, fehlenden oder gestörten Daten umzugehen, bewertet.
- Die **Update-Rate** wird durch die Zeit zwischen aufeinanderfolgenden Positionsmessungen dargestellt.
- Der **Schutz der Privatsphäre** wird in Abhängigkeit der verschiedenen Komponenten des Systems, an denen die Position bekannt ist oder sich bestimmen lässt, gegeben.

Die Bewertung in Bezug auf diese Eigenschaften ermöglicht eine Vergleichbarkeit von Positionierungssystemen auch über technologische Grenzen hinweg. Je nach Anwendungsfall kann das dazu am besten passende System ausgewählt werden. Daher werden die nachfolgenden Verfahren soweit wie möglich auf die genannten Eigenschaften hin untersucht und für den Einsatz in ortsbezogenen Diensten bewertet. Zunächst wird ein kurzer Überblick über die in heutigen Smartphones verbauten Sensoren gegeben:

- Aktuelle Smartphones verfügen über eine Vielzahl an Funkschnittstellen: Mobilfunk in GSM, UMTS und LTE, GPS, WLAN, Bluetooth und NFC. Funkbasierte Verfahren lassen sich vor allem zur absoluten Positionsbestimmung einsetzen, benötigen jedoch eine bestehende Infrastruktur.
- Die Front- und Rückkamera bieten als optische Sensoren die Möglichkeit zur absoluten und relativen Positionsbestimmung. Dazu wird keine Infrastruktur benötigt, allerdings ist eine initiale Kalibrierung zur absoluten Positionsbestimmung erforderlich.
- Manche Sensoren messen einen Erdeffekt. Der Magnetfeldsensor misst die Richtung und Stärke des Erdmagnetfeldes, wodurch eine absolute Richtungsbestimmung ermöglicht wird. Barometer erlauben es, relative Höhenänderungen zu ermitteln und somit vor allem Stockwerkswechsel zu erkennen. Beide Sensoren eigenen sich zur Verbesserung der Positionsbestimmung.
- Zuletzt können die im Smartphone verbauten inertialen Sensoren, der Beschleunigungssensor und das Gyroskop dazu genutzt werden, eine relative Positionsänderung zu messen. Der Beschleunigungssensor misst dabei die Beschleunigung bezüglich der drei Achsen des Smartphones und das Gyroskop die Winkelgeschwindigkeit der Drehung bezüglich dieser Achsen.

4.2 Koppelnavigation ohne Infrastruktur auf Smartphones

Um das Ziel einer einfachen und flexiblen Positionsbestimmung in Gebäuden von genügender Genauigkeit und ohne zusätzliche Kosten zu erreichen, kommt einem zunächst Dead Reckoning mit den inertialen Sensoren des Smartphones in den Sinn. Dead Reckoning oder Koppelnavigation beschreibt die Messung der zurückgelegten Entfernung mitsamt Richtung von einem bekannten Startpunkt aus und wurde bereits in Abschnitt 2.1 vorgestellt. An dieser Stelle werden nun verschiedene Komponenten für Dead Reckoning Systeme auf dem Smartphone vorgestellt und anschließend in einem Partikelfilter für probabilistisches Dead Reckoning auf dem Smartphone vereint (siehe Abbildung 4.1), wodurch eine Positionsbestimmung innerhalb von Gebäuden ohne Infrastruktur ermöglicht wird. Den Beginn macht dabei die Einführung von Schritterkennung mithilfe des Beschleunigungssensors auf dem Smartphone, wobei neben der Schritterkennung insbesondere auf die adaptive Schrittlängenschätzung und die Schrittrichtungserkennung eingegangen wird. Gerade in die Schritterkennung und die Schrittrichtungserkennung fließen dabei Ergebnisse aus einer Zusammenarbeit mit David Grotzky [41] ein. Die Mechanismen werden anschließend in einem Partikelfilter zum Dead Reckoning im Gebäude vereint. Zuletzt wird die im Filter angewandte Kartenkorrektur, auch Mapmatching genannt, eingeführt, welche die Plausibilitätsüberprüfung einer Position und gegebenenfalls Korrektur der Positionsschätzung anhand einer Karte oder eines Umgebungsmodells ermöglicht.

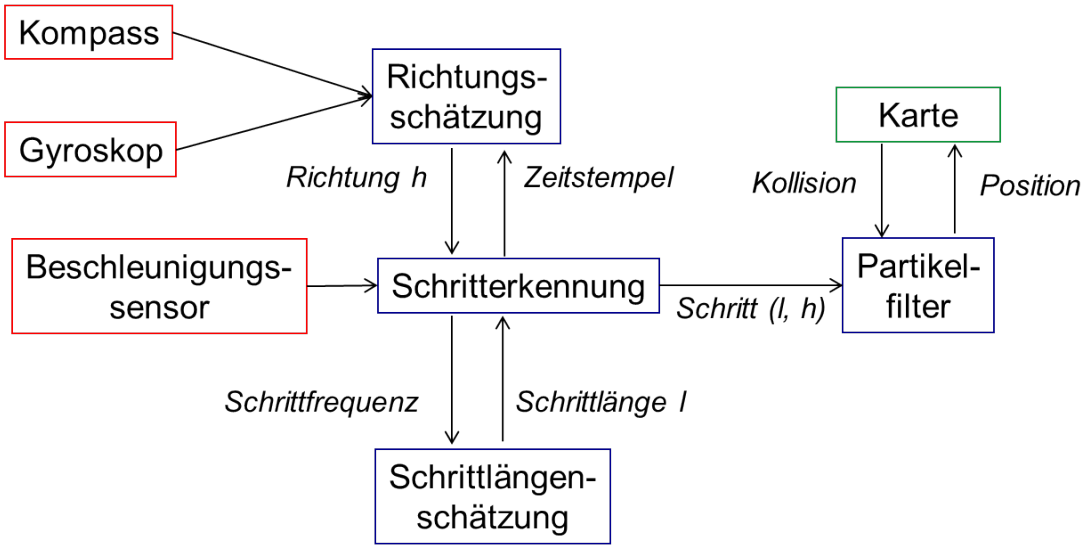


Abbildung 4.1: Überblick über die verschiedenen Komponenten des Partikelfilters.

Als reine relative Positionsbestimmung benötigt Dead Reckoning meist eine feste Startposition oder Startverteilung, mit deren Hilfe aus der relativen Positionsänderung eine absolute geometrische Position oder eine neue Verteilung bestimmt werden kann. Jeder Fehler in der relativen Positionsschätzung addiert sich über die Zeit auf und kann somit nach einiger Zeit zu großen Ungenauigkeiten führen, was durch Mapmatching vermieden oder zumindest verringert werden soll. Daher wird die Genauigkeit von Dead Reckoning zusätzlich zur mittleren Abweichung durch die Abweichung der Endposition von der tatsächlichen Position in Abhängigkeit von der zurückgelegten Strecke angegeben. Der große Vorteil solcher Systeme liegt darin, dass Koppelnavigation ohne jegliche Infrastruktur auskommt und somit sehr flexibel eingesetzt werden kann.

4.2.1 Schritterkennung

Viele ältere Koppelnavigationsverfahren, wie die Arbeit von Widyawan [140, Seite 33 ff.], nutzen Bewegungsmodelle, um die Bewegung eines Ziels zu modellieren. Dabei werden Verteilungen für die Geschwindigkeit und die Bewegungsrichtung angenommen, die für Fußgänger auch plötzliche Richungs- und Geschwindigkeitsänderungen beinhalten. Bessere Ergebnisse liefert die Schritterkennung auf den Sensoren des Smartphones, insbesondere dann, wenn die Position ohne zusätzliche Messverfahren bestimmt wird [42]. Es ist zwar prinzipiell auch möglich, mit Beschleunigungssensoren per Doppelintegration direkt auf die zurückgelegte Strecke zu schließen, allerdings sind die in aktuellen Smartphones verbauten Sensoren nicht für diese Nutzung ausgelegt und so ungenau, dass bereits nach wenigen Sekunden Abweichungen im Bereich einiger Meter auftreten [117].

Aktuell sind verschiedene Algorithmen zur Schritterkennung verbreitet, die sich in der Trageart des Smartphones, der Vorfilterung der Beschleunigungssensordaten und der ei-

gentlichen Schritterkennung unterscheiden. Link et al. stellen in [79] einen einfachen Algorithmus zur Schritterkennung auf einem in der Hand gehaltenen Smartphone vor, bei dem die Beschleunigungsänderung über ein bestimmtes Zeitintervall genutzt wird, um Schritte zu erkennen. Dabei können jedoch auch andere Bewegungen irrtümlicherweise als Schritt erkannt werden. In [42] nutzen Gusenbauer et al. eine Support Vector Maschine, um zwischen den Zuständen Gehen, Laufen, Treppensteigen, Aufzugfahren, still Stehen und unregelmäßigen Bewegungen zu unterscheiden. Allerdings wird zur reinen Schritterkennung die Erkennung lokaler Maxima auf der gefilterten Gesamtbeschleunigung genutzt. In [105] erkennen Rai et al. Schritte anhand von periodisch wiederkehrenden Signalen des Beschleunigungssensors.

Im Folgenden wird ein eigener Algorithmus vorgestellt, der die Ansätze aus [79] und [42] kombiniert. Zunächst wird dazu der Fall der waagerechten Handhaltung betrachtet und anschließend auf beliebige Haltung erweitert.

Waagerechte Handhaltung

Bei einer festen Ausrichtung des Smartphones wird davon ausgegangen, dass die Zuordnung des Smartphone-Bezugssystems zum Bezugssystem des Nutzers gleichbleibend und bekannt ist. Typische Ausrichtungen des Smartphones sind waagerechte Handhaltung, senkrechte Handhaltung, Befestigung am Gürtel, Transport in der Hosentasche und Transport in der Handtasche. In den letzten beiden Fällen wird meist ein Algorithmus für eine beliebige Ausrichtung angewandt, da das Smartphone innerhalb der Tasche verschiedene Ausrichtungen annehmen oder auch verrutschen kann. Da die Befestigung am Gürtel in der Realität nur wenig Anwendung findet und zudem in diesem Fall meist keine aktive Dienstnutzung ortsbezogener Dienste erfolgt, wird an dieser Stelle ein Ansatz zur Schritterkennung vorgestellt, der von einer nahezu waagerechten Haltung des Mobiltelefons ausgeht. Dies entspricht dem Fall, dass der Nutzer beispielsweise Karten- oder Navigationsdaten auf dem Display des Smartphones betrachtet.

In diesem Fall zeigt die z-Achse des Handy-Bezugssystems senkrecht auf den Boden. Da sich bei einem Schritt der menschliche Körper zuerst etwas hebt und anschließend wieder absenkt, sinkt im Verlauf eines Schrittes die vertikal nach unten gerichtete Beschleunigung zuerst und steigt dann wieder an. Solche Muster gilt es aus den rohen Daten des Beschleunigungssensors zu ermitteln. Da jedoch die Daten des Filters einem Rauschen unterliegen und die Muskelbewegung in einer ruckartigen Bewegung resultiert, müssen diese vor der Weiterverarbeitung gefiltert werden. Dazu bietet sich ein Tief-Pass Filter an, der die niederfrequenten Anteile des vom Beschleunigungssensor ausgegebenen diskretisierten Signals behält und das hochfrequente Rauschen entfernt. Ein diskreter Tief-Pass Filter kann durch die folgende Formel mithilfe eines glättenden Mittelwerts umgesetzt werden:

$$\overline{x}_i = \alpha x_i + (1 - \alpha)\overline{x}_{i-1} \tag{4.1}$$

$\overline{x}_k$ entspricht dabei dem glättenden Mittelwert, also dem Ergebnis des Tief-Pass Filters für den Messwert x_k des Sensors. α ist der Glättungskoeffizient, der die Abschnittsfrequenz

beeinflusst. Je nach gewählter Frequenz f_a und der Eingangsfrequenz f_e des Signals kann α mithilfe der folgenden Formel bestimmt werden [87]:

$$f_a = \frac{f_e}{2\pi\left(\frac{1}{\alpha} - 1\right)} \tag{4.2}$$

Zur Schritterkennung kann nun in den gefilterten Beschleunigungsdaten nach lokalen Minima und Maxima gesucht werden. Ein Schritt ist genau dann gemacht worden, wenn sich innerhalb einer gewissen Zeitspanne, die hier anhand der angenommenen minimalen Gehfrequenz von 1 Hz auf 1 s gelegt wurde, ein lokales Minimum auf ein lokales Maximum folgt und diese mindestens den empirisch bestimmten Wert 4 $\mathrm{m/s^2}$ auseinanderliegen. Nach jedem erkannten Schritt wird der Beginn der Zeitspanne auf den Zeitpunkt des lokalen Minimums gesetzt und anschließend zuerst das nächste lokale Maximum und anschließend das nächste lokale Minimum neu gesucht. Mit diesem Verfahren wurden in fünf Testläufen unterschiedlicher Schrittfrequenzen im Bereich $1,5 - 2,2$ Hz auf 50 Meter Länge zwischen 90 % und 100 % der Schritte korrekt erkannt. Zwischen 1,7 Hz und 1,9 Hz lag der maximale Fehler sogar unter 1,5 %. In keinem Fall wurden zu viele Schritte gezählt. Dennoch können bei dem Verfahren auch irreguläre Bewegungen als Schritt erkannt werden oder einzelne Schritte unentdeckt bleiben.

Beliebige Haltung

Zur Realisierung einer beliebigen Haltung des Smartphones gibt es zwei Möglichkeiten. Zum einen kann der eben beschriebene Algorithmus anstelle der vertikalen Beschleunigung wie in [42] auf die Gesamtbeschleunigung angewendet werden, zum anderen kann auch die Ausrichtung des Smartphones im Weltbezugssystem, Osten, Norden und senkrecht nach oben, nach [147] ermittelt und der obige Ansatz für die jeweilige senkrecht zum Boden stehende Beschleunigungskomponente durchgeführt werden. Dazu wird die Rotationsmatrix R zur Umrechnung zwischen Bezugssystemen mithilfe des normierten Gravitationsvektors $g = -G/\left|G\right|$ und des Magnetfeldvektors M des Smartphones nach [147] wie folgt ermittelt:

$$m = \frac{M - (g^T M)g}{|M - (g^T M)g|}$$

$$e = \frac{m \times g}{|m \times g|}$$

$$R = \begin{pmatrix} e_1 & e_2 & e_3 \\ m_1 & m_2 & m_3 \\ g_1 & g_2 & g_3 \end{pmatrix}$$

Daraus lässt sich nun aus den Beschleunigungsvektor V_S im Bezugssystem des Smartphones die vertikale Beschleunigung V_{W_3} bezüglich des Weltbezugssystems nach Gleichung (4.3) ermitteln und entsprechend dem Ansatz der waagrechten Handhaltung Schritte erkennen.

$$V_W = RV_S \tag{4.3}$$

Diese Umrechnung der Bezugssysteme lässt sich nicht nur auf die Schritterkennung anwenden, sondern ebenfalls zur Erkennung der Schrittrichtung aus den Werten des Gyroskops.

Schrittrichtungserkennung

Neben der reinen Erkennung eines Schrittes ist die Bestimmung der Schrittrichtung von großer Bedeutung für die Koppelnavigation. Generell lässt sich die Richtung besonders gut mithilfe zweier Sensoren im Smartphone berechnen: dem digitalen Kompass und dem Gyroskop. Erster zeigt in Richtung des magnetischen Nordpols und lässt somit die Berechnung der Abweichung der aktuellen Bewegungsrichtung von Norden zu. Das Gyroskop hingegen kann nicht die absolute Ausrichtung des Smartphones bestimmen, sondern lediglich Änderungen in der Ausrichtung in Form der Winkelgeschwindigkeit der Rotation. Da das Gyroskop allerdings im Gegensatz zum Kompass nicht von magnetischen Störquellen oder elektromagnetischer Strahlung beeinflusst wird, kommt es häufig als Korrekturgröße für den Kompass zum Einsatz [105]. Die direkte Nutzung der Rohdaten des Gyroskops ist aufgrund der akkumulierenden Ungenauigkeit der verbauten schlechten Sensoren nach derzeitigem Stand nicht zu empfehlen [117].

Bei einer annähernd waagerechten Haltung des Smartphones lässt sich die Schrittrichtung direkt aus dem Kompass auslesen, bei beliebiger Haltung kann die Bewegungsrichtung aus der Richtung der Hauptkomponente der nicht senkrechten Beschleunigung geschlossen werden [147] oder durch bekannte Anfangsausrichtung und kontinuierliche Überwachung der Änderungen [77], wobei bei einem Partikelfilter auch die Bewegungsrichtung der überlebenden Partikel miteinbezogen werden kann. Aufgrund des Rauschens ist in beiden Fällen eine Filterung des Rohsignals von Nutzen, was durch Gleichung 4.1 erreicht wird. α wird hier anhand der erkannten Schrittfrequenz dynamisch bestimmt, wobei die Abschnittsfrequenz der Schrittfrequenz entspricht. Auch die Werte des Gyroskops werden analog gefiltert, und mithilfe von Gleichung (4.3) wird die Bewegungsrichtung durch einfache Integration der Rotationsgeschwindigkeit um die senkrecht zum Boden stehende Z-Achse im Weltbezugssystem ermittelt.

Wie bereits angesprochen, ist der Kompass gegenüber von Störquellen sehr empfindlich, weswegen eine Korrektur mithilfe des Gyroskops die Genauigkeit eines Dead Reckoning Systems stark erhöhen kann. Dazu muss man sich nochmal in Erinnerung rufen, dass die Fehlerquellen beim Kompass zu kurzfristigen starken Schwankungen führen, während beim Gyroskop die langfristige Aufsummierung einzelner Fehler das größte Problem darstellt. Daher wird für die Bewegungsrichtung ein gewichtetes Mittel zwischen beiden Sensoren genutzt, dessen Gewicht dynamisch an die Werte angepasst wird. Dazu wird aus den letzten zwei Kompass Messungen die Differenz durch den zeitlichen Abstand der Messungen geteilt, wodurch eine Winkelgeschwindigkeit in Grad pro Sekunde berechnet wird. Der Einfluss des Gyroskops auf die Richtungsschätzung wird dabei durch den Abstand der vom Kompass gemessenen Winkelgeschwindigkeit ω_k von der vom Gyroskop gemessenen Geschwindigkeit

ω_g bestimmt (siehe Gleichung (4.4)).

$$\beta = \frac{|\omega_k - \omega_g|}{180°\text{s}^{-1}} \tag{4.4}$$

Je größer die Abweichung, desto größer ist vermutlich der Fehler des Kompass und desto größer sollte der Einfluss des Gyroskops auf die Richtungsschätzung sein. Daher wird die Bewegungsrichtung d durch das mit β bzw. $(1 - \beta)$ gewichtete Mittel der Richtungsschätzung des Gyroskops d_g und des Kompass d_k berechnet:

$$d = \beta d_g + (1 - \beta) d_k \tag{4.5}$$

Eine Alternative zu diesem Vorgehen ist die Definition eines Schwellwertes σ. Ist die Abweichung $|\omega_k - \omega_g|$ größer als σ, so wird der Wert des Gyroskops anstelle des Kompass genommen. Dieses Verfahren wird mit der vorher genannten Interpolation kombiniert, wodurch kleine Fehler kontinuierlich korrigiert und große Fehler vermieden werden. Aufgrund von Versuchen in einer geringfügig gestörten Umgebung wurde eine Standardabweichung des Kompass von 5° festgestellt, die durch vergleichbare Werte aus der Literatur [42] bestätigt wird. Im Falle einer Abweichung des Kompass um mehr als 15° kann also mit über 99 % Sicherheit von einer Störung ausgehen. Nimmt man an, dass die Genauigkeit des Gyroskops über kurze Zeit deutlich höher ist, so kann man bei einer Abweichung von ca. 20° von einem Fehler ausgehen. Daher wurde im entwickelten System $\sigma = 20°$ gewählt. Die Einbeziehung des Gyroskops bietet eine merkliche Stabilisierung der Richtungsinformationen, wie in Abbildung 4.2 zu sehen ist.

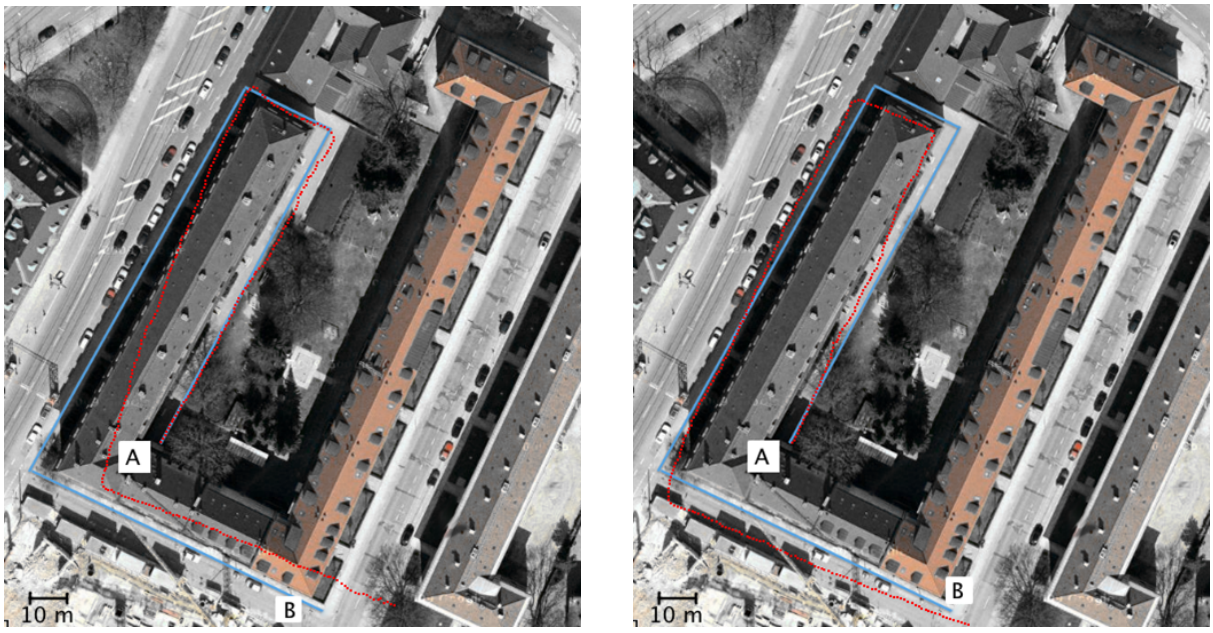
Schrittlängenerkennung

Nach der Schritterkennung und der Schätzung der Schrittrichtung muss noch die Länge des Schrittes bestimmt werden. Diese ist abhängig von den physischen Eigenschaften, wie Größe oder Beinlänge, und der Schrittfrequenz des Nutzers. Die einfachste aber auch ungenaueste Möglichkeit ist dadurch gegeben, dass die Schrittlänge für alle Nutzer als konstant und fest angenommen wird. Hier sind statistisch ermittelte Werte zwischen 60 und 100 Zentimetern sinnvoll.

Eine deutlich genauere Möglichkeit besteht darin, die Schrittlänge l in Anhängigkeit von der Schrittfrequenz f_l zu berechnen. Tests der linearen Regression legen eine lineare Beziehung der Schrittlänge von der Schrittfrequenz nahe (siehe Gleichung (4.6)), wobei der Koeffizient k jedoch personenabhängig ist und durch die Körpergröße bzw. Beinlänge beeinflusst wird. λ entspricht dabei einem Rauschen, das einer normalverteilten Zufallsvariablen entsprechend dem Residuum der lineare Regression entspricht.

$$l = k f_l + \lambda \tag{4.6}$$

Der Koeffizient kann durch explizite Kalibrierung gewonnen werden. Dazu läuft der Nutzer eine bekannte Distanz, wobei die Anzahl der Schritte und die Schrittfrequenz gemessen werden. Geht man von einer direkt proportionalen Abhängigkeit aus, die durch



(a) Ohne Gyroskop (b) Mit Gyroskop

Abbildung 4.2: Verbesserung der Richtungsschätzung durch die Einbeziehung des Gyroskops. Startpunkt A, Endpunkt B, Grundwahrheit (blau), Positionsschätzung (rot). [41]

den Nullpunkt geht (Schrittfrequenz null heißt Schrittlänge null, vgl. Gleichung (4.6)), so lässt sich bereits mit einer Kalibrierung der Koeffizient k ermitteln. Diese Methode ist zumindest im Bereich gebräuchlicher Schrittfrequenzen zwischen 1,7 Hz und 2,0 Hz recht zuverlässig. In diesem Fall wurden bei drei Testläufen mit Schrittfrequenzen im Bereich 1,7 Hz bis 1,9 Hz auf einer Strecke von 50 Metern Abweichungen unter 2 % gemessen. Bei Schrittfrequenzen von 1,5 Hz und 2,2 Hz wurden allerdings auf derselben Strecke Abweichungen zwischen 13 % und 17 % ermittelt [41]. Leider besteht in den seltensten Fällen die Möglichkeit der expliziten Kalibrierung. Hierfür können andere Sensoren genutzt werden, wie GPS [42] oder die über eine bestimmte Zeit vom Partikelfilter gemessene Entfernung.

Eine weitere Möglichkeit besteht darin, die Schrittlänge über die Zeit der Positionsbestimmung hinweg zu verfeinern, indem über ein externes Korrekturverfahren wie Mapmatching falsche Annahmen erkannt und korrigiert werden. Im Falle eines Mehrhypothesenansatzes wie einem Partikelfilter kann die Schrittlänge in den Zustandsvektor mit aufgenommen werden. Werden manche Hypothesen über die Zeit wahrscheinlicher, während andere Hypothesen wegfallen, so entspricht der Zustand und damit insbesondere die Schrittlänge der gewichtigeren Hypothesen mit hoher Wahrscheinlichkeit der Realität. So lässt sich ein initial geschätzter oder empirisch ermittelter Koeffizient anpassen oder sogar eine initial als fest angenommene und durch zufälliges Rauschen gestörte Schrittlänge mit der Zeit verbessern [105]. Auf diesen Ansatz wird bei der Vorstellung des Partikelfilters zum Dead Reckoning im folgenden Abschnitt genauer eingegangen.

4.2.2 Partikelfilter

Zur Umsetzung des Dead Reckoning Systems auf dem Smartphone wurde ein Partikelfilter gewählt. Die Wahl begründet sich auf den Eigenschaften des Filters, nämlich der Fähigkeit, nicht normalverteilte Daten bearbeiten und darstellen zu können, insbesondere bei multi-modalen Verteilungen. Zudem lässt sich bei einem Partikelfilter ein besonders effizientes Mapmatching Verfahren einsetzen, welches im nächsten Abschnitt genauer erläutert wird. Das Mapmatching ist auch der Grund, warum der Partikelfilter dem Kalmanfilter vorzuziehen ist, da nach erfolgtem Mapmatching keine Normalverteilung mehr vorliegt. Die Grundlagen eines Partikelfilters wurden bereits in Abschnitt 2.1 dargestellt, daher wird an dieser Stelle vor allem auf die Details der Umsetzung eingegangen.

Der Systemzustand und der Zustand eines Partikels bestehen aus einem Vektor, in dem die zwei-dimensionale Position, eine Stockwerksangabe, eine Gebäudeangabe, die Blickrichtung und die Schrittlänge modelliert werden. Jedes Partikel hat zudem noch ein Gewicht, das die Wahrscheinlichkeit darstellt, mit welcher sich der Nutzer in dem Zustand des Partikels befindet. Der Systemzustand enthält stattdessen die gewichtete Standardabweichung der Positionen der Partikel von deren Mittelwert, die als Indikator der Genauigkeit und Präzision des Filters im aktuellen Zustand dient.

Der Systemzustand z berechnet sich als gewichtetes Mittel der Partikelzustände p, wobei das Gewicht durch die Wahrscheinlichkeit w_i jedes Partikels p_i bestimmt wird:

$$z = \sum_{i=1}^{N} w_i p_i \tag{4.7}$$

Die gewichtete Standardabweichung σ wird durch folgende Gleichung bestimmt:

$$\sigma = \sum_{i=1}^{N} n \left((x - x_i)^2 + (y - y_i)^2 \right) w_i \tag{4.8}$$

Initialisiert wird der Partikelfilter mit einer festen Anzahl N von Partikeln entweder an einem durch Nutzereingabe festgelegten Punkt oder gleichmäßig über die begehbare Fläche des Gebäudes. Daher kann die Anzahl N der Partikel an das Gebäude und die Art der Initialisierung angepasst werden. Initial hat jedes Partikel das gleiche Gewicht N^{-1}.

Als rekursiver Filter wird der Partikelfilter immer dann aktualisiert, wenn eine Änderung im beobachteten dynamischen System wahrgenommen wird. Im Fall von Dead Reckoning wird die Änderung durch einen Schritt ausgelöst. Immer, wenn ein Schritt erkannt wird, wird die neue Position p_i jedes Partikels mithilfe der Schrittrichtungsschätzung d_i und Schrittlängenschätzung l_{i-1} ausgehend von der letzten Position p_{i-1} durchgeführt (siehe Gleichung 4.9). Dazu wird die Position eines Partikels um eine zufällig um λ_{i-1} gestörte Länge l_{i-1} in eine zufällig um θ_i gestörte Richtung d_i verschoben. Die zufälligen Störungen sind unabhängig und jeweils normalverteilt entsprechend $\mathcal{N}_{0,\sigma_\lambda}$ bzw. $\mathcal{N}_{0,\sigma_\theta}$. Die jeweiligen Standardabweichungen σ_λ und σ_θ sind fest im System eingestellt und können sowohl von dem Nutzer, als auch der Umgebung abhängen. Die Schrittlänge l_{i-1} ist die Schrittlänge

des Systemzustands zum Zeitpunkt $i-1$, wodurch die Länge der durch Mapmatching entfernten Partikeln einer offensichtlich falschen Länge oder Richtung nicht in die Berechnung einfließt. Somit kann eine anfänglich unbekannte Schrittlänge über die Zeit immer genauer eingegrenzt werden.

$$x_i = x_{i-1} + (l_{i-1} + \lambda_{i-1}) \begin{pmatrix} \cos(d_i + \theta_i) \\ \sin(d_i + \theta_i) \end{pmatrix} \tag{4.9}$$

Neben der reinen Bewegung der Partikel wird zudem das Gewicht geändert. Dazu wird die Plausibilität der Bewegung eines Partikels durch das Mapmatching überprüft. Unmögliche Bewegungen, also Kollisionen, eines Partikels werden dadurch gehandhabt, dass die Wahrscheinlichkeit des Partikels auf null gesetzt wird. Damit die Gesamtwahrscheinlichkeit jedoch noch eins entspricht und die Partikel somit eine Wahrscheinlichkeitsverteilung approximieren, müssen alle Partikel neu gewichtet werden. Dieser Vorgang wird nach der Bearbeitung jedes erkannten Schrittes durchgeführt. Dabei werden alle Partikel entfernt, deren Gewicht kleiner als N^{-2} ist und Partikel, deren Gewicht w_i größer als $1,5\ \mathrm{N}^{-1}$ in die echt gerundete ganzzahlige Anzahl $n = w_i N$ geteilt. Letzteres geschieht, um Degenerationen, wie in [4] beschrieben, zu vermeiden.

Sollte dennoch der Fall eintreten, dass alle Partikel sterben, so wird eine neue Partikelwolke berechnet, welche entweder in einem gewissen Radius um die letzte bekannte Position, gleichverteilt über das Gebäude oder durch manuelle Nutzereingabe initialisiert wird.

4.2.3 Mapmatching

Wie bereits angesprochen wurde, addieren sich kleine Fehler dadurch über die Zeit auf, dass beim Dead Reckoning nur relative Positionsänderungen erfasst werden. Dies führt zu einer immer ungenaueren Ortung über die Zeit (siehe Abbildung 4.3). Um eine zeitunabhängige Korrektur der Fehler zu ermöglichen, können weitere Informationen über die Umgebung, beispielsweise Baupläne oder Umgebungsmodelle, hinzugezogen werden.

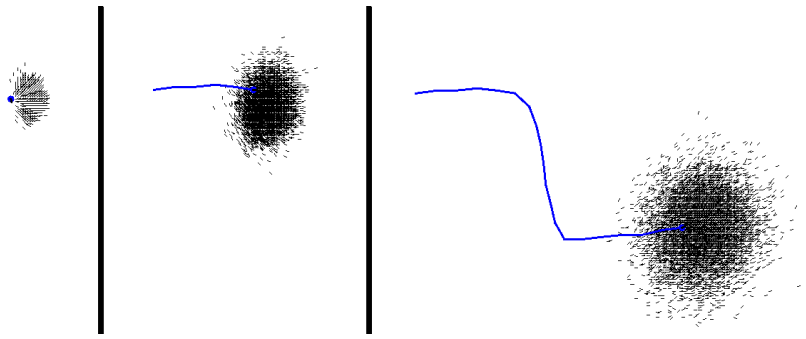


Abbildung 4.3: Divergenz der Partikelwolke ohne Mapmatching nach 1, 4 und 15 Schritten.

An dieser Stelle wird für den Partikelfilter ein effizientes Mapmatching Verfahren auf Basis des in Abschnitt 3.3 eingeführten Bitmap-basierten Gebäudemodells vorgestellt. Da

für das Mapmatching vor allem Kollisionsanfragen relevant sind, eignet sich dieses Modell am besten. Schwarze Pixel repräsentieren Kollisionsgeometrie, die nicht durchdrungen werden kann, weiße Pixel kennzeichnen begehbare Bereiche und Pixel in bestimmten Markerfarben Übergänge zwischen Stockwerken, wie Treppen oder Aufzüge. Eine gelbe Markerfarbe markiert einen möglichen Wechsel zum darunterliegenden Stockwerk, eine blaue Farbe einen Wechsel ins darüberliegende Stockwerk und grün schließlich einen Bereich, bei dem sowohl nach oben als auch nach unten gewechselt werden kann. Zudem ist zu beachten, dass alle Türen mithilfe des ebenfalls in Abschnitt 3.3 vorgestellten Türerkennungsalgorithmus entfernt werden, damit eine durchgehende begehbare Fläche entsteht, die der tatsächlichen Erreichbarkeit von Gebieten in der realen Welt entspricht.

Trifft ein Partikel bei der Bewegung auf ein schwarzes Pixel, so stirb es. Die durch das Partikel repräsentierte Hypothese war offensichtlich falsch, da der Nutzer die Kollisionsgeometrie nicht durchschritten haben kann. Trifft ein Partikel auf ein gelbes Pixel, so wird es nach einer gewissen Anzahl an Schritten im Bereich gelber Pixel mit einer bestimmten Wahrscheinlichkeit auf das darunterliegende Stockwerk gesetzt. Weitere nachfolgende Stockwerkswechsel sind erst nach derselben bestimmten Anzahl an Schritten möglich. Da Treppen für einen Stockwerkswechsel im Normalfall mindestens 13 Stufen aufweisen und selten mehr als zwei Stufen auf einmal genommen werden, wird diese Anzahl auf 7 Schritte festgelegt. Nach den 7 Schritten wird das Partikel mit einer Wahrscheinlichkeit von 50 % auf das angrenzende Stockwerk versetzt, mit 50 % verbleibt es im ursprünglichen Stockwerk. Selbiges gilt für ein blaues Pixel. Bei einem grünen Pixel werden die Partikel mit gleicher Wahrscheinlichkeit zufällig nach oben oder unten versetzt. Überquert ein Partikel auf seinem Weg nur weiße Pixel, so tritt keinerlei Sonderbehandlung ein. Die Art der überquerten Pixel je Schritt wird auf der Bitmap durch Bresenhams Algorithmus überprüft.

Verfügt das Smartphone zusätzlich zu den vorhandenen Sensoren über ein Barometer, so kann dieses genutzt werden, um Höhenunterschiede zu erkennen und somit genauer auf Stockwerkswechsel zu schließen. Auch wenn sich keine absolute Höhe aus den Werten berechnen lässt, da der Luftdruck zudem von der Temperatur, dem Wetter und in Gebäuden von der Lüftungsanlage abhängt, so lassen sich doch zumindest Höhenunterschiede aus der internationalen Höhenformel für den Druck p in Abhängigkeit der Höhe h ermitteln:

$$p(h) = 1013,25 \left(1 - \frac{0,0065h}{288,15}\right)^{5,255} \text{hPa} \tag{4.10}$$

Auf Basis der Druckunterschiede lässt sich so mithilfe von Gleichung (4.10) ein ungefährer Wert für den Höhenunterschied ermitteln. Sind mehr als 2 Meter Höhenunterschied gemessen worden, so kann man von einem Stockwerkswechsel ausgehen. Im Fall des in Abschnitt 3.2 vorgestellten Umgebungsmodells wird die Stockwerkshöhe sogar explizit angegeben.

4.2.4 Testergebnisse

Der Partikelfilter wurde in einer Büroumgebung in einem Gebäude der Ludwig-Maximilians-Universität München evaluiert. Dazu wurden auf verschiedenen Pfaden Sensormessungen

aufgenommen und zur Positionsbestimmung genutzt. Die tatsächlichen Referenzpositionen wurden durch Bodenmarker in Verbindung mit manuell erzeugten Zeitstempeln beim Passieren der Marker aufgezeichnet. Zwischen zwei Markern wurde eine konstante Geschwindigkeit angenommen.

Die Pfade wurden mit zwei unterschiedlichen Initialisierungsmöglichkeiten evaluiert. Im ersten Fall wird von einer initialen Gleichverteilung über das ganze Gebiet ausgegangen, im zweiten Fall von einer bekannten initialen Position an einem bestimmten Punkt in der Testumgebung.

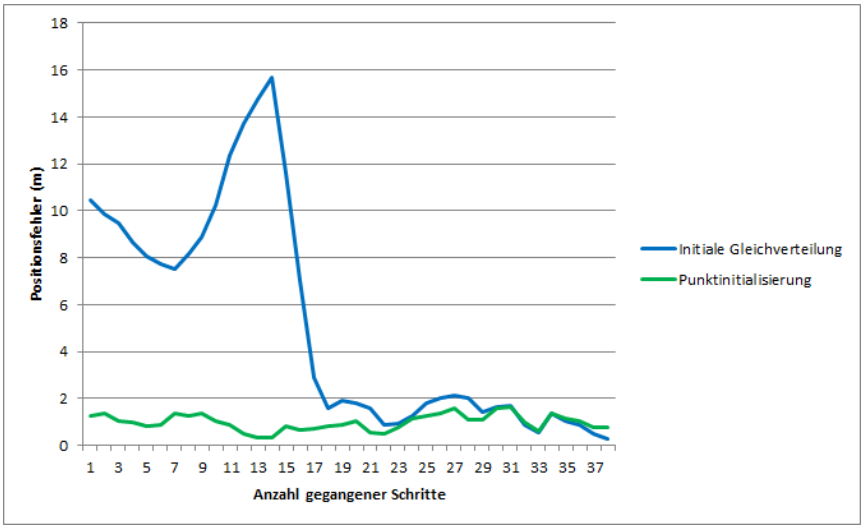


Abbildung 4.4: Genauigkeit des Partikelfilters bei initialer Gleichverteilung im Vergleich zur Initialisierung an einem Punkt im in Abbildung 4.5 gezeigten Testpfad.

Abbildung 4.4 zeigt die unterschiedlichen Genauigkeiten des Partikelfilters in Abhängigkeit von der Initialisierung bei einem der Testpfade. Erst nachdem die Partikelwolke konvergiert ist, ist die Genauigkeit bei der initialen Gleichverteilung vergleichbar mit der erreichbaren Genauigkeit bei bekanntem Startpunkt und liegt kontinuierlich unterhalb der geforderten 2 Meter. Abbildung 4.5 zeigt die Konvergenz des Partikelfilters über die Zeit.

Die Ergebnisse konnten auf weiteren Pfaden bestätigt werden, eine Konvergenz konnte in den meisten Fällen nach 15 bis 40 Schritten erreicht werden. In einem Fall konnte jedoch über eine Strecke von knapp 50 Metern keine Konvergenz erzielt werden. Allerdings beziehen sich die Versuche nur auf ein Stockwerk. Aufgrund der Symmetrie benachbarter Stockwerke ist eine Einschränkung auf ein einzelnes Stockwerk allein anhand von Mapmatching kaum möglich. Lediglich in Fällen, in denen der Nutzer Stockwerkswechsel vollführt, kann eine Konvergenz in einem einzelnen Stockwerk erreicht werden.

- Die Anforderungen an die **Genauigkeit und Präzision** der Positionsbestimmung werden bei initial unbekannter Position erst nach einer Konvergenz der Partikelwolke erreicht.

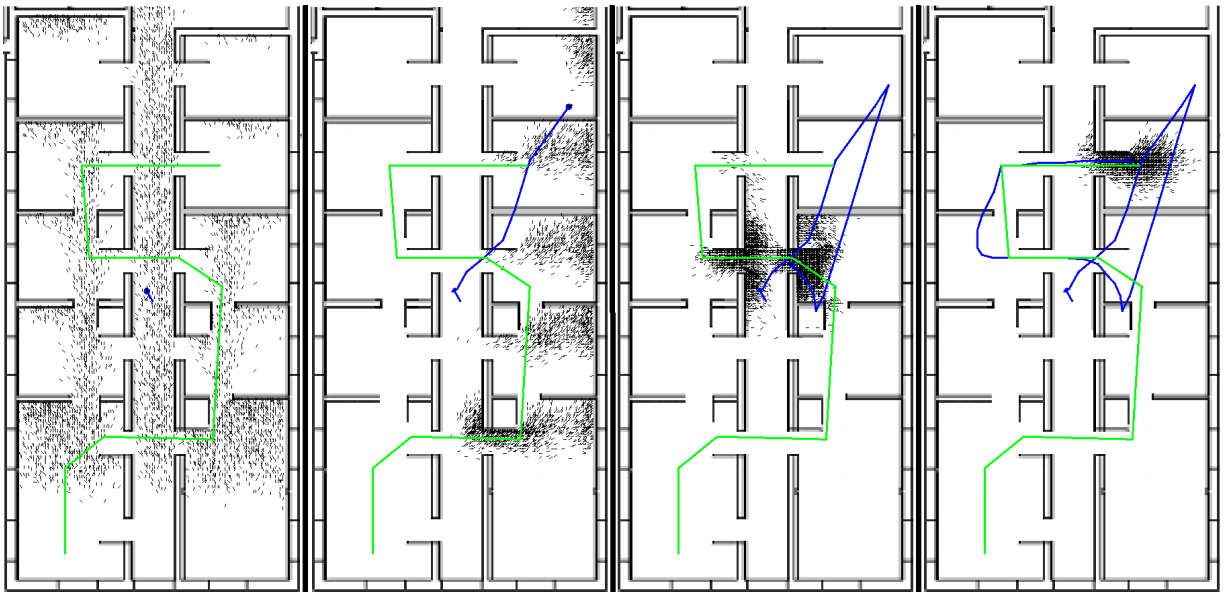


Abbildung 4.5: Von links nach rechts: Konvergenz des Partikelfilters bei initialer Gleichverteilung über die Zeit. Geschätzter Pfad des Partikelfilters(blau), tatsächlich zurückgelegte Strecke (grün) und Partikelwolke (schwarze Striche).

- Eine **Abschätzung von Genauigkeit und Präzision** kann durch die Standardabweichung der Partikelwolke gegeben werden.
- Die Positionsbestimmung ist ohne Einschränkungen an bestimmte Umgebungen überall möglich, wo Kartenmaterial in Form einer Bitmap, die den Anforderungen aus Abschnitt 3.3 genügt, existiert.
- Die Positionsbestimmung ist aufgrund der aufwendigen Berechnungen des Partikelfilters nicht **Energie-effizient**.
- Das System benötigt keine zusätzliche Hardware und verursacht keine Kosten für Inbetriebnahme und Betrieb.
- Durch die Endgeräte-zentrische Positionsbestimmung ist die **Privatsphäre** des Nutzers geschützt.
- Das System ist **ausfallsicher** und kann durch die Neuinitialisierung der Partikel flexibel auf Änderungen in der Umgebung reagieren.
- Das System stellt Positionsinformationen mit einer genügend hohen Frequenz in Abhängigkeit von der Schrittfrequenz mit bis zu 5 Hz zu Verfügung.

Die Positionsbestimmung mithilfe des infrastrukturlosen Partikelfilters erfüllt also in den meisten Fällen bereits die Anforderungen aus Abschnitt 2.5.4. Allerdings gibt es neben dem hohen Energieverbrauch zwei Probleme, die durch Hinzunahme von absoluten Positionsbestimmungsverfahren zu lösen sind:

- Bei initial unbekannter Startposition im Gebäude muss erst eine Konvergenz der Partikel durch Mapmatching erreicht werden, bevor eine genügend genaue Position für viele Dienste bereit steht. Gerade bei Gebäudesymmetrien, auch in Bezug auf übereinanderliegende Stockwerke, kann hier einige Zeit vergehen oder in einigen Fällen keine Konvergenz herbeigeführt werden, was durch die Ergebnisse in [146] bestätigt wird.
- Existiert nur wenig Kollisionsgeometrie, wie beispielsweise in größeren Räumen oder Hallen, so divergiert die Partikelwolke aufgrund der möglichen Aufsummierung von Fehlern in der relativen Positionsbestimmung. Auch dadurch kann die Dienstnutzung vorübergehend eingeschränkt werden, da die Positionsgenauigkeit sinkt.

Im Folgenden werden daher verschiedene Verfahren zur absoluten Positionsbestimmung vorgestellt und verbessert, die mit möglichst wenig Kosten und Kalibrierungsaufwand mithilfe von bestehender Infrastruktur als Messmodell in den Partikelfilter integriert werden können.

4.3 Fingerprinting als Messmodell zur absoluten Positionsbestimmung

Im letzten Abschnitt hat sich gezeigt, dass Systeme, die rein auf Dead Reckoning mit aktuellen Smartphones basieren, noch mit zwei Problemen kämpfen. Die initial unbekannte Startposition und die Divergenz auf Freiflächen. In diesem Abschnitt werden daher Methoden zur absoluten Positionsbestimmung durch Fingerprinting untersucht und weiterentwickelt, um eine Alternative zur Koppelnavigation zu bieten. Fingerprinting kann im Gegensatz zu anderen Verfahren mit bestehender Hardware durchgeführt werden. Andere Techniken zur absoluten Positionsbestimmung, wie Triangulation, benötigen entweder eine spezielle Infrastruktur oder verbesserte Hardware, um die erforderliche Genauigkeit zu erreichen.

Bereits in Abschnitt 2.1 wurde Fingerprinting als allgemeine Möglichkeit zur Positionsbestimmung vorgestellt. In diesem Abschnitt wird detaillierter auf den Einsatz von Fingerprinting zur Kamera-Selbst-Ortung und zur Ortung mit WLAN eingegangen und insbesondere eigene Verbesserungen hervorgehoben. Zudem wird hier eine Bewertung der einzelnen Systeme entsprechend der Anforderungen gegeben.

4.3.1 Kamera-Selbst-Ortung

Bei der Positionsbestimmung mithilfe von Kameras gibt es die verschiedensten Ansätze, die von stationären Kameras mit Personenerkennung bis zur relativen Selbst-Ortung durch optischen Fluss reichen (vgl. Abschnitt 2.3.2). Im Folgenden wird vor allem auf die Kamera-Selbst-Ortung eingegangen und ein eigener Ansatz zur Positionsbestimmung mit Smartphone-Kameras vorgestellt. In der Literatur werden in solch einem Fall oftmals Bildtransformationen eingesetzt, um bestimmte visuelle Eigenschaften eines Bildes zu extrahieren. Kanten-

und Eckpunkterkennung (vgl. [21] bzw. [48]) werden dazu genutzt, eine einfachere aber dennoch wiedererkennbare Version eines Bildes zu erzeugen. Zudem gibt es Ansätze, um lokal hervorstechende Punkte, sogenannte Feature-Punkte zu erkennen, die möglichst Rotations- und Skalen-invariant beschrieben werden [82, 13]. Dabei ist vor allem letzterer Ansatz robust gegenüber unterschiedlichen Blickwinkeln und geringfügigen Veränderungen im Bild, benötigt allerdings deutlich mehr Rechenkapazität, als beispielsweise Kantenerkennung.

Bestehende Kamera-basierte Positionsbestimmungssysteme können beide Arten von Bildtransformationen nutzen, um auf die Position der Kamera zu schließen. Hile und Borriello extrahieren in [52] die Kanten eines Bildes und versuchen, aus diesen aufgrund bestimmter Eigenschaften Türstöcke zu erkennen. Diese werden gezählt und mit einem Gebäudeplan verglichen. Dazu wird von einer aus einem WLAN-Ortungssystem grob bekannten Position ausgegangen, um das Problem der Selbstähnlichkeit von Gebäudeteilen oder Korridoren zu vermeiden. Dieser Ansatz hat jedoch laut der Autoren Probleme in großen Räumen oder bei durch Einrichtungsgegenstände verdeckten Türstöcken. Kawaji et al. nutzen in [58] natürliche Feature-Punkte und omnidirektionale Bilder, um die Position in einem Museum zu bestimmen. Die Feature-Punkte werden mit PCA-SIFT, einer schnelleren Variante des SIFT-Algorithmus, extrahiert und gespeichert. Zur Positionsbestimmung werden die ebenfalls extrahierten Feature-Punkte mit den gespeicherten verglichen und die Position des ähnlichsten Referenzbildes als eigene Position angenommen.

In [138] wurde ein Fingerprint-basierter Ansatz zu Selbst-Ortung namens MOVIPS (Mobile Visual Indoor Positioning System) entwickelt, der mit handelsüblichen Smartphone-Kameras und einer Distanzschätzung zum Motiv arbeitet. MOVIPS kombiniert dabei ähnlich zu [52] WLAN-Fingerprinting mit Fingerprinting auf Bilddaten, wobei die mit WLAN ermittelte Position hauptsächlich zur groben Positionsbestimmung und der damit verbundenen Reduktion des Suchraums für die Kamerabilder genutzt wird. Zur Kamera Selbst-Ortung wird in der Kalibrierungsphase eine Datenbank von Bildern zusammen mit den jeweiligen Aufnahmepunkten erzeugt und in der Ortungsphase anhand eines Ähnlichkeitsvergleichs des aktuellen Kamerabilds mit den Datenbankbildern die exakte Position und Blickrichtung des Nutzers bestimmt. Dazu wird aus den Bildern mit Hilfe des SURF-Algorithmus [13] eine Anzahl an skalen- und rotationsinvarianten Feature-Punkten extrahiert, die einen hohen Wiedererkennungswert haben. In der Phase der Positionsbestimmung können nun auf aktuellen Kamerabildern nach demselben Prinzip ebenfalls Feature-Punkte extrahiert werden und mit den in der Datenbank gespeicherten Punkten verglichen werden. Der Ort, an dem das Datenbankbild mit der größten Ähnlichkeit in Bezug auf die Features aufgenommen wurde, entspricht mit hoher Wahrscheinlichkeit annähernd dem aktuellen Aufenthaltsort des Nutzers. Da die Features skaleninvariant sind, kann ein Distanzkorrekturverfahren angewendet werden. Dieser Ansatz wird im Folgenden genauer vorgestellt.

Erkennung von Feature-Punkten

Zum besseren Verständnis der Eigenschaften des Systems wird zunächst die Funktionsweise der Feature-Erzeugung am Beispiel des SIFT-Algorithmus [82] erklärt: Hierbei wird ein Bild in Grautönen durch Hinzufügen von Gauss'scher Unschärfe verschiedener Stärke in

mehrere Skalen abgebildet, wodurch ein Skalenraum des Bildes entsteht. In diesen Skalen wird nach lokalen Extrema gesucht, die nicht nur bezüglich einer einzelnen Skala, sondern auch bezüglich der Nachbarskalen minimal oder maximal sind. Diese Extrema stellen die skaleninvarianten Feature-Punkte dar. Die Rotationsinvarianz wird durch die Beschreibung eines Feature-Punktes durch lokale Gradienten erreicht, welche für die Grid-Zellen eines um den Feature-Punkt aufgespannten Rasters berechnet werden. Eine anschließende Normalisierung verschafft zudem eine gewisse Robustheit gegenüber Beleuchtungswechseln. SURF [13] folgt generell dem gleichen Vorgehen, verwendet jedoch schnellere Approximationen und weniger Grid-Zellen und ist daher schneller berechenbar und platzsparender.

Ähnlichkeitsvergleich zweier Bilder

Der Ähnlichkeitsvergleich zweier Bilder wird nun auf Basis der Feature-Punkte durchgeführt. Dabei stimmen zwei Feature-Punkte auf unterschiedlichen Bildern überein, wenn diese die kleinste Euklidische Distanz zwischen allen Kandidaten haben und zudem der Abstand mindesten 20 % kleiner ist, als der Abstand zum zweitbesten Kandidaten. Die Anzahl der übereinstimmenden Feature-Punkte ist ein gutes Maß für die Ähnlichkeit zweier Bilder. Allerding ist zu beachten, dass dieses Maß nicht symmetrisch ist. In MOVIPS [138] wird zur Positionsbestimmung das aktuelle Kamerabild mit allen Bildern aus der Datenbank verglichen und die Ähnlichkeit entsprechend des eben beschriebenen Algorithmus in beide Richtungen berechnet. Das bedeutet, dass sowohl die Anzahl der Matches des aktuellen Bildes mit jedem Bild aus der Datenbank, als auch die Anzahl der Matches jedes Bildes der Datenbank mit dem aktuellen Bild berechnet wird. Von den beiden Bildern mit der so ermittelten größten Ähnlichkeit in Form der Summe der Matches aus beiden Richtungen wird nun das Bild aus der Datenbank zur Ermittlung des Aufenthaltsortes herangezogen, das den geringeren Abstand zwischen der Anzahl der Matches aus jeweils einer Richtung aufweist.

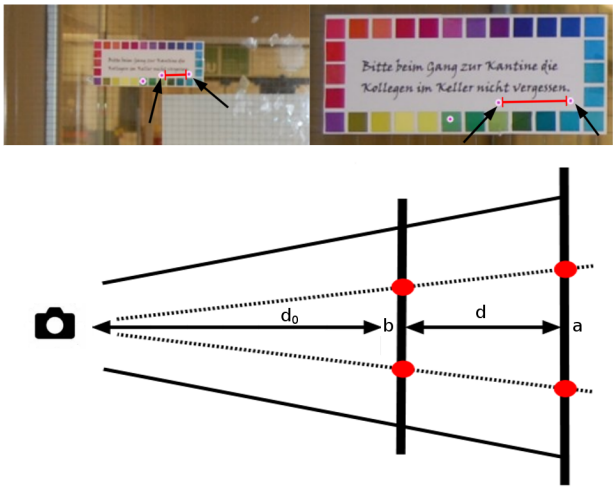
Schätzung des Abstands zum Motiv

Zu diesem Zeitpunkt ist das Referenzbild aus der Datenbank ermittelt worden, das mit größter Wahrscheinlichkeit dem aktuellen Motiv entspricht. Allerding ist der Abstand zwischen Kamera und Motiv nicht zwingendermaßen gleich dem Abstand bei der Erzeugung des Referenzbildes. Daher kann die aktuelle Position nicht mit der Referenzposition bei Aufnahme des Bildes gleichgesetzt werden. Wäre an dieser Stelle der Abstand zum Motiv auf dem Referenzbild bekannt, so könnte mittels Strahlensatz der aktuelle Abstand zum Motiv ermittelt werden (vergleiche Abbildung 4.6):

$$\frac{d_0 + d}{d_0} = \frac{b}{a} \Leftrightarrow d = \left(\frac{a_p}{b_p} - 1\right) d_0 \tag{4.11}$$

Hierbei entspricht d_0 dem Abstand des Referenzbildes zum Motiv, d dem Abstand zwischen Referenzbild und aktuellem Kamerabild, b dem Abstand zwischen zwei Punkten im Motiv, die das aktuelle Kamerabild links bzw. rechts begrenzen, und a dem Abstand

(a) Screenshots der mobilen Anwendung.



(b) Anwendung des Strahlensatzes auf Kamerabilder mit unterschiedlichem Abstand zum Motiv.

Abbildung 4.6: Screenshots der mobilen Anwendung, welche die Positions- und Richtungsschätzung zeigen, sowie das erkannte Referenzbild mit der Positionsinformation der Datenbank. Zudem ist der Algorithmus zur Abstandsschätzung auf Basis der Pixelabstände gematchter Punkte dargestellt. [138]

zwischen zwei Punkten im Motiv, die das Referenzbild links bzw. rechts begrenzen. Da b/a jedoch nicht bekannt ist, wird der Wert durch das Verhältnis des Pixel-Abstands a_p/b_p zwischen jeweils zwei gleichen Punkten auf den Bildern ersetzt, welches konstant dem gleichen Wert wie b/a entspricht. Somit lässt sich d, wie in Gleichung (4.11) angegeben, berechnen.

Um die Komplexität der Erzeugung von Referenzdaten so gering wie möglich zu halten, wird d_0, der Abstand des Referenzpunktes zum Motiv, in MOVIPS [138] bei der Erzeugung der Referenzdatenbank nicht gespeichert. Um dennoch eine Annäherung für den tatsächlichen Abstand zum Motiv zu erhalten, wird hier eine Annäherung in Form einer linearen Regression durchgeführt.

$$d = \alpha \frac{a_p}{b_p} \tag{4.12}$$

Dazu wird in Abhängigkeit der zur Ortung genutzten Kamera mit Hilfe mehrerer Bilder in bekannten Abständen der Koeffizient α kalibriert. Auch wenn dabei ein Fehler in Kauf genommen wird, beispielsweise ein Fehler von α bei aktuellem Aufenthalt an der Referenzposition, so ist doch eine genauere Positionsbestimmung möglich, als wenn $d = 0$ für alle Fälle angenommen wird.

So bleibt zur Abstandsschätzung die Ermittlung des Verhältnisses von a_p/b_p. Dieses lässt sich in der Theorie leicht durch Paare von korrespondierenden Feature-Punkten in beiden Bildern ermitteln. Allerdings kann es beim Matching vorkommen, dass falsche Punk-

te einander zugeordnet werden, wodurch die Abstandsschätzung von fehlerhaften Angaben ausgeht und zu großen Fehlern führen kann. Daher werden nicht einzelne Punktpaare verglichen, sondern der mittlere Abstand zweier Punkte über alle übereinstimmenden Punkte hinweg. In [138] wurde festgestellt, dass das harmonische Mittel den Abstand unterschätzt und der Mittelwert den Abstand überschätzt. Daher wird der Durchschnitt beider Werte für den mittleren Punktabstand je Bild genommen und für a_p bzw. b_p verwendet.

Evaluation

Neben dem Einsatz des beschriebenen Algorithmus zur Positionsbestimmung mittels Kamerabildern wurde in MOVIPS [138] ein Variante entwickelt, die auf Basis eines Video-Streams kontinuierliche Ortung unterstützt. Hierbei werden alle Frames in einem Video niedriger Auflösung entsprechend dem Algorithmus untersucht. Da durch die niedrige Auflösung und die Bewegungsunschärfe jedoch einige Unsicherheiten eingeführt werden, wird eine zusätzliche Korrektur über die Zeit eingeführt. Dabei wurden zwei Alterativen untersucht: die erste Version mittelt alle geschätzten Positionen über eine gewisse Zeitspanne hinweg, während die zweite Version nur die korrigierten Positionen des Bildes zur Mittelung heranzieht, die innerhalb der festgelegten Zeitspanne am häufigsten als Referenzbild erkannt wurden.

Zur Evaluation wurde das Verfahren in einem knapp 25 Meter langen Gang mit gleichartiger Struktur getestet. Dazu wurden 68 Bilder aufgenommen und zusammen mit der Aufnahmeposition und Blickrichtung in einer Datenbank gespeichert. Zudem wurden aus den Bildern die Feature-Punkte extrahiert und ebenfalls zu jedem Bild gespeichert. Diese initiale Kalibrierung benötigte ca. 10 Minuten für die Aufnahme der Bilder und 5 Minuten für die Berechnung der Feature-Punkte. Der Speicherbedarf bei einer Auflösung von 2560x1920 Pixeln und einem SURF-Grenzwert von 0.0004 betrug 134,3 MB für die Bilder und 32,2 MB für die Feature-Punkte.

Für die Evaluation der *Genauigkeit* und *Präzision* der Positionsbestimmung wurden weitere 17 Bilder an unterschiedlichen Stellen im Gang mit der Smartphone-Kamera aufgenommen und die geschätzte Position mit der tatsächlichen Position verglichen. Die beiden Varianten, die mit einem Video-Stream arbeiten, wurden auf Basis eines Videos weniger Sekunden Länge mit 640x480 Pixeln Auflösung evaluiert, die Zeitspanne wurde hier auf 500 ms, was ca. 15 Frames entspricht, festgelegt. Die Ergebnisse sind in Tabelle 4.1 festgehalten.

System	**25% [m]**	**50% [m]**	**75% [m]**
Photo Modus (Stationär)	0.28	0.68	1.25
Video Modus (Durchschnitt)	2.39	3.88	4.4
Video Modus (Auswahl & Durchschnitt)	1.0	2.85	4.4

Tabelle 4.1: Genauigkeit und Präzision anhand dreier Quantile der kumulativen Verteilungsfunktion. [138]

Die Bewertung bezüglich der weiteren Metriken fällt bei MOVIPS folgendermaßen aus:

- Die *Abdeckung*, die durch MOVIPS erreicht wird, hängt in erster Linie von dem Gebiet ab, in dem das System kalibriert wird. Bei entsprechendem Aufwand kann eine 100%-ige Abdeckung eines Gebäudes erreicht werden. Hierbei ist jedoch zu beachten, dass das System auf eine genügende Ausleuchtung der Umgebung angewiesen ist und in manchen Umgebungen das Aufnehmen von Bildmaterial nicht gestattet ist.

- Die *Anschaffungskosten* sind gering, es wird lediglich eine Serverkomponente zur Verwaltung der Datenbank und Berechnung von Positionsanfragen benötigt. Der große Aufwand einer initialen Kalibrierung gehört allerdings ebenso zu den bei Installation anfallenden Kosten.

- Die *Betriebskosten* entstehen durch den Betrieb der Serverkomponente und etwaigen Nachkalibrierungen. Diese müssen jedoch nur in Fällen von visuellen Änderungen lokal durchgeführt werden.

- Außer einer Serverkomponente ist eine Kommunikations*infrastruktur* notwendig, damit Anfragen vom Smartphone an den Server gestellt werden können. Aufgrund der großen zu übertragenden Datenmenge ist eine für den Nutzer kostenneutrale Lösung, wie WLAN, einer möglicherweise kostenintensiven Mobilfunkanbindung vorzuziehen.

- Die *unterstützte Nutzerzahl* wird lediglich durch die Kapazität der Serverkomponente beschränkt und lässt sich gut skalieren.

- Der *Ergebnistyp* entspricht einer absoluten, geometrischen und deterministischen Positionsangabe.

- Die *Verfügbarkeit* ist gegeben, da heutige Smartphones grundsätzlich mit einer Kamera ausgestattet sind.

- Das System weist noch keine große *Zuverlässigkeit* auf, da weder mit unbekannten noch mit fehlenden Daten umgegangen werden kann. Diese resultieren in einer falschen oder keiner Positionsangabe. Lediglich gestörte Daten, wie teilweise Verdeckungen oder Bildunschärfe, können bis zu einem gewissen Grad kompensiert werden.

- Die *Update-Rate* im stationären Modus ergibt sich aus ca. 3 Sekunden für die Positionsbestimmung zuzüglich der Zeit für die Datenübertragung des Kamerabildes an den Server. Im Videomodus ist die Update-Rate etwa 500 Milisekunden.

- Ein *Schutz der Privatsphäre* ist nicht gegeben, da das System aufgrund des Speicher- und Rechenbedarfs gerade in großen Gebäuden kaum auf dem Smartphone alleine laufen kann. Zudem werden Kamerabilder zur Positionsbestimmung genutzt, wodurch möglicherweise das Persönlichkeitsrecht von Passanten verletzt wird.

Die Positionsbestimmung mit der Smartphone-Kamera erfüllt somit zwar schon einige der Anforderungen ortsbezogener Dienste, allerdings ist eine Positionsbestimmung immer

nur nach aktiver Nutzeraktion möglich, nämlich dem Schießen eines Fotos. Dadurch werden bestimmte Dienste, wie z.B. Friendfinder, nur eingeschränkt unterstützt. Einer der größten Vorteile ist zwar die Unabhängigkeit von jeglicher Positionierungsinfrastruktur, allerdings ist eine zusätzliche grobe Positionsbestimmung zur Einschränkung der Bilddatenbank empfehlenswert, um die Antwortzeit gering zu halten.

4.3.2 Grundlagen zum WLAN-Fingerprinting

Neben der Kamera, den Erdeffekt- und Inertialsensoren können auch die Funkkomponenten des Smartphones zur Positionsbestimmung genutzt werden. Da Satellitensignale in Gebäuden für die Empfänger im Smartphone meist zu schwach sind und die GPS-Genauigkeit in Innenbereichen für viele Dienste nicht ausreicht, werden andere Funktechniken, wie WLAN oder Mobilfunk zur Positionsbestimmung verwendet. Sowohl bei WLAN als auch bei Mobilfunk dient die Infrastruktur der Kommunikation und ist somit oftmals bereits vorhanden. Dabei sind WLAN Access Points direkt im Gebäude angebracht, während Mobilfunkstationen eher in Außenbereichen und auf Dächern montiert sind. Im Folgenden wird die Positionsbestimmung durch WLAN-Fingerprinting analysiert und verbessert, wobei sich viele Erkenntnisse auch auf den Mobilfunk übertragen lassen, solange eine vergleichbare Dichte von Referenzstationen vorhanden ist.

Auch WLAN-Fingerprinting Systeme können sich in verschiedener Hinsicht voneinander unterscheiden. Einen großen Einfluss hat insbesondere der Algorithmus zur Positionsbestimmung. Hier sind die beiden bekanntesten Ansätze k-Nächste-Nachbarn (kNN) [10, 54, 64] und Bayesisches Fingerprinting [148, 68]. Prinzipiell ist fast jeder Algorithmus aus dem Bereich des maschinellen Lernens geeignet, der anhand von Trainingsdaten eine Zustandserkennung zulässt. Weitere Unterschiede treten bei der Organisation und Erstellung der Trainingsdaten auf. So existieren Systeme, in denen jeder Fingerprint einem einzelnen Referenzpunkt, einem Referenzgebiet oder einem symbolischen Namen zugeordnet wird. Der Name kann z.B. für den Raum stehen kann, in dem der Fingerprint gemessen wurde. Die Trainingsdatenbank kann von einem Administrator oder von den Nutzern selbst erzeugt werden und schließlich lokal auf dem Endgerät oder auf einem Server bereitgestellt werden. Je nach Einsatzgebiet sollte das Design vorsichtig ausgewählt werden. Im Folgenden wird erst allgemein das Vorgehen erläutert und anschließend werden ein paar spezielle Formen des WLAN-Fingerprintings genauer vorgestellt und evaluiert.

WLAN-Fingerprinting teilt sich wie alle Fingerprinting-Ansätze in zwei Phasen. In der Offline-Phase wird eine Trainingsdatenbank bezüglich eines Referenzsystems erstellt. In der Datenbank werden Ortsinformationen mit Signalstärkeinformationen in einem sogenannten Fingerprint verknüpft. Die Signalstärke wird durch den RSSI (Received Signal Strength Indicator) angegeben, die MAC-Adresse liefert eine eindeutige Identifikation des sendenden Access Point. Die Datenbank kann entweder bei bekannter Position aller Access Points mit Hilfe eines Ausbreitungsmodells gewonnen oder durch empirischen Messungen an bestimmten Orten im Gebäude ermittelt werden. Da die letztere Methode im allgemeinen für genauere Ortungsergebnisse sorgt [10], wird an dieser Stelle nicht weiter auf Ausbreitungsmodelle eingegangen, auch wenn diese aufgrund des geringeren zeitlichen Aufwands

weniger Kosten verursachen. Die Genauigkeit von Fingerprinting Ansätzen hängt im Fall empirischer Messungen von der Aktualität der Messung, der Dichte der Referenzpunkte, der Anzahl der Messungen je Referenzpunkt und den Ausbreitungscharakteristiken des Gebäudes ab. Dabei gilt, dass eine höhere Aktualität, größere Dichte oder höhere Anzahl an Messungen zwar zu besseren Ortungsergebnissen führt, allerdings ebenso einen größeren zeitlichen Aufwand, also höhere Investitions- oder Instandhaltungskosten, bedeutet. Zudem existiert nach [15] eine maximal zu erwartende Genauigkeit der Positionsbestimmung, die sich direkt aus der Datenbasis herleiten lässt.

kNN

Im Folgenden wird davon ausgegangen, dass die Referenzpunkte zu jedem Fingerprint in Form von geometrischen Koordinaten bezüglich eines beliebigen Umgebungsmodells angegeben sind. Am Ende des Abschnitts werden Variationsmöglichkeiten des Algorithmus angegeben, wenn die Positionsbestimmung auf symbolischen Koordinaten bestimmt wird.

Beim kNN Algorithmus wird die aktuelle WLAN-Signalstärkemessung m mit allen Messungen $\{f_i\}_{i=1}^{n}$ aus der Datenbank auf Ähnlichkeit verglichen. Die Ähnlichkeit wird als Distanzmaß auf dem Vektor der Signalstärken der Messung m.RSSI und der Fingerprints f_i.RSSI realisiert. Dabei kommen verschiedene Maße, wie der Euklidische Abstand oder der Manhattan Abstand, zum Einsatz (siehe Gleichung (4.13)). Die einzelnen Dimensionen werden dabei durch die MAC-Adressen der empfangenen Access Points festgelegt. $|f_i|$ bzw. $|m|$ kennzeichnen jeweils die Dimension des Signalstärke-Vektors des i-ten Fingerprints f_i bzw. der aktuellen Messung m. $f_i.\mathrm{MAC}_j$ kennzeichnet die j-te MAC-Adresse im i-ten Fingerprint.

$$\hat{d}_i = \sqrt{\sum_{j=1}^{|f_i|} \sum_{h=1}^{|m|} \delta_{f_i.\mathrm{MAC}_j, m.\mathrm{MAC}_h} (f_i.\mathrm{RSSI}_j - m.\mathrm{RSSI}_h)^2} \tag{4.13}$$

Da meist nicht jeder Access Point an jedem Ort im Gebäude sichtbar ist, muss die Information nicht übereinstimmender Vektoren oder Einträge in die Distanzberechnung d_i mit einbezogen werden. Dies wird dadurch erreicht, dass die Distanz $\hat{d}_i$ übereinstimmender Vektoren mit einer Strafentfernung d_i^* je nicht übereinstimmendem Eintrag verrechnet wird (siehe Gleichung (4.14)):

$$d_i = \hat{d}_i + d_i^* \left(|f_i| + |m| - 2 \sum_{j=1}^{|f_i|} \sum_{h=1}^{|m|} \delta_{f_i.\mathrm{MAC}_j, m.\mathrm{MAC}_h} \right) \tag{4.14}$$

Ist der Abstand d_i zu jedem Fingerprint f_i bestimmt, so wird bei kNN die Referenzposition p_{est} als Mittel der Referenzpositionen $\{f_i.p\}_{i \in K}$ aus der Menge K der k nächsten Fingerprints in der Datenbank angegeben. Dabei ist nicht nur eine Variation des Parameters k möglich, sondern es kann neben der gleichmäßigen Gewichtung aller k nächsten

Nachbarn mit zusätzlich eine normierte Gewichtung w_i entsprechend der Ähnlichkeit der einzelnen Nachbarn angegeben werden.

$$p_{\text{est}} = \sum_{i \in K} w_i f_i.p \tag{4.15}$$

Dahinter steckt die Überlegung, je ähnlicher ein Fingerprint aus der Datenbank ist, desto wahrscheinlicher ist der Aufenthalt am Ort der Aufnahme, also der Referenzposition des Fingerprints. Als Gewicht eignet sich somit das normierte Inverse des Abstands:

$$w_i = \left(d_i \sum_{j=1}^{k} \frac{1}{d_j} \right)^{-1} \tag{4.16}$$

Es ist leicht zu sehen, dass dieses Gewicht normiert ist, da $\sum_{i=1}^{k} w_i = 1$ gilt. Die Position wird nun als das gewichtete Mittel der k Referenzpositionen der nächsten Fingerprints angenommen.

Da kNN die Varianz der Signalstärke an einem bestimmten Ort nicht in die Positionsbestimmung mit einbezieht, werden meist sowohl für die Referenzdatenbank, als auch für die aktuelle Messung die Mittelwerte der RSSI mehrerer Messungen hergenommen. Dies gilt natürlich insbesondere dann, wenn anstelle von geometrischen Koordinaten symbolische Bezeichner als Referenzposition angegeben werden. Hier lässt sich der Abstand wie oben bestimmen, allerdings ist eine Mittelung von den k nächsten symbolischen Bezeichnern nicht wohldefiniert. Daher wird entweder der nächste Nachbar als Position angenommen (1NN) oder eine Wahrscheinlichkeit für jeden Bezeichner entsprechend der Anzahl der jeweiligen Treffer in den nächsten Nachbarn oder bezüglich des obigen Gewichtes der nächsten Nachbarn angegeben.

Bayes

Bayesische Verfahren stützen sich auf die Regel von Bayes (vgl. Abschnitt 2.2.1) zur Rechnung mit bedingten Wahrscheinlichkeiten (siehe Gleichung (4.17)). Im Kontext der Positionsbestimmung wird Bayes regel wie folgt interpretiert: Die Wahrscheinlichkeit $p(x|m)$, sich an einem Ort x zu befinden, wenn man die Beobachtung m gemacht hat, lässt sich durch die Wahrscheinlichkeit $p(m|x)$, eine Messung m am Ort x gemacht zu haben, ausdrücken:

$$p(x|m) = \frac{p(m|x)p(x)}{p(m)} \tag{4.17}$$

Da die Wahrscheinlichkeit $p(m)$, eine bestimmte Messung m zu machen, konstant ist und die Aufenthaltswahrscheinlichkeit $p(x)$ am Ort x ohne weiteres Wissen als gleichverteilt angenommen werden kann, hängt $p(x|m)$ nur von $p(m|x)$ ab. Genau diese Wahrscheinlichkeit wird durch die Referenzdatenbank abgebildet, solange für jeden Ort x mehrere Messungen vorliegen, aus denen auf die Wahrscheinlichkeitsverteilung geschlossen werden kann.

Aufgrund der verrauschten Messungen und der Überlagerung unterschiedlicher Störungen kann bei WLAN-Signalen an einem festen Ort eine Normalverteilung der Signalstärken angenommen werden [15]. Somit charakterisiert die Dichtefunktion $\rho_{\mu,\sigma}(y)$ aus Gleichung 4.18 die Wahrscheinlichkeit, eine Signalstärke y zu messen, wenn vorher Signalstärken mit Mittelwert μ und Standardabweichung σ gemessen wurden. Letztere werden für jeden Access Point innerhalb eines Fingerprints aus allen zu diesem AP gehörigen Signalstärkemessungen berechnet. Als Alternative kann die Standardabweichung auch mit einem festen Wert initialisiert werden [139].

$$\rho_{\mu,\sigma}(y) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(\mu-y)^2}{2\sigma^2}} \tag{4.18}$$

Nimmt man nun noch die Unabhängigkeit der Signalstärke verschiedener Access Points an, so lässt sich für eine Messung m die Wahrscheinlichkeit berechnen, mit welcher diese am Ort der Referenzposition $f_i.p$ gemacht wurde, indem über alle Wahrscheinlichkeiten einzelner Signalstärken m_j der Messung multipliziert wird und fehlende Werte wie bei kNN mit einem Strafmaß p_{miss} belegt werden:

$$p(m|f_i.p) = \prod_{j \in (m.\text{MAC} \cap f_i.\text{MAC})} \rho_{\mu_j,\sigma_j}(m_j) \cdot p_{\text{miss}}\,|(m.\text{MAC} \setminus f_i.\text{MAC})| \cdot p_{\text{miss}}\,|(f_i.\text{MAC} \setminus m.\text{MAC})| \tag{4.19}$$

Nimmt man $p(f_i.p)$, die Wahrscheinlichkeit des Aufenthalts am Ort $f_i.p$ eines Fingerprints f_i, als gleichverteilt und $p(m)$ als konstant an, so ist die wahrscheinlichste Position durch das Maximum von $p(m|f_i.p)$ gegeben. Analog zu kNN wird jedoch meist die Position nicht an der Stelle des wahrscheinlichsten Fingerprints angenommen, sondern das gewichtete Mittel einer bestimmte Anzahl an nächsten Positionen, gewichtet entsprechend der normierten Wahrscheinlichkeit aus Gleichung (4.20):

$$p(f_i.p|m) \approx \frac{p(m|f_i.p)}{\sum_{j=1}^{|f|} p(m|f_j.p)} \tag{4.20}$$

Bayesisches Fingerprinting auf Raumebene funktioniert ähnlich, mit dem Unterschied dass alle Messungen innerhalb desselben Raumes zur Berechnung der Parameter der Dichtefunktion hinzugezogen werden. Das Ergebnis ist hierbei eine Wahrscheinlichkeit des Aufenthalts zu jedem Raum, die je nach Anwendungsfall weiter verarbeitet werden kann.

4.3.3 Verbesserung von WLAN-Fingerprinting durch Kompass-Daten

Bereits in vergangenen Forschungsarbeiten wurde der große Einfluss der Orientierung des Endgerätes zum Körper des Nutzers auf WLAN-Fingerprinting Ansätze aufgezeigt [10, 56, 68, 22, 64]. Kaemarungsi et al. stellen in [56] eine Minderung der Signalstärke durch den menschlichen Körper von mehr als 9dBm fest. Bahl und Padmanabhan schlagen in [10]

vor, zur Erstellung der Trainingsdatenbank an jeder Referenzposition mehrere Messungen in jede Himmelsrichtung durchzuführen. Dadurch werden charakteristische Abschottungen durch eine größere Menge an Daten kompensiert. King et al. schlagen in [68] als erstes die Kombination mit einem Kompass vor, indem die aktuelle Kompassmessung in die Berechnung der Aufenthaltswahrscheinlichkeit mithilfe von Bayes Regel einbezogen wird. Dazu werden ebenfalls an jeder Referenzposition Messungen in unterschiedlichen Winkeln, typischerweise 45° oder 90° durchgeführt. Dieses Vorgehen wurde in [64] aufgegriffen, wobei die Orientierung hier in Form eines Filters für kNN bei WLAN-Fingerprinting eingesetzt wurde. Zudem wurde eine Untersuchung der Eignung dieses Vorgehens für Bayesische Ansätze auf Raumebene durchgeführt.

WLAN-Fingerprinting auf dem Smartphone bietet ein paar Besonderheiten im Vergleich zu herkömmlichen Ansätzen. Zum einen kann die Erstellung der Datenbank gleich auf dem Smartphone mit erledigt werden. Damit werden Nutzer-generierte Datenbanken gut unterstützt. Zum anderen stehen einige weitere Sensoren zu Verfügung, die die Ortung verbessern können. Zudem kann zwischen Endgeräte-zentrischer Ortung und Endgeräte-unterstützter Ortung gewählt werden, was vor allem einen Einfluss auf die Privatsphäre, aber auch auf die Komplexität der Datenbasis und der Berechnungen hat. Smartphones sind zwar inzwischen sehr leistungsfähig und vergleichbar mit handelsüblichen Computern vor wenigen Jahren. Allerdings sollte die Leistung nicht dauerhaft abgerufen werden, da der Akku sonst sehr schnell aufgebraucht ist. Die Endgeräte-unterstützte Positionsbestimmung benötigt hingegen eine Internetverbindung, welche ebenfalls Strom verbraucht und einer gewissen Latenz unterliegt. Zudem ist eine Internetverbindung nicht immer verfügbar oder es können durch die Verbindung weitere Kosten entstehen.

Das SMARTPOS-System

Das SMARTPOS System [64] wurde so entwickelt, dass beide Ansätze der Positionsbestimmung unterstützt werden. Insbesondere kann ohne Kommunikation mit einer Infrastruktur die Positionsbestimmung einzig auf dem Endgerät ausgeführt werden. Der Nutzer kann sich selbst eine Trainingsdatenbank erzeugen oder eine bestehende Datenbank zusammen mit einem Bitmap-basierten Gebäudeplan herunterladen. Da verschiedene Endgeräte oftmals eine starke Divergenz bezüglich der gemessenen Signalstärke aufweisen [69], können auch Daten eines bekanntermaßen ähnlichen Geräts zur Verfügung gestellt werden. Der Plan wird in jedem Fall als Referenz und Visualisierung der Position benötigt, kann jedoch dauerhaft auf dem Gerät gespeichert werden und steht für alle nachfolgenden Aufgaben der Positionsbestimmung im Gebiet zu Verfügung. Die Erstellung der Trainingsdaten ist einfach und intuitiv gehalten. Der Nutzer muss nur auf den Ort der Karte klicken, an dem er sich gerade befindet, um eine aktive Signalstärkemessung auszulösen. Dadurch wird ein Fingerprint an der Stelle erzeugt und zusammen mit der Position und der aus der aktuellen Kopassmessung ermittelten Blickrichtung persistent gespeichert.

In SMARTPOS wird zur Positionsbestimmung mithilfe des Smartphones kontinuierlich und aktiv nach umliegenden Access Points gescannt und die Signalstärke der empfangenen Antwort zusammen mit der MAC-Adresse des Access Points in einem Vektor gespeichert.

Zeitgleich wird die Orientierung des Geräts mithilfe des digitalen Kompass ermittelt. Da auch die Fingerprints in der Datenbank die Orientierung des Smartphones zu Zeitpunkt der Messung gespeichert haben, lässt sich diese mit der aktuellen Ausrichtung vergleichen. Einen Fingerprint, welcher bei einer deutlich anderen Ausrichtung aufgenommen wurde, entspricht trotz gleicher Signalstärkecharakteristiken oftmals einem anderen Aufenthaltsort. Kompassmessungen sind am selben Ort meist zeitunabhängig ähnlichen Störungen ausgesetzt. Zudem verursacht die Abschottung des Körpers innerhalb von 45° Signalstärkeabweichungen in der Größenordnung des Rauschens von $3 - 5$dBm. Daher kommen in SMARTPOS nur solche Fingerprints mit einer maximalen Abweichung von 50° als nächste Nachbarn in Frage. Dadurch wird die Genauigkeit der Positionsbestimmung erhöht und die zu untersuchende Trainingsdatenbank auf fast 25 % reduziert. Die Reduktion ist besonders wichtig, da dadurch die Berechnungskomplexität und damit der Energieverbrauch auf dem Smartphone sinkt. Zur weiteren Positionsbestimmung auf der reduzierten Datenbank wird ein gewichteter kNN Algorithmus genutzt (siehe Gleichung (4.14)). Die Wahl fällt auf kNN anstelle von Bayes, da kNN mit wenig Testdaten oftmals ein besseres Ergebnis als Bayes erzielt. Für Bayes sind möglichst viele Messungen an jedem Punkt notwendig, um die Verteilung gut anzunähern. Laut [68] müssen an jedem Referenzpunkt mindesten 20 Messungen durchgeführt werden, um die Signalstärkeverteilung gut anzunähern. Auch der Einsatz einer festen Standardabweichung kann zu Problemen führen, wenn die Signalstärke aufgrund von großem Personenaufkommen, unterschiedlichen Endgeräten oder Energiesparmechanismen schwankt.

Testumgebung und Daten

Der Einfluss des Richtungsfilters wurde anhand von Testdaten in einer Büroumgebung in einem Gebäude der Ludwig-Maximilians-Universität München evaluiert. Dazu wurden mit einem HTC Desire an 79 Referenzpunkten bezüglich jeder der vier Hauptachsen des Gebäudes jeweils 3 Signalstärkemessungen gemacht. Der Mittelwert der drei Messungen wurde als Fingerprint in die Datenbank gespeichert, wodurch insgesamt 316 Fingerprints aufgenommen wurden. Die Referenzpunkte sind in Abbildung 4.7 als graue Punkte dargestellt. Die Abbildung zeigt zudem die Menge der 16 zufällig verteilten Testreferenzpunkte mit insgesamt 64 Fingerprints, dargestellt als schwarze Punkte, welche jeweils eine Signalstärkemessung beinhalten und als Ersatz für Online-Messungen hergenommen werden. Dementsprechend wurde das Gerät zur Evaluation in der Hand gehalten.

Evaluation des Richtungsfilters für kNN

Die Ergebnisse zeigen, dass der Richtungsfilter nicht nur die Komplexität auf ein Viertel reduziert, da alle Berechnungen nur auf den Fingerprints einer ähnlichen Richtung durchgeführt werden, sondern zudem die Genauigkeit und Präzision deutlich vergrößert. Dieser Effekt tritt bereits bei einer geringeren Anzahl an nächsten Nachbarn ein und zeigt somit, dass im Falle des Richtungsfilters mit großer Wahrscheinlichkeit der tatsächlichen Position naheliegende Fingerprints bevorzugt werden. Ebenso werden nur wenige Fingerprints in

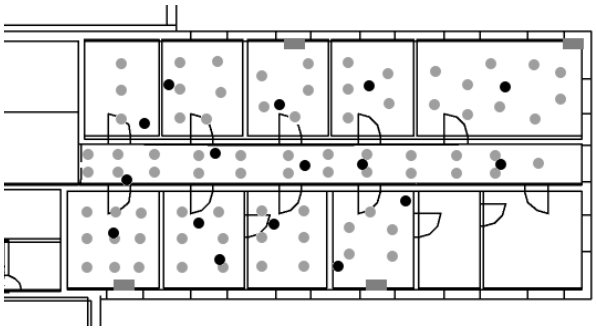
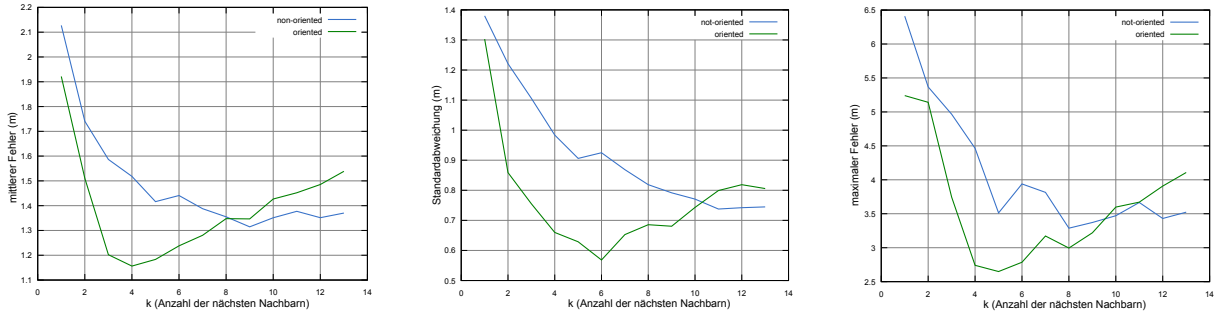


Abbildung 4.7: Trainingsdaten (graue Punkte) und Testdaten (schwarze Punkte). APs sind als graue Rechtecke eingezeichnet.



(a) Der mittlere Fehler in Meter

(b) Die Standardabweichung in Meter

(c) Der maximale Fehler in Meter

Abbildung 4.8: Einfluss der Blickrichtung auf die Positionsbestimmung. [64]

die Positionsbestimmung einbezogen, welche nur aufgrund der körperlichen Abschottung ein ähnliches Muster ergeben, aber von der tatsächlichen Position entfernt liegen. Wie man in Abbildung 4.8 erkennen kann, liegt der minimale Fehler bei $1,16$ Meter für $k = 4$ (von kNN, also die Anzahl der einbezogenen nächsten Nachbarn) unter Einsatz des Orientierungsfilters, während im herkömmlichen gewichteten kNN der minimale Fehler bei $1,31$ Metern liegt und erst für $k = 9$ erreicht wird. Auch die Standardabweichung und die maximale Abweichung sind unter Einbezug der Orientierung bereits bei kleineren Werten von k deutlich niedriger.

Evaluation des Richtungsfilters bei Bayes auf Raumebene

Als alternative Positionsbestimmung wird in diesem Abschnitt ein Bayesischer Ansatz auf Raumebene vorgestellt. Durch die Aggregation von Fingerprints im selben Raum kann von einer realistische Annäherung an die Verteilung der Signalstärken ausgegangen werden. Um Bayes auf Fingerprint-Ebene durchzuführen, muss der Nutzer für die Erzeugung eines Fingerprints einige Sekunden am selben Ort verharren, um eine genügende Anzahl an Messungen zu sammeln. Um den Aufwand zu reduzieren werden dementsprechend alle

Messungen pro Raum zusammengefasst und daraus die Verteilung der Messwerte berechnet. Auf Basis dieser Verteilung wird anschließend nach Gleichung (4.19) der Raum mit der höchsten Aufenthaltswahrscheinlichkeit als tatsächliche Position angenommen. Um im langen Flur eine große Streuung der Messwerte zu vermeiden, wurde dieser in drei logische Orte zerteilt (siehe Abbildung 4.9).

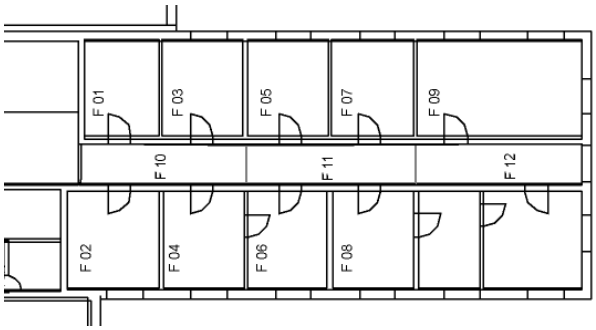


Abbildung 4.9: Die Räume für den Naiven Bayesischen Schätzer. [64]

Der Bayesische Ansatz wurde mittels 10-facher stratifizierter Kreuzvalidierung evaluiert, wobei als Referenzdaten ein zufälliges Viertel der Messungen genommen wurde und dieses mit den Mengen aller Fingerprints einer gleichen Orientierung verglichen wurde. Dementsprechend ist die Datenbasis von ähnlicher Größe und eine Vergleichbarkeit gewährleistet. Interessanterweise ist das Ergebnis, wie man in Tabelle 4.2 sehen kann, negativ. Das bedeutet, dass die Erkennungsrate des richtigen Raumes sinkt, sobald die Richtung als Filter genutzt wird.

Daten	**Anzahl an Fingerprints**	**Erfolgsrate**
Alle Richtungen	78	79%
Norden	72	62,5%
Westen	77	70,13%
Osten	82	65,85%
Süden	82	71,95%

Tabelle 4.2: Evaluation von Bayes auf Raumebene mit Richtungsfilter.

Neben der empirisch ermittelten Verschlechterung gibt es noch einen weiteren Grund den Richtungsfilter bei Raum-basierter Ortung zu vermeiden. Da Klassifikationsverfahren im Allgemeinen besser arbeiten, wenn sie eine größere Datenbasis zu Verfügung haben, ist es auch bei Naiven Bayesischen Ansätzen sinnvoll, alle Testdaten in die Parameterschätzung mit einzubeziehen. Zudem wird bei Bayes durch den Richtungsfilter keine Reduktion der Komplexität erreicht: Unabhängig vom Filter besteht die Verteilung für jeden Raum aus einer Normalverteilung für jeden Access Point. Damit hängt die Anzahl der Berechnungsschritte nur von der Anzahl der APs ab. Alle Messungen werden bereits in der Offline-Phase zur Schätzung der Parameter der Verteilung verwendet, anschließend wird nur noch mit der Verteilung gerechnet.

Evaluation der weiteren Eigenschaften

Die Bewertung bezüglich der weiteren Eigenschaften fällt bei SMARTPOS folgendermaßen aus:

- Die *Abdeckung*, die durch SMARTPOS erreicht wird, hängt wiederum von dem Gebiet ab, in dem das System kalibriert wird. Bei entsprechendem Aufwand kann eine 100%-ige Abdeckung eines Gebäudes erreicht werden, solange auch eine 100%-ige Abdeckung mit genügend überlappenden WLAN APs gegeben ist.

- Die *Anschaffungskosten* bei bestehender WLAN-Infrastruktur sind gering, zudem sollte eine Komponente zum Austausch von nutzergenerierten Datenbanken existieren. Der große Aufwand einer initialen Kalibrierung gehört wiederum zu den bei Installation anfallenden Kosten.

- Die *Betriebskosten* entstehen durch möglicherweise notwendige Nachkalibrierungen, die im Fall von Änderungen in der AP Konfiguration durchgeführt werden müssen.

- Es wird eine flächendeckende WLAN **Infrastruktur** benötigt.

- Durch die endgeräte-zentrische Positionsbestimmung ist die *unterstützte Nutzeranzahl* unbegrenzt.

- Der *Ergebnistyp* entspricht einer absoluten, geometrischen und deterministischen Positionsangabe.

- Die *Verfügbarkeit* ist gegeben, da heutige Smartphones grundsätzlich mit WLAN ausgestattet sind.

- Solange keine Änderungen in der Umgebung auftreten, weist das System eine recht hohe *Zuverlässigkeit* auf.

- Die *Update-Rate* entspricht ungefähr einer Sekunde für aktive Signalstärkemessung und in Abhängigkeit von der Größe der Datenbank bis zu einer weiteren Sekunde für die Positionsbestimmung.

- Ein 100%-iger *Schutz der Privatsphäre* ist nicht gegeben, da das System zur Positionsbestimmung mit den Access Points kommunizieren muss. Da dies bei aktiviertem WLAN sowieso der Fall ist, kann zumindest von einem grundlegenden Schutz ausgegangen werden.

Somit eignet sich SMARTPOS bereits für viele ortsbezogene Dienste, kann aber in Bezug auf die Update-Rate, den Kalibrierungsaufwand und die Positionsgenauigkeit weiter verbessert werden. Ohne bestehende WLAN-Infrastruktur ist gar keine Positionsbestimmung möglich.

4.3.4 Hinzunahme von Bewegungsmodellen zum kontinuierlichen Tracken

Trotz der stark erhöhten Genauigkeit und Präzision konnte die Hinzunahme des Kompass ein bekanntes Problem des WLAN-Fingerprintings nicht lösen. Bei aufeinanderfolgenden Ortungsergebnissen kommt es gerade bei bewegten Zielen aufgrund von Fading häufig zu unrealistischen Sprüngen in der Positionsbestimmung, die bei sequentieller Verbindung der geschätzten Positionen an ein Zick-Zack-Muster erinnern. Dieser Effekt soll unbedingt vermieden werden, um dem Nutzer eine nachvollziehbare Positionsbestimmung zu bieten.

In [83] stellen Machaj und Brida einen Ansatz vor, bei dem die aktuelle Bewegung aus aufeinanderfolgenden Positionsschätzungen berechnet wird und die betrachteten Fingerprints auf eine aus der Bewegung berechnete Region eingeschränkt werden. Dadurch werden große Sprünge vermieden, kleinere Sprünge sind jedoch nach wie vor möglich. Ein anderer Ansatz wird in [55] von Hotta et al. vorgestellt. Dabei wird kNN genutzt, um den Nutzer auf Raumebene zu lokalisieren. Verbessert wird das Verfahren dadurch, dass die Nähe zu Türen für jede Position ermittelt wird und in Abhängigkeit davon die Übergangswahrscheinlichkeit zwischen angrenzenden Räumen ermittelt wird. Beide Verfahren haben gemeinsam, dass auf Basis von WLAN-Messungen Sprünge über größere Distanzen vermieden werden. Zu diesem Zweck wurde ein neuartiges Verfahren entwickelt, welches die aktuelle Position des Nutzers auf Basis der letztbekannten Position vorhersagt und in die Distanzberechnung bei der Ermittlung der nächsten Nachbarn einbezieht [66]. Dieses wird im Folgenden vorgestellt.

Verbesserung von SMARTPOS durch Vorhersagemodelle

Zur Vermeidung von großen Sprüngen in WLAN-Fingerprinting Systemen und zur weiteren Steigerung von Genauigkeit und Präzision wird in diesem Abschnitt ein Mechanismus vorgestellt, der Fingerprints bevorzugt, die nahe zur letzten Positionsschätzung liegen. Da sich der Nutzer seit der letzten Positionsschätzung meist weiterbewegt hat, wird mithilfe eines Bewegungsmodells die Vorhersage über den aktuellen Aufenthaltsort des Nutzers getroffen. Bei einer jeder folgenden Positionsschätzung wird zusätzlich zur Distanz im Signalstärkeraum, also dem Abstand der aktuellen Messung von der im Fingerprint gespeicherten Messung, die euklidische Distanz der vorhergesagten Position zum Fingerprint einbezogen. Dadurch erhalten Fingerprints in der Nähe der vorhergesagten Position einen Vorteil gegenüber weiter entfernt liegenden Fingerprints, was einer kontinuierlichen Fortbewegung entspricht und die auftretenden Sprünge und die dabei zurückgelegten Entfernungen drastisch reduziert. [66]

Gerade bei großen Datenmengen kann die Vorhersage zudem genutzt werden, um ähnlich zu [83] die Größe der Datenbank zu verkleinern. So ist unter realistischen Annahmen die Reduktion der Daten auf einen Umkreis um die geschätzte Position möglich, wobei der Kreisradius aus dem maximalen Fehler pro Zeiteinheit und der vergangenen Zeit seit der letzten Messung berechnet wird. Damit wird die Positionsbestimmung in großen Gebäudekomplexen auch ohne zusätzliche Infrastruktur in Echtzeit auf dem Smartphone alleine

möglich.

Setzt man ein einfaches Bewegungsmodell ein, so ist der Ansatz auch auf Geräten anwendbar, die keine realistischen Daten über die Fortbewegung aus anderen Sensoren gewinnen können. In diesem Fall erzielt der vorgeschlagene Algorithmus sogar bessere Ergebnisse als ein klassischer linearer Filter, der eine Verallgemeinerung des Kalmanfilters darstellt. Im Fall von qualitativ hochwertiger Schritterkennung erzielt der lineare Filter bessere Ergebnisse.

Ein klassischer linearer Filter

Ein Kalmanfilter stellt einen Spezialfall eines klassischen linearen Filters dar (siehe Gleichung (4.21)), in dem zwei verschiedene Modelle entsprechend ihrer jeweiligen Vertrauenswürdigkeit gewichtet kombiniert werden. Während beim Kalmanfilter das Kalman-Gain als Gewicht für jede Messung aus den Kovarianzmatrizen neu berechnet wird, lassen sich in der verallgemeinerten Version des linearen Filters auch konstante Gewichtungen nutzen. Dies entspricht einem Kalmanfilter mit konstanten Kovarianzmatrizen. Ist also keine Aussage über die Verlässlichkeit der Messung oder Vorhersage möglich und wird zudem angenommen, dass mögliche Fehler aus der gleichen Verteilung kommen, so sind beide Filter äquivalent. Im Folgenden wird davon ausgegangen, dass die Positionsschätzung p_k als Linearkombination der durch das Messmodel geschätzten Position $p_{\text{mes},k}$ und der aus dem Vorhersagemodell stammenden Positionsschätzung $p_{\text{pre},k-1}$ geschätzt wird. $p_{\text{pre},k-1}$ wird aus der vorhergehenden Positionsschätzung p_{k-1} berechnet.

$$p_{k|k-1} = p_{\text{mes},k} \cdot (1 - \gamma) + p_{\text{pre},k-1} \cdot \gamma \tag{4.21}$$

Dieser Ansatz hat beim Einsatz von WLAN-Fingerprinting im Messmodell einen Nachteil. Die geschätzte Position kann aufgrund des Fadings unrealistisch weit von der zuletzt geschätzten Position entfernt liegen. Zwar wird die geschätzte Position in diesem Fall durch das Vorhersagemodell entsprechend der Gewichtung näher zur alten Position verlagert, allerdings kann die Vorhersage bereits zum Zeitpunkt der Ortsschätzung in das Messmodell einbezogen werden, um näherliegende Fingerprints stärker zu gewichten.

Das Messmodell für WLAN-Fingerprinting

In [66] wird ein Ansatz vorgestellt, bei dem die Position kontinuierlich vorhergesagt wird und in die Gewichtung der nächsten Nachbarn mit einfließt. Das Messmodell besteht aus dem gewichteten kNN-Algorithmus aus Abschnitt 4.3.3, wobei ein modifiziertes Distanzmaß zur Ähnlichkeitsbestimmung zwischen einer Messung und einem Fingerprint genutzt wird. Dabei wird die räumliche Entfernung von Vorhersage und Fingerprint im Umgebungsmodell mit der Distanz der Signalstärken zwischen Messung und Fingerprint kombiniert: Sei $s_{k,i}$ die Distanz im Signalstärkeraum zwischen der aktuellen Messung zur Zeit k zu dem Signalstärkevektor des i-ten Fingerprints f_i. Zudem sei $m_{k,i}$ der Euklidische Abstand im Umgebungsmodell zwischen der vorhergesagten Position $p_{\text{pre},k-1}$ zum Zeitpunkt k und der Position $f_i.p$ des i-ten Fingerprints. Definiert man nun die Entfernung $d_{k,i}$ zwischen einer

Messung und dem i-ten Fingerprint analog zur Gleichung (4.22) als Linearkombination der beiden Distanzen $s_{k,i}$ und $m_{k,i}$, so werden aufeinanderfolgende Positionsschätzungen dadurch geglättet, dass nähere Fingerprints gegenüber weiter entfernten bevorzugt werden.

$$d_{k,i} = (1 - \alpha) \cdot s_{k,i} + \alpha \cdot m_{k,i} \tag{4.22}$$

Ohne weitere Informationen über die Zuverlässigkeit der Modelle muss das Gewicht $\alpha \in [0; 1]$ entsprechend empirisch ermittelt werden. Dabei favorisiert ein hohes α das Vorhersagemodell, während ein niedriger Wert das Messmodell stärker gewichtet. Ist $\alpha = 0$, so entspricht der Ansatz dem bekannten kNN.

Konstante Geschwindigkeit vs. Schritterkennung

Sobald mindesten einmal eine Position $p_{\text{mes},k-1}$ bestimmt wurde, lässt sich mithilfe eines beliebigen Bewegungsmodells und besagter Position kontinuierlich die Position $p_{\text{pre},k-1}$ vorhersagen, bis diese mithilfe einer neuen Messung rekalibriert wird. Um die grundsätzliche Funktion des Algorithmus zu evaluieren, werden zwei verschiedene Bewegungsmodelle genutzt. Das einfachstmögliche Umgebungsmodell geht von einer konstanten Geschwindigkeit in Blickrichtung aus. Dabei wird untersucht, ob solch ein einfaches Modell bereits ausreicht, um eine Verbesserung zu erreichen. Zudem wird analysiert, ob die Verbesserung auch dann eintritt, wenn sich der Nutzer mit unterschiedlichen Geschwindigkeiten bewegt oder stehen bleibt. Dieses Modell kann entsprechend zum Einsatz kommen, wenn nur WLAN und Kompass zur Verfügung stehen. Zudem wird ein möglichst realitätsnahes Modell der Fortbewegung durch Schritterkennung anhand der Beschleunigungssensoren auf dem Smartphone umgesetzt, um die Performanz in einem solchen Szenario zu evaluieren. Dabei wird der Ansatz zur Schritterkennung aus Abschnitt 4.2.1 verwendet und bei einem erkannten Schritt die Position um die feste Schrittlänge in Richtung der aktuellen Kompassmessung verschoben.

Testumgebung und Daten

Zur Evaluation auf aktuellen Testdaten wurde wiederum eine Fingerprint-Datenbank ähnlich zu Abbildung 4.7 erzeugt. Anstelle von einzelnen Testreferenzpunkten wurden jedoch zwei unterschiedliche Testpfade T1 und T2 aufgezeichnet (siehe Abbildung 4.10). T1 beinhaltet dabei zwei Stellen, an denen angehalten und umgedreht wurde. T2 wurde mit annähernd konstanter Geschwindigkeit durchlaufen. Um verschiedene Parameter auf der gleichen Datenbasis zu evaluieren, wurden die Testdaten zur späteren Verarbeitung aufgezeichnet. Dazu wurde eine vorher festgelegte Strecke abgelaufen, während am in der Hand gehaltenen Smartphone aktive Suchläufe nach WLAN Access Points mit einer Rate von ca. 1 Hz durchgeführt wurden. Die Ergebnisse wurden ebenso wie die Messungen des Kompass für die Blickrichtung und des Beschleunigungssensors für die Schritterkennung (jeweils mit einer Rate von ca. 5 Hz) mit einem Zeitstempel angereichert und in eine Datei geschrieben. Ähnlich zu SMARTPOS wird die Genauigkeit durch die mittlere Abweichung und die Präzision durch die Standardabweichung bestimmt.

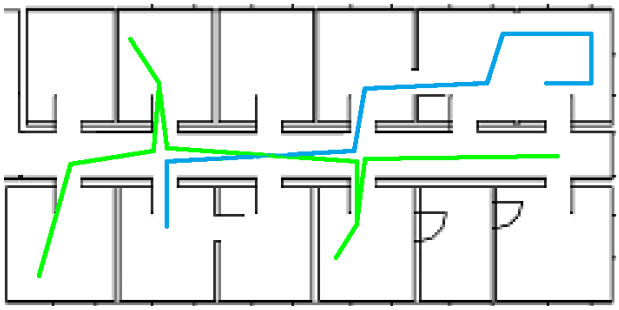


Abbildung 4.10: Zwei Testpfade für die Evaluation, jeweils ausgehend von rechts nach links. T1 (grün) hat eine Länge von 42 m, T2 (hellblau) ist 27 m lang. [66]

Im Folgenden werden verschiedene Details der verbesserten Abstandschätzung evaluiert. Als Referenz dienen das unmodifizierte SMARTPOS System und der klassische lineare Filter, allerdings wird auch die Genauigkeit der Kompassinformationen untersucht, da dieser naturbedingt anfällig gegenüber elektromagnetischen Störquellen ist, welche in Gebäuden vorkommen können.

Evaluation von SMARTPOS beim Tracking

Die Genauigkeit und Präzision von SMARTPOS ist im Fall der kontinuierlichen Bewegung schlechter als im stationären Fall. Dies liegt an Fading-Effekten, die durch die Bewegung hervorgerufen werden, und der Verzerrung von Messungen, da die unterschiedlichen Frequenzen durch die Fortbewegung an leicht unterschiedlichen Orten gescannt werden. Allerdings gilt auch hier, dass die Einbeziehung der Blickrichtung bei der gleichen Anzahl von nächsten Nachbarn k deutlich bessere Ergebnisse erzielt. Tabelle 4.3 zeigt die Genauigkeit und Präzision von gewichtetem 3NN und der SMARTPOS-Variante O3NN bei beiden Tracks. Dabei kann man erkennen, dass der Einbezug der Orientierung die Genauigkeit bereits um ca. 15 % und die Präzision gar um ca. 33 % verbessert.

	3NN T1	**3NN T2**	**O3NN T1**	**O3NN T2**
Mittlerer Fehler	2,01 m	1,74 m	1,72 m	1,52 m
Standardabweichung	1,52 m	1,00 m	1,02 m	0,67 m

Tabelle 4.3: Genauigkeit der kontinuierliche Positionsbestimmung kNN und SMARTPOS.

Qualität des Richtungssensors

Zunächst wird die Qualität des Richtungssensors untersucht. Dazu wird die empirisch ermittelte konstante Geschwindigkeit in Track T2 mit den tiefpassgefilterten Kompassdaten zur Positionsschätzung kombiniert. Obwohl vielfach davon abgeraten wird, den Kompass zu benutzen, werden in der Testumgebung sehr genaue Richtungsinformationen gemessen. Trotz elektronischen Geräten in allen Räumen sowie metallischen Bauteilen in Türen,

Tischen und Stühlen, können die gefilterten Kompassdaten als zuverlässig bezeichnet werden. Abbildung 4.11 zeigt die gute Annäherung des geschätzten Pfades an die tatsächlich zurückgelegte Strecke. Dabei fällt auf, dass die Daten die Realität gut abbilden, bei abrupten Richtungsänderungen ist jedoch eine kleine Verzögerung zu erkennen. Der mittlere Fehler beträgt in diesem Fall lediglich $0,82$ m bei einer Standardabweichung von $0,55$ m.

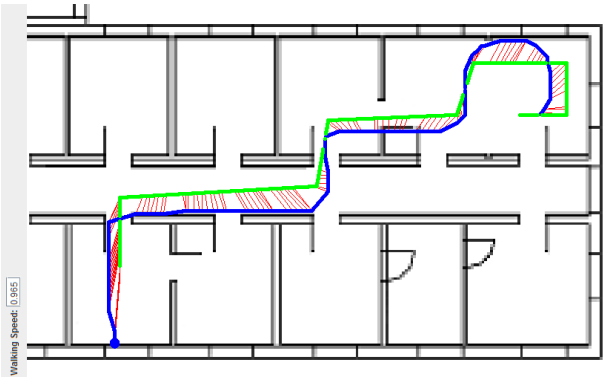


Abbildung 4.11: Die gute Qualität der Kompassmessungen ist an der Nähe des geschätzten Pfads (dunkelblau) zum tatsächlichen Pfad (hellgrün) zu erkennen. Die Abweichungen (rot) zwischen den Pfaden sind minimal. [66]

Wird anstelle der konstanten Geschwindigkeit ein Bewegungsmodell auf Schritterkennung genutzt, so tritt bei einer konstanten Schrittlänge von 1 m ein mittlerer Fehler von $1,20$ m im Pfad T1 und $1,10$ m im Pfad T2 auf, die Standardabweichungen liegen bei $0,77$ m bzw. $0,61$ m. Der Fehler am Ende des Pfades entspricht mit $1,9$ m bzw. $0,85$ m weniger als 5 % der zurückgelegten Strecke, obwohl eine konstante Schrittlänge angenommen wird und kein Mapmatching erfolgt. Die Ergebnisse zeigen damit deutlich, dass die Kompassdaten für die Positionsbestimmung von handgehaltenen Smartphones zuverlässig sind und sich zum Einsatz in einem Bewegungsmodell eignen. Da der Fehler prozentual von der zurückgelegten Strecke abhängt, sind Korrekturverfahren nichtsdestotrotz unerlässlich.

Evaluation des neuen Distanzmaßes

Im Folgenden wird die Kombination von der Euklidischen Distanz im Karten- und Signalstärkeraum untersucht. Dazu werden die variablen Parameter der konstanten Geschwindigkeit im einfachen Bewegungsmodell und der Gewichtung α variiert. Bei einer Geschwindigkeit von 1 ms^{-1} ergeben sich für $\alpha = 0,7$ die besten Ergebnisse in beiden Pfaden, wobei ein mittlerer Fehler von $1,33$ m in T1 und $1,00$ m in T2 mit Standardabweichungen von $0,85$ m bzw. $0,62$ m auftritt. Dabei kommt die deutlich niedrigere Genauigkeit in T1 aus der nicht konstanten Geschwindigkeit, die von dem einfachen Bewegungsmodell nicht erfasst werden kann.

Mit dem empirisch ermittelten α werden nun verschiedene Geschwindigkeiten für das konstante Bewegungsmodell getestet. Geschwindigkeiten zwischen $0,8$ ms^{-1} und $1,0$ ms^{-1}

erscheinen dabei passend, um auch Fälle mit nicht konstanter Bewegung abzudecken. Dabei liegt der Fehler maximal 10 % höher als der minimale Fehler, wobei dieser bei einer Geschwindigkeit von $0,8\ \mathrm{ms}^{-1}$ mit $1,22$ m in T1 und $0,90$ m in T2 auftritt. Die Standardabweichung beträgt $0,81$ m bzw. $0,65$ m.

Schließlich wird die Genauigkeit des Verfahrens ermittelt, wenn Schritterkennung anstelle des konstanten Bewegungsmodells zum Einsatz kommt. Dabei wird die Position bei einem erkannten Schritt in Richtung der aktuellen Kompassmessung um einen Meter verschoben. Wie Versuche mit $\alpha \in \{0,7; 0,75; 0,8; 0,85; 0,9; 0,95\}$ zeigen, soll der Einfluss des Vorhersagemodells aufgrund der hohen Verlässlichkeit der Schritterkennung steigen. Dementsprechend wird das in diesem Fall bei den Tests optimale $\alpha = 0,85$ gewählt, wodurch ein mittlerer Fehler von $1,33$ m in T1 und $0,88$ m in T2 bei einer Standardabweichung von $0,78$ m bzw. $0,46$ m erreicht wird. Auch beim gewählten Ansatz zur Kombination von Vorhersage- und Messmodell kann letzteres dazu genutzt werden, um aufsummierende Fehler durch fehlerhafte Schritterkennung und Schrittrichtungserkennung sowie falsche Schrittlängen zu kompensieren.

Vergleich mit dem klassischen linearen Filter

Nun wird der eigene Ansatz aus [66] mit den Ergebnissen des klassischen linearen Filter verglichen. In diesem Fall wird herkömmliches kNN mit dem jeweiligen Vorhersagemodell durch die in Gleichung (4.21) dargestellte Linearkombination der berechneten Positionen kombiniert. Zuerst erfolgt eine Evaluierung des Bewegungsmodells mit konstanter Geschwindigkeit. Hierbei werden verschiedene Werte für γ ausgewertet, wobei der kleinste mittlere Fehler $1,51$ m bei T1 mit $\gamma = 0,7$ und $1,14$ m bei T2 mit $\gamma = 0,8$ beträgt. Die Standardabweichung beträgt $0,89$ m bzw. $0,53$ m. Damit ist der Fehler mehr als 10 % größer, als der mittlere Fehler unter der modifizierten Distanzberechnung.

Vergleicht man beide Ansätze jedoch miteinander, wenn Schritterkennung im Vorhersagemodell genutzt wird, so ist das Gegenteil der Fall, und der klassische lineare Filter verzeichnet die besseren Ergebnisse. Mit $\gamma = 0,85$ beträgt der mittlere Fehler in diesem Fall $1,24$ m bei T1 und $0,83$ m bei T2, mit entsprechenden Standardabweichungen von $0,69$ m bzw. $0,42$ m. Damit ist das Ergebnis etwa 7 % besser, als mit der modifizierten Distanzschätzung. Dies ist darauf zurückzuführen, dass das Vorhersagemodell genauer wird, als das Messmodel. Bei der modifizierten Distanzschätzung wird das Ergebnis jedoch auf ein gewichtetes Mittel an Fingerprint-Positionen beschränkt, wodurch trotz Einbezug der Vorhersage ein großer Fokus auf das Messmodell gelegt wird.

Beide Ansätze lassen sich miteinander einfach kombinieren, indem das Vorhersagemodell zunächst in die Distanzberechnung einbezogen wird und anschließend die ermittelte Position zusätzlich mit der Vorhersage linear kombiniert wird. Dabei müssen natürlich die Interpolationsparameter an den geänderten Fall angepasst werden. Bei den vorliegenden Testdaten wurde bei Schritterkennung der minimale Fehler für $\alpha = 7,5$ und $\gamma = 0,5$ erreicht. Mit $1,14$ m mittlerer Fehler bei einer Standardabweichung von $0,81$ m ist das Ergebnis für T1 besser, mit $0,87$ m mittlerer Fehler und $0,50$ m Standardabweichung ist das Ergebnis bei T2 jedoch leicht schlechter. Abbildung 4.12 zeigt eine Zusammenfassung

der verschiedenen Ergebnisse aggregiert auf beiden Testpfaden.

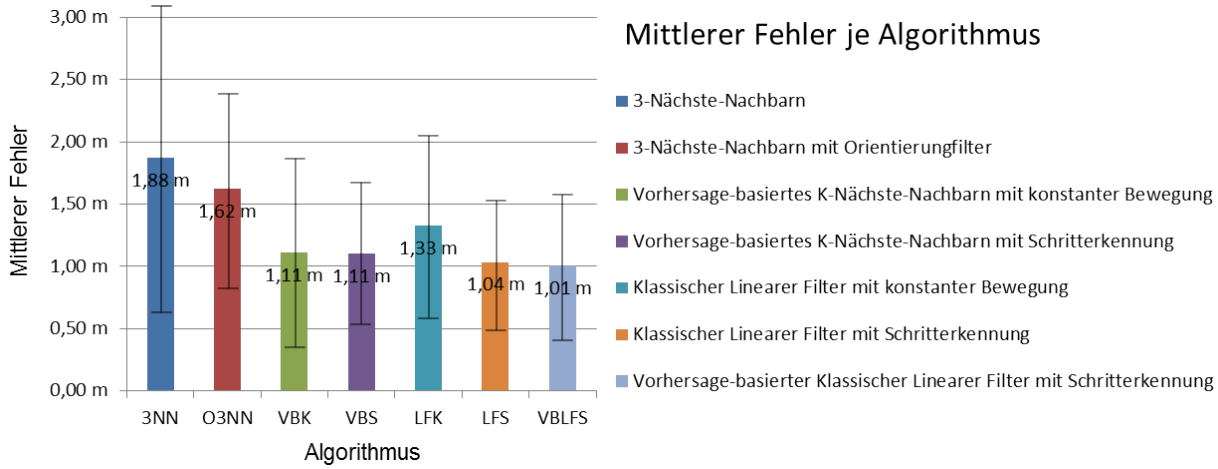


Abbildung 4.12: Überblick über die durchschnittliche Genauigkeit und Präzision.

Fazit

Die Auswertung der Ergebnisse und verschiedenen Konfigurationen liefert einen Einblick in die Kombinationsmöglichkeiten von Vorhersage- und Messmodell. Verwendet man ein eher ungenaues Vorhersagemodell, das rein auf Annahmen und nicht auf tatsächlich gemessenen Werten beruht, so erscheint es anhand der gemessenen Daten sinnvoll, dass diese Informationen zwar zur Verbesserung des Messmodells hergenommen werden sollen, allerdings eher zur Auswahl geeigneter Messpunkte, als zur Korrektur der tatsächlich ermittelten Position. Unterstützt man seine Vorhersage durch reale Beobachtungen, so ist eine realistischere Aussage über die vorhergesagte Position möglich. Das muss bei der Wahl der Gewichte berücksichtigt werden. Am besten wird die vorhergesagte Position mit der gemessenen Position entsprechend einem Filter verrechnet. Zudem kann die Genauigkeit dadurch weiter verbessert werden, dass die Vorhersage zusätzlich in das Messmodell mit einbezogen wird. Auf diese Erkenntnisse stützt sich auch das Design des in [65] und Abschnitt 5.1 vorgestellten Partikelfilters. Dabei wird das Messmodell nur noch zur Gewichtung der einzelnen Partikel genutzt, während deren Position einzig durch das Vorhersagemodell verändert wird.

4.3.5 Fehlerabschätzung bei WLAN-Fingerprinting

Um WLAN-Fingerprinting als Messmodell in einem Bayesischen Filter einzusetzen, ist die Angabe einer möglichst realistischen Wahrscheinlichkeitsverteilung zu einer Positionsmessung notwendig. Aber auch für andere Anwendungsfälle, wie die Auswertung verschiedener Regeln in der ortsbezogenen Zugriffskontrolle (vgl. [84]), ist die Angabe der Vertrauenswürdigkeit einer Positionsschätzung notwendig.

Bestehende Arbeiten schätzen den Positionsfehler bei WLAN-Fingerprinting oftmals durch einen zellbasierten Ansatz [140, 144, 139]. Dazu wird ein Bayesischer Ansatz des Fingerprintings verfolgt und für jede Zelle eine entsprechende Signalstärkeverteilung ermittelt. Zur Positionsschätzung wird für jede Zelle die Aufenthaltswahrscheinlichkeit entsprechend der gemessenen Signalstärke ermittelt und somit eine diskrete Wahrscheinlichkeitsverteilung gewonnen. Andere Arbeiten, wie die von Evennou und Marx [29], gehen von einer Normalverteilung um die ermittelte Position aus, wobei die Varianz entsprechend der Signalstärkeverteilung an der geschätzten Position berechnet wird. Beder et al. bestimmen in [15] den erwarteten Fehler offline aus der inneren Struktur der Fingerprint Daten, indem die Kovarianzmatrix der Signalstärken eines Fingerprints verwendet wird, um den erwarteten Fehler abzuschätzen.

Für den KNN Algorithmus stellen Lemelson et al. in [76] mehrere Methoden zu Fehlerabschätzung vor. Dabei werden entweder Cluster aus ähnlichen Fingerprints gewählt, um das Fehlergebiet anzunähern, oder die Position der nächsten Nachbarn untereinander genutzt, um einen Fehlerkreis anzugeben, in dem sich die tatsächliche Position befindet. Im Folgenden wird die Art des Fehlers untersucht und eine eigene Methode aus [85] vorgestellt, welche die Fehlerverteilung zur Laufzeit aus der geschätzten Position sowie dem Gewicht und der Referenzposition der nächsten Nachbarn ermittelt. Die Ergebnisse werden mit den Schätzern aus verwandten Arbeiten verglichen und die Überlegenheit des eigenen Ansatzes gezeigt.

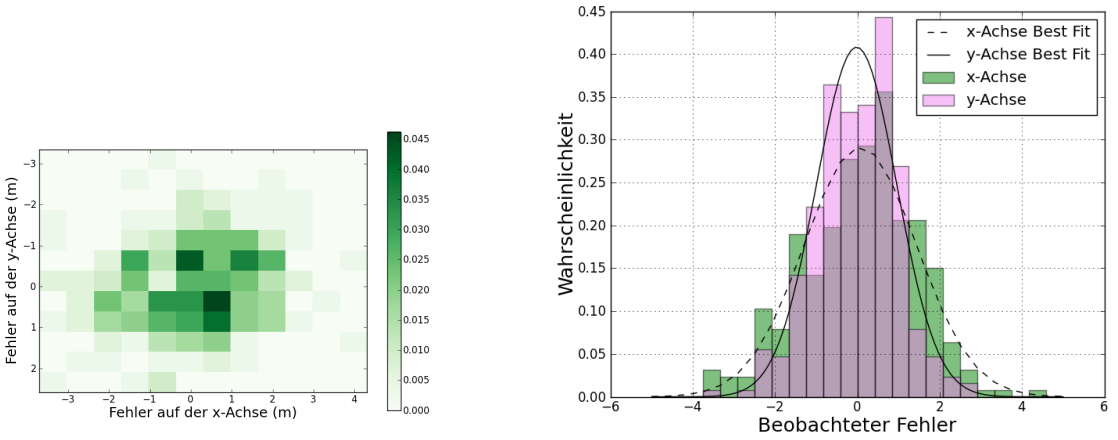
Fehlerverteilung

Zur Ermittlung der Fehlerverteilung wurde auf Basis der Testdaten aus Abschnitt 4.3.3 mit den optimalen Einstellungen des SMARTPOS Systems eine *leave-one-out* Kreuzvalidierung durchgeführt. Die Fehlerverteilung ist in Abbildung 4.13a in Form eines zweidimensionalen Histogramms zu sehen, wobei die beobachteten Fehler dem Abstand der geschätzten Position relativ zur bekannten tatsächlichen Position in x- und y-Richtung entsprechen.

Die Ergebnisse lassen die Vermutung zu, dass die Fehler je Achse unkorreliert und stochastisch unabhängig sind, wodurch eine unabhängige Betrachtung der Fehler möglich wird. Die unabhängige Betrachtung der einzelnen Fehler je Achse findet sich in Abbildung 4.13b. Allein die graphische Analyse der Histogramme deutet stark auf eine Normalverteilung des Positionsfehlers hin. Diese Hypothese wurde zusätzlich mit einem Wilk-Shapiro Test mit 50 zufällig ausgewählten Datenpunkten je Achse überprüft. Bei einem Signifikanzniveau von 5 % lagen die Werte der Teststatistik mit $X = 0,960$ und $Y = 0,955$ bei beiden Achsen über dem kritischen Wert von $0,947$ für eine Stichprobe der Größe 50. Damit liegt mit einer Wahrscheinlichkeit von mindestens 95 % tatsächlich eine Normalverteilung vor.

Online Fehlerschätzung

Nachdem die Vermutung einer Normalverteilung der Fehler bestärkt wurde, wird in diesem Abschnitt eine neue Methode zur Schätzung der Parameter der Normalverteilung vorgestellt und mit bestehenden Methoden aus [76] verglichen. Die Autoren beschreiben drei



(a) Fehlerverteilung in 2D.

(b) Fehlerverteilung bezüglich x- und y-Achse.

Abbildung 4.13: Histogramme zur Fehlerverteilung im SMARTPOS System. [85]

verschiedene Methoden, um den Fehler einer Positionsbestimmung mit kNN aus den Referenzpositionen $(l_1, ..., l_k)$ der k nächsten Nachbarn abzuschätzen.

Die erste Methode (Gleichung (4.23)) nutzt den mittleren Abstand der nächsten Nachbarn zum nächstgelegenen Nachbarn im Signalstärkeraum:

$$\sigma_{m1}(l_1, l_2, \cdots, l_k) = \frac{1}{k-1} \sum_{i=2}^{k} \|l_1 - l_i\|_2 \tag{4.23}$$

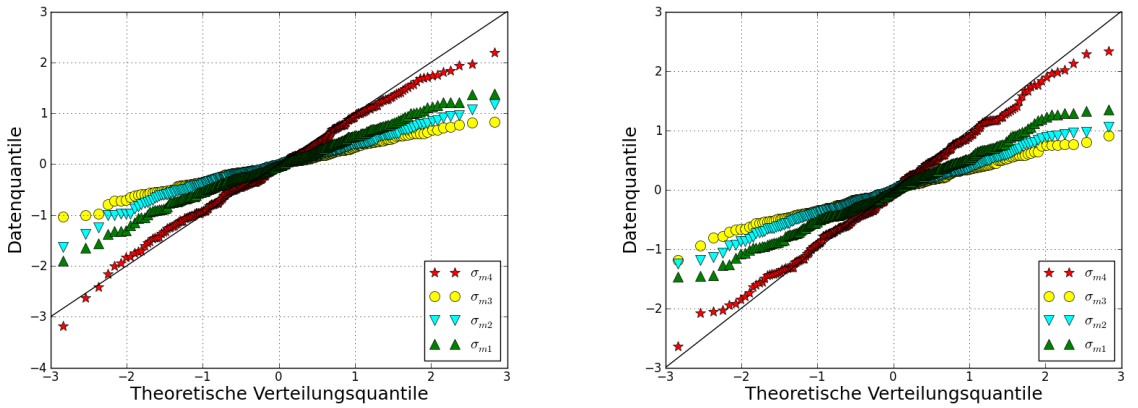
Die zweite Methode (Gleichung (4.24)) nutzt den höchsten Abstand der nächsten Nachbarn zum nächstgelegenen Nachbarn im Signalstärkeraum als Fehlerschätzer:

$$\sigma_{m2}(l_1, l_2, \cdots, l_k) = \max_{i \in \{2,...,k\}} \|l_1 - l_i\|_2 \tag{4.24}$$

Die letzte Methode (Gleichung (4.25)) nutzt den höchsten Abstand unter zwei beliebigen der nächsten Nachbarn als Fehlerschätzer:

$$\sigma_{m3}(l_1, l_1, \cdots, l_k) = \max_{(i,j) \in \{1,...,k\} \times \{2,...,k\}} \|l_i - l_j\|_2 \tag{4.25}$$

Lemelson et al. nutzen die Fehlerschätzer σ_{mi} in [76] jedoch noch nicht, um die Parameter einer Normalverteilung zu schätzen, sondern lediglich als Fehlerradius um die ermittelte Position. Nach den Ergebnissen aus dem letzten Abschnitt scheint jedoch eine Schätzung der Normalverteilung eine feingranularere und deutlich genauere Fehlerabschätzung zu bieten. Daher wird unter der Annahme der Unabhängigkeit der Fehler an dieser Stelle ein Schätzverfahren für Mittelwert und Standardabweichung einer Normalverteilung gegeben. Für eine Positionsschätzung $l = (l_x, l_y)$ werden zwei Normalverteilungen $\mathcal{N}(l_x, \sigma_{mi})$



(a) qq-Plot der Testdaten auf der x-Achse.

(b) qq-Plot der Testdaten auf der y-Achse.

Abbildung 4.14: qq-Plots der Testdaten bezüglich beider Achsen. [85]

und $\mathcal{N}(l_y, \sigma_{mi})$ angegeben, welche die Fehlerwahrscheinlichkeit für die entsprechende Positionsschätzung angeben. Zu diesem Zweck wird σ_{mi} für die Standardabweichung der Normalverteilung eingesetzt.

Zudem wurde ein neuer Schätzer σ_{m4} für die Standardabweichung in Gleichung (4.26) definiert, der aus der gewichteten Abweichung der nächsten Nachbarn zur geschätzten Position $l = (l_x, l_y)$ berechnet wird. Die jeweiligen Gewichte jedes Nachbarn entsprechen dabei den Gewichten des gewichteten kNN Algorithmus aus Gleichung (4.16):

$$\sigma_{m4}(l, l_1, l_2, \ldots, l_k) = \sum_{i=1}^{k} w_i \| l - l_i \|_2 \tag{4.26}$$

Um die Qualität der einzelnen Schätzer zu testen, wurden die beobachteten tatsächlichen Fehler mithilfe der bekannten tatsächlichen Position *gtp* durch die Schätzung der Standardabweichung mithilfe der Formel $(gtp_x - l_x)/\sigma_{mi}$ normalisiert, um die entsprechenden Zufallsvariablen auf die Standardnormalverteilung $\mathcal{N}(0, 1)$ abzubilden. Damit ist ein direkter Vergleich der Fehlerschätzer in einem qq-Plot möglich. Je näher die Stichprobe dabei an der Geraden $x = y$ liegt, desto besser approximiert der Schätzer σ_{m4} die tatsächliche Standardabweichung der zugrundeliegenden Normalverteilung.

Wie in Abbildung 4.14 zu erkennen, schneidet die selbst entwickelte Methode σ_{m4} am besten ab und liefert im Bereich -2σ bis 2σ eine zuverlässige Schätzung der Standardabweichung. Lediglich die Wahrscheinlichkeit eines großen Fehlers wird geringfügig unterschätzt, allerdings immer noch deutlich weniger, als von den anderen Methoden. Damit lässt sich der Schätzer gut von bestehenden Bayesischen Filtern nutzen, um SMARTPOS als Messmodell in die Positionsschätzung einfließen zu lassen.

4.4 Zusammenfassung

In diesem Kapitel wurden zunächst Eigenschaften von Lokalisierungssystemen genannt, die eine Vergleichbarkeit unterschiedlicher Systeme ermöglichen. Dazu gehören Genauigkeit und Präzision ebenso wie die Abdeckung, die Kosten, benötigte Infrastruktur, Frequenz, Verfügbarkeit, Zuverlässigkeit, Skalierbarkeit und Schutz der Privatsphäre.

Anschließend wurde eine infrastrukturlose Positionsbestimmung durch einen Partikelfilter vorgestellt, welcher Beschleunigungssensor, Gyroskop und Magnetfeldsensor des Smartphones vereint und durch Kartenkorrektur eine genaue relative Positionsbestimmung ermöglicht. Die Neuartigkeit liegt dabei in der dynamischen Kombination von Kompass und Gyroskop entsprechend der gegenseitigen Abweichung und der Verbesserung der Schrittlängenschätzung durch Kalibrierung und Regression oder adaptives Mapmatching. Allerdings wurden zwei bestehende Probleme identifiziert: Die rein relative Positionsbestimmung und die Divergenz auf Freiflächen. Dadurch, dass lediglich relative Positionsänderungen nachvollzogen werden können, ist zunächst einmal keine absolute Positionsbestimmung möglich. Ist also keine initiale Position bekannt, so kann durch die Kartenkorrektur nach einiger Zeit aus einer initialen Gleichverteilung eine Konvergenz der Partikelwolke erreicht werden. Dies lässt zwar den Rückschluss auf eine absolute Position zu, jedoch erschwert oder verhindert Gebäudesymmetrie eine schnelle Konvergenz. Ist in der Umgebung der Partikelwolke nur wenig Kollisionsgeometrie vorhanden, so streuen die Partikel immer stärker, und die Genauigkeit lässt über die Zeit nach.

Um den Problemen zu begegnen, wurden anschließend zwei Fingerprinting-Verfahren zur absoluten Positionsbestimmung entwickelt. Das erste basiert auf Kamerabildern und ist somit unabhängig von installierter Infrastruktur, erlaubt jedoch keine implizite Positionsbestimmung ohne Nutzeraktion. Das zweite Verfahren basiert auf WLAN und nutzt eine Verbesserung des kNN Algorithmus durch Kompassmessungen und optional auch durch Beschleunigungssensoren. Zur Kombination von WLAN-Fingerprinting mit der Koppelnavigation in einem Partikelfilter wurde zudem das Fehlerverhalten genauer untersucht und ein zuverlässiger online Fehlerschätzer entwickelt.

Kapitel 5

Globale Verbreitung und nahtlose Positionsbestimmung

Nachdem im letzten Kapitel einige Ansätze zur Verbesserung der Positionsbestimmung mit WLAN und Kamera-basiertem Fingerprinting sowie zur Koppelnavigation auf Smartphones vorgestellt wurden, werden die Ansätze in diesem Kapitel miteinander kombiniert, um die jeweiligen Schwächen zu überwinden. Zudem werden Algorithmen zur Förderung der globalen Verfügbarkeit, namentlich die automatische Kalibrierung von Fingerprint-Datenbanken, und zur Umsetzung einer nahtlosen Positionsbestimmung zwischen Innen- und Außenbereichen eingegangen. Das Ziel besteht dabei einerseits darin, die bestehenden Probleme der aufwändigen Erhebung von Trainingsdaten in Musterabgleichsalgorithmen zu beheben, und andererseits die globale Bereitstellung von Verfahren zur Positionsbestimmung und Kartenmaterial zu ermöglichen, wodurch auch eine nahtlose Positionsbestimmung ermöglicht wird.

Um den Aufwand der Erhebung von Trainingsdaten zu minimieren, wurde ein Algorithmus entwickelt, der sich durch eine Modifikation des im Abschnitt 4.2.2 vorgestellten Partikelfilters zur automatischen Erzeugung und Kalibrierung von WLAN-Fingerprint-Datenbanken einsetzen lässt. Dabei wird zuerst das in Abschnitt 4.3.3 vorgestellte WLAN-Fingerprinting als Messmodell in den Partikelfilter zur Sensorfusion mit dem bereits vorgestellten Dead Reckoning System als Vorhersagemodell auf dem Smartphone vereint. Anschließend werden die Daten des Dead Reckoning Systems per Rückberechnung, auch Backtracking genannt, verbessert und diese zuletzt zur automatischen Kalibrierung der Trainingsdatenbank für das WLAN-Fingerprinting genutzt. Dies erlaubt die beständige Aktualisierung bestehender Datenbanken ebenso wie die Erzeugung komplett neuer Datenbanken durch die Nutzer selbst und trägt zu einer deutlichen Aufwandsverringerung und somit einer Kostenreduktion bei.

Zur globalen Bereitstellung von Positionsdaten wird eine Plattform vorgestellt, welche auf Basis einer Beschreibungssprache die beliebige und dynamische Kombination eines sich ändernden Angebots von Sensoren ermöglicht. Dadurch kann zu jedem Zeitpunkt die beste Kombination von Sensoren zur Positionsbestimmung genutzt werden, wobei sich die Güte auf vom Nutzer vorgegebene Kriterien bezieht. Insbesondere kann die Plattform

auf wechselnde Verfügbarkeit von Sensoren, wie GPS, reagieren und alternative Möglichkeiten zur Positionsbestimmung anbieten. Die Funktionalität der Plattform wird mithilfe eines Prototyps in einem Szenario der nahtlosen Positionsbestimmung zwischen Innen- und Außenbereichen aufgezeigt. Dabei wird gezeigt, dass allein aufgrund von Regeln zur Genauigkeit und Verfügbarkeit von Sensoren eine nahtlose Positionsbestimmung möglich ist. Zudem wird ein Ansatz zur expliziten Erkennung von Übergängen vorgestellt, der die Genauigkeit beim kritischen Übergang von außen nach innen nochmals deutlich erhöht. Hier kommt das Wissen um die Position der Eingänge zum Einsatz, welches in den in Kapitel 3 vorgestellten Umgebungsmodellen explizit enthalten ist.

5.1 Kombination von Dead Reckoning und WLAN-Fingerprinting

Die Kombination verschiedener Sensoren zur Positionsbestimmung in einem Partikelfilter wird bereits seit vielen Jahren eingesetzt [31, 43, 29]. Dennoch existieren bislang nur wenige Ansätze, die diese rechenaufwändigen und speicherintensiven Filter für den Einsatz in Echtzeit auf dem Smartphone umsetzen [67, 72, 77, 115, 105]. Dies liegt nicht nur am Rechenaufwand, da aktuelle Smartphones die Rechenkapazität von Computern vor wenigen Jahren aufweisen, sondern auch an den sehr günstigen und daher oftmals ungenauen und stark fehleranfälligen Sensoren, die heutzutage in Smartphones verbaut sind.

Der im Rahmen dieser Arbeit entwickelte Filter zur Fusion von WLAN, Kompass, Gyroskop und Beschleunigungssensor ist ein sogenannter *Sequential Importance Resampling* (SIR) Partikelfilter. Gemäß [4] schätzen diese Filter rekursiv den Zustand eines dynamischen Systems auf Basis von verrauschten Messungen. In diesem Fall ist das dynamische System das bewegte Ziel der Ortung und der Zustand sowohl die Position, als auch die Fortbewegung in Form von Bewegungsrichtung und Schrittlänge. In den folgenden Abschnitten wird vor allem auf die Umsetzung des SIR Partikelfilters eingegangen, für eine formale Einführung von Partikelfiltern sei auf Abschnitt 2.2.2 verwiesen.

Zur Positionsbestimmung werden fünf verschiedene Informationsquellen auf dem Smartphone genutzt: Daten des Beschleunigungssensors, des Gyroskops und des Kompasses, WLAN-Signalstärkeinformationen und das Bitmap-basierte Gebäudemodell aus Abschnitt 3.3. Letzteres geschieht aus dem Grund, da für den Partikelfilter insbesondere Kollisionsüberprüfungen möglichst effizient beantwortet werden müssen. Der Systemzustand und damit auch der Zustand jedes Partikels bestehen aus zwei Pixel-Koordinaten x und y auf der Bitmap, der Stockwerksangabe, welche die Nummer der entsprechenden Bitmap darstellt, der Blickrichtung und der Schrittlänge. Das System ist so konzipiert, dass auch bei Ausfall von Gyroskop und/oder WLAN eine Position bestimmt werden kann.

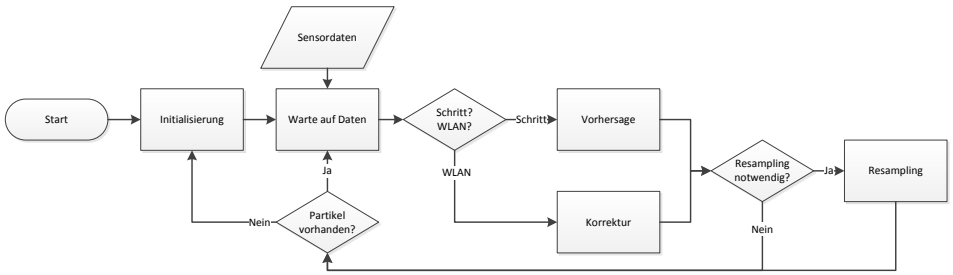


Abbildung 5.1: Ablauf der einzelnen Phasen des SIR Partikelfilters.

5.1.1 Umsetzung von Initialisierung, Vorhersage, Korrektur und Resampling

Wie bereits dargestellt, besteht ein Partikelfilter aus mehreren Phasen, die aufgrund der rekursiven Natur des Filters mehrfach zyklisch ausgeführt werden. Angefangen wird mit der Initialisierung der Partikel, wobei eine initiale Menge von Partikeln nach einer bestimmten Vorschrift erzeugt wird. Anschließend beginnt der rekursive Teil des Filters, wobei dieser stark von den Sensoren und den Zeitpunkten der Messungen abhängt. Eintreffende Daten des Beschleunigungssensors werden von einer Schritterkennungskomponente mithilfe eines ähnlichen Algorithmus, wie in Abschnitt 4.2.1 oder in [79] erläutert wurde, analysiert. Immer, wenn ein Schritt erkannt wird, wird die Vorhersagephase durchgeführt, in der die Partikel bewegt werden. Anschließend wird aufgrund der beim Mapmatching möglicherweise entfernten Partikel die Notwendigkeit einer Resampling-Phase überprüft, in der Partikel geringen Gewichts entfernt und Partikel eines hohen Gewichts auf mehrere Partikel aufgeteilt werden. Immer, wenn neue WLAN-Signalstärkeinformationen verfügbar sind, wird die Korrekturphase ausgeführt, wobei das Gewicht der Partikel entsprechend dem Messmodell neu berechnet wird. Anschließend folgt die Resampling-Phase. Die einzelnen Phasen und der Ablauf sind in Abbildung 5.1 dargestellt. Diese werden nun in den folgenden Abschnitten detaillierter vorgestellt.

Initialisierung

Zur Initialisierung des Filters wird eine feste Menge von N Partikeln entsprechend eines der drei nachstehenden Initialisierungsschemas erzeugt: Das erste Schema verteilt die Partikel gleichmäßig über die begehbare Fläche eines bestimmten Gebietes. Dies entspricht dem Fall, dass keinerlei initiale Informationen über den Aufenthaltsort des Zieles bekannt sind. Beim zweiten Schema werden alle Partikel auf derselben Pixelkoordinate im selben Stockwerk erzeugt. Dies entspricht dem Fall, dass genaue Positionsinformationen verfügbar sind. Das letzte Schema verteilt die Partikel entsprechend einer zweidimensionalen und bivariaten Normalverteilung, die aus einer WLAN-Positionsbestimmung gewonnen wird. Die Normalverteilung spiegelt hierbei die Genauigkeit und Präzision der Schätzung wider und

wird analog zu Abschnitt 4.3.5 aus der vom kNN Algorithmus geschätzten Position und der Position der nächsten Nachbarn konstruiert.

Vorhersagemodell

Die Vorhersagephase wird bei einem erkannten Schritt ausgeführt und beruht somit selbst auf einem verrauschten Messmodell. Ein Schritt wird dabei durch die Analyse der durch einen Tiefpass-Filter gefilterten Beschleunigungswerte erreicht (siehe Abschnitt 4.2.1). In der Vorhersagephase wird die Schrittlänge und die Schrittrichtung wie in Abschnitt 4.2.1 geschätzt und die Plausibilität mit Mapmatching überprüft. Üblicherweise erfolgt im Partikelfilter eine zufälligen Erzeugung neuer Partikel entsprechend der aktuellen Partikel-Verteilung [105]. Im gewählten Ansatz wird hingegen jedes Partikel entsprechend der Gleichung (4.9) separat gestört und bewegt. Die Störung über alle Partikel hinweg bei einer genügend großen Anzahl bietet eine ähnlich gute Annäherung an Gleichung (2.5). Dabei wird zudem Rechenzeit gespart und der Speicherbedarf niedrig gehalten, da nicht bei jedem Schritt N Partikel zufällig gezogen werden müssen.

Nach der Vorhersagephase kann ein Resampling notwendig sein, falls durch Mapmatching Partikel entfernt wurden. Dies wird über die Anzahl der vorhandenen Partikel und das Verhältnis der Partikelgewichte getestet. Sinkt die Anzahl der Partikel durch Kollisionen unter 2/3 der maximalen Anzahl der Partikel, so existieren genügend Partikel mit einem Gewicht von über $1,5N^{-1}$, die in mehrere Partikel geringeren Gewichtes aufgeteilt werden können. Alternativ kann Resampling ausgelöst werden, falls das effektive Gewicht N_{eff} unter einen Grenzwert N_{T} fällt [4]. Dabei gilt Gleichung (5.1) zur Berechnung des effektiven Gewichts.

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} \tag{5.1}$$

Zusätzlich zur Bewegung der Partikel wird mithilfe des Bitmap-basierten Umgebungsmodells das in Abschnitt 4.2.3 vorgestellte Mapmatching durchgeführt. Dabei wird für jedes Partikel eine Kollisionsanfrage mit dem neuen Zustand x_k und dem Vorzustand x_{k-1} an das Umgebungsmodell gestellt, wobei eine negative Antwort bedeutet, dass das Partikel auf seinem Weg eine Kollision hatte. Allein auf Basis der Schritterkennung und des Mapmatchings ist bereits eine kontinuierliche Positionsbestimmung auch bei inital unbekannter Position möglich (siehe 4.2.2).

Korrekturmodell

Wie am Beispiel von Abschnitt 4.3.4 deutlich wurde, kann eine weitere Sensorquelle die Genauigkeit und Präzision erheblich verbessern und im Fall von WLAN zur Korrektur von Dead Reckoning nicht nur zu besseren Ortungsergebnissen führen, sondern im Fall einer unbekannten Startposition eine schnellere Konvergenz herbeiführen, die Streuung auch über längere Zeiträume niedrig halten oder als initiale Schätzung der Position dienen. Daher wird zusätzlich zum Mapmatching eine Korrektur der Positionsinformationen

durch WLAN-Fingerprinting vorgeschlagen, was im Folgenden auch als WLAN-Korrektur bezeichnet wird. Für diese Zwecke reicht eine grobe Positionsbestimmung aus, um die Partikel entsprechend der WLAN-Messungen zu gewichten. Daher wird das effiziente gewichtete kNN Verfahren aus Abschnitt 4.3.3 genutzt, von welchem die Aufenthaltswahrscheinlichkeit $p(x_k|z_k)$ aus Gleichung (2.6) entsprechend Abschnitt 4.3.5 ermittelt wird. Mit einer genügend großen Anzahl k an Nachbarn wird eine bivariate Normalverteilung $\mathcal{N}_{m_x,m_y,\sigma_x,\sigma_y}$ konstruiert, deren Mittelwert (m_x, m_y) durch das gewichtete Mittel der nächsten Nachbarn gegeben ist und deren Standardabweichung (σ_x, σ_y) aus der Abweichung der einzelnen Nachbarn zu diesem Mittelwert berechnet wird. Genauer wird für alle Partikel x_k^i die Aufenthaltswahrscheinlichkeit $p(x_k^i|z_k)$ gemäß der Messung z_k anhand von Gleichung (5.2) bestimmt:

$$p(x_k^i|z_k) = \mathcal{N}_{m_x,m_y,\sigma_x,\sigma_y}(x_k^i) \tag{5.2}$$

Man beachte, dass der Funktionswert der Dichtefunktion an sich keine Wahrscheinlichkeit darstellt, da diese bei einer kontinuierlichen Verteilung für jeden Punkt als Integral ohne Ausdehnung gleich 0 ist, allerdings liefert die Dichtefunktion einen Indikator für die relative Wahrscheinlichkeit. Da das Gewicht der Partikel am Ende der Korrekturphase wieder normalisiert wird, ist die Approximation an dieser Stelle zulässig.

Um der Gleichung (2.6) gerecht zu werden und die Markov Eigenschaft nicht zu verletzen, wird die Aufenthaltswahrscheinlichkeit $p(x_k|z_k)$ anschließend noch mit dem vorherigen Gewicht w_{k-1}^i des Partikels multipliziert:

$$w_k^i = p(x_k^i|z_k) w_{k-1}^i \tag{5.3}$$

Anschließend wird das Gewicht jedes Partikels normalisiert, damit die Partikel wieder eine diskrete Wahrscheinlichkeitsverteilung darstellen. Man beachte, dass die Korrektur durch ein beliebiges absolutes Ortungssystem durchgeführt werden kann, insbesondere durch die in der Literatur als genauer angesehenen Methoden des Bayesischen Fingerprintings. Wichtig sind hierbei nur die zeitliche Unabhängigkeit des Messfehlers, welche zur Korrektur der zeitlich abhängigen Fehler eines Dead Reckoning Systems notwendig ist, und die möglichst genaue Angabe einer passenden Verteilung. Die Entscheidung, ein kNN-basiertes System zu nutzen, hängt mit dem Design der im Abschnitt 5.2 vorgestellten WLAN-Kalibrierung zusammen. In diesem Fall kommt es meist nur zu einer Signalstärkemessung je Fingerprint, wodurch keine Verteilung der Signalstärke berechnet werden kann. Damit lässt sich ein Bayesischer Ansatz durch empirisch ermittelte Verteilungen nicht umsetzen. Auch für die Annahme fester Varianzen im Signalstärkeraum eignet sich eine solcherart kalibrierte Datenbank nicht, da der gemessene Wert teils stark vom erwarteten Wert abweichen kann. In diesem Fall ist kNN das robustere Verfahren.

Resampling

Nach der Vorhersagephase oder der Korrekturphase wird ein Resampling durchgeführt, um Partikel mit einem geringen Gewicht zu entfernen und Partikel mit einem hohen Gewicht in mehrere Partikel aufzuteilen. Dies geschieht, um das Degenerieren des Partikelfilters

wie in [4] beschrieben zu vermeiden und lokal um relevante Zustände eine möglichst hohe Auflösung an Partikeln zu haben. Aufgrund von Echtzeitanforderungen auf einem Smartphone wird die maximale Anzahl an Partikeln auf die initiale Anzahl N beschränkt. Daher wird ein Partikel x_k^i in n neue Partikel aufgeteilt, sobald sein Gewicht um mindesten 50% größer ist als das durchschnittliche Gewicht N^{-1}, wobei $n = w_k^i N$ eine gerundete natürliche Zahl ist, so dass alle neuen Partikel ein Gewicht kleiner als N^{-1} haben. Zudem wird ein Partikel gelöscht, sobald sein Gewicht kleiner als N^{-2} ist.

5.1.2 Evaluation von Genauigkeit und Präzision

Der Partikelfilter wurde in einer Büroumgebung in einem Gebäude der Ludwig-Maximilians-Universität München getestet. Dabei wurden sämtliche Informationen und Sensordaten mit einem HTC Desire Smartphone aufgezeichnet und in einer Desktop-Umgebung ausgewertet. Zu Testzwecken wurde eine Referenzdatenbank angelegt, die Aufnahmen an 64 Referenzpositionen jeweils in Richtung aller vier Gebäudeachsen enthält. Dabei wurden für jeden der insgesamt 256 Fingerprints 10 aktive Scans nach Signalstärkeinformationen ausgeführt und die gemittelte Signalstärke für jeden Access Point zusammen mit der gemittelten Kompass-Orientierung und der Position des Smartphones in die Datenbank gespeichert. Die Referenzpunkte und die Position der Access Points sind in Abbildung 5.2 zu sehen.

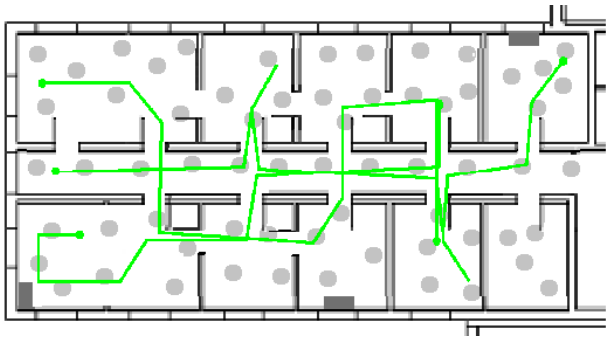


Abbildung 5.2: Referenzpositionen (graue Punkte), Access Points (graue Rechtecke) und Grundwahrheit der Testpfade (grün). Jeder der Pfade wurde einmal von links nach rechts und einmal umgekehrt aufgenommen.

Zudem wurden, wie in Abbildung 5.2 zu sehen ist, 6 Pfade aufgenommen. Während ein Pfad abgelaufen wurde, wurden die Daten des Beschleunigungssensors und des Kompasses mit einem Zeitstempel angereichert und gespeichert. Ein Gyroskop stand auf dem Testgerät leider nicht zur Verfügung. Die Frequenz, mit der die Daten ausgelesen wurden, betrug ca. 5 Hz. Zusätzlich wurden kontinuierlich aktive Scans nach Signalstärkeinformationen mit einer Rate von ca. 1 Hz durchgeführt, ebenfalls mit einem Zeitstempel versehen und gespeichert. Zudem wurde beim Passieren einer vorher festgelegten und am Boden markierten Referenzposition im Pfad ein Zeitstempel gespeichert. Dadurch konnte im Nachhinein eine hochqualitative Grundwahrheit berechnet werden.

Im Folgenden wird die Gesamtperformanz des Systems in Bezug auf Genauigkeit und Präzision beschrieben. Die Genauigkeit wird durch den mittleren Fehler und den Fehler am Ende eines Pfades angegeben. Die Präzision wird durch die Standardabweichung gegeben. Da es sich um einen probabilistischen Filter mit Zufallszahlen einer bestimmten Verteilung handelt, wurde die Auswertung jeweils 50 mal mit 10.000 Partikeln für jeden einzelnen Pfad durchgeführt, wobei Beschleunigungssensor, Kompass, WLAN-Signalstärkeinformationen und Kartendaten genutzt wurden. Es stand eine aktuelle hochauflösende Referenzdatenbank zur Verfügung. Die initiale Schrittlänge von 0.8 m mit einer Standardabweichung σ_λ von 0.2 m wurde empirisch ermittelt. Die Abweichung des Kompass σ_θ wurde mit 15° höher gewählt als die empirischen Messungen ergaben, um die Verzögerung des Kompasses zu kompensieren und auch im Fall von stärkeren elektromagnetischen Störungen noch zuverlässige Ergebnisse zu erhalten. Die Korrektur mithilfe des Gyroskops stand bei der Testhardware in diesem Fall nicht zur Verfügung.

Über alle Pfade gemittelt ergab sich eine Genauigkeit des SIR Partikelfilters von 1.10 m und eine Standardabweichung von 0.86 m. Der durchschnittliche Fehler am Ende des Pfades war 0.71 m, was einem sehr guten Genauigkeitszuwachs im Vergleich zur Gesamtgenauigkeit entspricht. Dies ist dank Mapmatching und WLAN-Korrektur möglich, da so die Ungenauigkeit in der Längen- und Richtungsschätzung einzelner Schritte kompensiert werden.

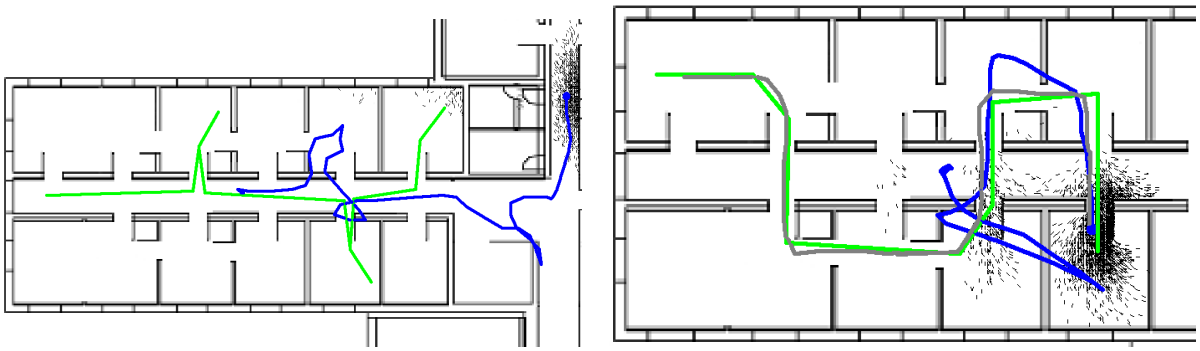
Im dem Fall, dass keine WLAN-Fingerprint-Informationen verfügbar sind, muss sich das System allein auf die Informationen des Dead Reckonings mit Mapmatching verlassen. Um die Abwesenheit von WLAN zu simulieren, wurden die Signalstärkeinformationen ignoriert und die Genauigkeit bei Initialisierung durch die drei verschiedenen Initialisierungsmöglichkeiten ermittelt. Diese sind eine grobe Anfangsposition, welche hier durch eine initiale WLAN-Position simuliert wird, aber genauso gut durch Bluetooth oder Funkzellenpositionierung erreicht werden kann, eine genaue Anfangsposition und eine unbekannte Anfangsposition. Davon ausgehend wurden nur die Vohersagephase sowie das Resampling durchgeführt und es wurde komplett auf die Korrekturphase verzichtet. Die Ergebnisse sind in Tabelle 5.1 festgehalten, wobei der mittlere Fehler, der Endfehler und die Standardabweichung über alle Tracks gemittelt wurden.

Tabelle 5.1: Überblick über Genauigkeit und Präzision

Initialisierung	**mittlerer Fehler**	**Std.Abweichung**	**Endfehler**
Gleichverteilung	6.30 m	3.02 m	2.30 m
Grobe WLAN-Startposition	3.70 m	2.06 m	2.23 m
Feste Startposition	1.33 m	0.78 m	1.06 m
Mit WLAN-Korrektur	1.10 m	0.86 m	0.71 m

Wie in Tabelle 5.1 gesehen werden kann, ist die Genauigkeit und Präzision mit der WLAN-Korrektur am höchsten. Hierbei sollte beachtet werden, dass die Tracks relativ kurz sind und die Umgebung aufgrund der offenen Türen nur wenige Restriktionen für die

Partikel-Bewegung bietet. Dies spiegelt sich in der großen mittleren Abweichung bei der Gleichverteilung wider. Der größte Einfluss auf den Endfehler kommt von einem einzelnen Pfad, der mehr als 6 m Abweichung von der tatsächlichen Endposition bei der Gleichverteilung und der groben WLAN-Position aufweist. Dies liegt daran, dass bei diesem Pfad keine Konvergenz erreicht werden konnte und ein Großteil der Partikel aufgrund von Gebäudesymmetrien einer falschen Hypothese folgt (vergleiche Abbildung 5.3a). In den anderen Fällen erreichte der Partikelfilter eine Konvergenz auf eine einzelne Hypothese vor dem Ende der Positionsbestimmung (vergleiche Abbildung 5.3b). Wie erwartet arbeitet der Algorithmus im Falle der bekannten Startposition am besten.



(a) Fehlerhafte Konvergenz ohne WLAN (b) Richtige Konvergenz ohne WLAN

Abbildung 5.3: Der echte Pfad (hellgrün), der geschätzte Pfad mit anfänglicher Gleichverteilung (dunkelblau) und der mit Backtracking berechnete Pfad (grau).

Aus den Ergebnissen kann geschlossen werden, dass der Algorithmus bei verfügbarer WLAN-Korrektur oder bekannter Startposition qualitativ hochwertige Positionsschätzungen liefert. Falls nur eine grobe oder gar keine Startposition bekannt ist, erreicht der Partikelfilter erst nach einer gewissen Zeitspanne seine Qualität, wenn durch Mapmatching nur noch ein einzelnes Cluster von Partikeln überlebt hat. Die hierfür benötigte Zeit hängt von der zurückgelegten Strecke, den Richtungsänderungen und der Gebäudesymmetrie ab und variiert in den Testläufen zwischen 15 und 30 Metern bei gleichverteilter Initialisierung, zwischen 10 und 20 Metern bei grob bekannter Anfangsposition. Nur in einem Fall konnte auch nach 42 Metern keine Konvergenz beobachtet werden, da am Ende der Aufnahme noch drei unbalancierte Cluster existierten, wie in Abbildung 5.3a zu sehen ist.

Die Ergebnisse sind in Abbildung 5.4 noch einmal auf die Genauigkeit in Bezug auf einen einzelnen Track zusammengefasst, um eine qualitative Aussage des Auftretens der Fehler machen zu können. Man kann erkennen, dass die Genauigkeit bei der WLAN-Korrektur im Bereich der Genauigkeit einer initial bekannten Position liegt. Dies liegt jedoch auch an der Form und Länge des Tracks. Bei längeren geraden Strecken oder größeren Freiflächen hält die WLAN-Korrektur die Partikelwolke zusammen und auf dem richtigen Weg. Ein Beispiel hierfür ist in Abbildung 5.5 zu sehen. Abbildung 5.5a zeigt eine deutlich größere Streuung der Partikel auf einem langgezogenen geraden Gang als Abbildung 5.5b, bei welcher die WLAN-Korrektur auf sonst identischen Daten aktiv war.

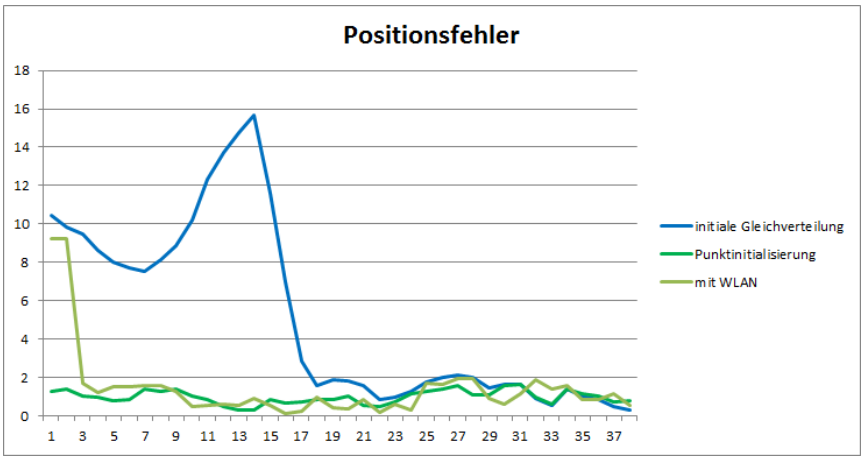
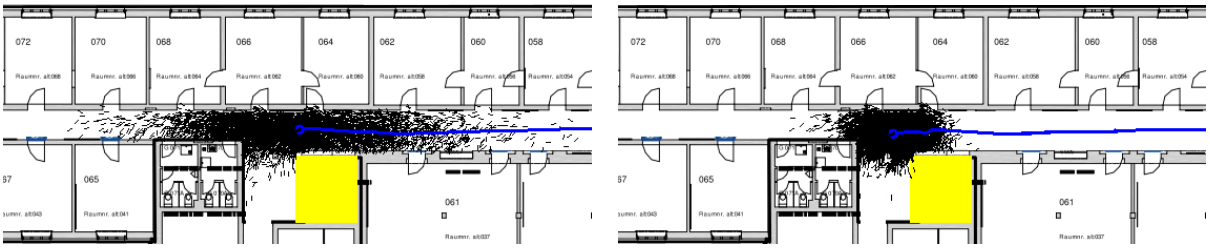


Abbildung 5.4: Genauigkeit des Partikelfilters bei Koppelnavigation und initialer Gleichverteilung im Vergleich zur Genauigkeit mit WLAN-Korrektur.



(a) Streuung der Partikel ohne WLAN-Korrektur (b) Streuung der Partikel mit WLAN-Korrektur

Abbildung 5.5: Unterschiedliche Streuung der Partikel, je nachdem ob die WLAN-Korrektur genutzt wird.

5.1.3 Zusammenfassung

Der Partikelfilter bietet durch die Kombination von WLAN-Fingerprinting als Messmodell und Dead Reckoning als Vorhersagemodell bereits von Beginn der Positionsbestimmung an eine deutliche Steigerung der Genauigkeit. Alternativ funktioniert der Filter auch ohne WLAN und somit ohne die Korrekturphase. Dies kann dann geschehen, wenn entweder kein WLAN oder keine kalibrierte Datenbank verfügbar ist. In diesem Fall startet man mit einer Gleichverteilung als Initialisierung und führt nur Vorhersagen auf Basis der Schritterkennung aus. Das Vorgehen entspricht dem bereits in Abschnitt 4.2.2 vorgestellten Dead Reckoning System, welches ohne Infrastruktur auskommt. Da die Korrekturphase jedoch die Genauigkeit und Präzision des Systems erheblich steigert, zu einer schnelleren Konvergenz der Partikel-Wolke führt und meist hilft, mehrere Modi zu vermeiden, wird die Nutzung des Systems mit WLAN empfohlen. Im nächsten Abschnitt wird zudem eine Erweiterung vorgestellt, welche durch Backtracking allein auf Basis des Dead Reckoning Systems eine qualitativ hochwertige Track-Historie erzeugt, die unter anderem für eine automatische WLAN-Fingerprint-Kalibrierung geeignet ist.

5.2 Erzeugung von Fingerprint-Datenbanken mithilfe von Backtracking

Die Reduktion des Kalibrierungsaufwands ist eine der größten Herausforderungen für den Einsatz von Fingerprint-Technologien zur Ortung. Um eine möglichst genaue Ortung zu erreichen, müssen möglichst viele Messungen an jedem Referenzpunkt gesammelt werden. Gerade bei Bayesischen Ansätzen sind Größenordnungen von 100 und mehr Messungen an jedem Punkt üblich, um eine möglichst gute Approximation der Signalstärkeverteilung an einem Punkt zu bekommen [148]. In [11] wurde vorgeschlagen, für verschiedene Uhrzeiten unterschiedliche Datenbanken anzulegen, so dass diese der Anzahl und Bewegung der Personen im Gebäude angepasst werden können. Zudem hat die Dichte der Referenzpunkte einen großen Einfluss auf die erreichbare Genauigkeit, wobei man im allgemeinen mit einer höheren Dichte eine genauere Positionsbestimmung erreichen kann.

In der Vergangenheit wurden viele Versuche unternommen, eine kalibrierungsfreie oder möglichst automatisch kalibrierende Lösung für WLAN-Fingerprinting zu finden. Die Arbeiten reichen dabei von der Nutzung von Ausbreitungsmodellen [10, 17] über automatische Nachkalibrierung unter Hinzunahme weniger Kalibrierungspunkte [132] bis zur automatischen Kalibrierung durch die Beacon Frames der installierten Access Points [6]. Die Nachteile hierbei sind einerseits in der Komplexität von guten Ausbreitungsmodellen gegeben, die eine Vielzahl an Faktoren mitberücksichtigen müssen. Dazu werden oftmals Ray-Tracing oder Ray-Casting Techniken eingesetzt, die Beschaffenheit der Wände einbezogen oder zumindest ein gewisser Teil an Daten gemessen, um die Gebäudeabhängigen Parameter eines Ausbreitungsmodells per Regression zu ermitteln [132]. Andererseits erreichen modellbasierte Ansätze selten die Genauigkeit einer aktuell kalibrierten Datenbank [10]. Andere Ansätze verwenden autonome Roboter [100] oder gleich Nutzer-generierte Eingaben [75], um eine initiale Datenbank zu erzeugen oder eine bestehende zu aktualisieren. Noch einen Schritt weiter gehen Ansätze, die Datenbanken automatisch basierend auf bei der Positionsbestimmung aufgezeichneter Sensordaten kalibrieren: Chintalapudi et al. stellen einen genetischen Algorithmus vor, der anhand von opportunistisch empfangenen Signalen ohne Kartenwissen und bei unbekannter Position der Access Points eine Fingerprint-Datenbank erzeugt [25]. Dieser Ansatz erlaubt zwar eine recht genaue und präzise Ortung, verursacht auf der anderen Seite aber einen hohen Berechnungsaufwand. Woodman und Harle stellen in [146] einen Ansatz vor, bei dem Positionsschätzungen eines Partikelfilters bei geringer Varianz genutzt werden, eine WLAN-Datenbank initial aufzubauen. Der Filter baut auf am Fuß getragenen Sensoren und Mapmatching mit einem Gebäudeplan auf und kann die Fingerprints für eine initiale Ortung nutzen. Zudem zeigen die Autoren, dass die erzeugte Datenbank ebenfalls ausreicht, um eine Gebiets-basierte Korrektur der Position mithilfe von probabilistischem WLAN-Fingerprinting zu ermöglichen, um Probleme bei Gebäudesymmetrien in ihrem Dead Reckoning Ansatz zu vermeiden. In [141] wird ein Backtracking Partikelfilter zu Positionsschätzung eingeführt. Es wird allerdings noch keine Anwendung für die durch Backtracking genaueren offline Positionsdaten genannt. Zu guter Letzt stellen Rai et al. mit Zee in [105] einen Ansatz vor, bei dem ebenfalls ein Partikelfilter basierend

auf Sensordaten eines Smartphones genutzt wird, um WLAN-Fingerprint-Datenbanken zu erzeugen. Dazu wird zusätzlich Backtracking eingesetzt, um eine größere Menge möglichst genauer Positionsschätzungen von jedem Nutzer zu erhalten. Allerdings wird dabei nur angesprochen, dass die durch Backtracking erworbenen genaueren Positionsdaten zur Kalibrierung genutzt werden können. Eine Empfehlung, wie diese Nutzung abläuft, fehlt jedoch.

Zeitgleich zu Zee [105] von Rai et al. wurde ein eigener Crowdsourcing-Ansatz entwickelt, welcher ebenfalls auf Dead Reckoning und Backtracking mit Smartphone Sensoren aufbaut, um eine WLAN-Fingerprint-Datenbank initial zu erzeugen und kontinuierlich aktuell zu halten [65]. Dazu wird der im vorangehenden Abschnitt vorgestellte Partikelfilter verwendet, der sich durch große Flexibilität und Effizienz auszeichnet und somit optimal für den Einsatz auf dem Smartphone geeignet ist. Im Folgenden wird auf die Verwendung des Backtrackings zur automatischen WLAN-Kalibrierung eingegangen und anschließend werden die Ergebnisse einer prototypischen Evaluation diskutiert.

5.2.1 Steigerung der Genauigkeit durch Backtracking

Möchte man Positionsdaten eines Ortungssystems für die Kalibrierung von Fingerprint-Datenbanken zu nutzen, so besteht das Problem, dass die Genauigkeit und Präzision des kalibrierenden Systems besser sein müssen, als die erreichbaren Werte des Fingerprinting-Systems. Man könnte meinen, dass das genaue System das ungenauere Fingerprinting-System überflüssig macht. Es gibt allerdings Fälle, in denen ein solches Szenario nicht nur denkbar, sondern sogar sinnvoll ist. Zum Beispiel kann ein Kalibrierungssystem in Form eines mit hochqualitativen Sensoren ausgestatteten Roboters ein WLAN-Fingerprinting System sicherlich kalibrieren, während die Vergabe der Roboter-Hardware an Nutzer zu teuer und unpraktisch wäre. Die erzeugte Datenbank kann nun aber von den Nutzern mit eigenen Geräten genutzt werden, um eine günstige und dennoch genügend genaue Positionsbestimmung durchzuführen.

Im hier geschilderten Fall ist eine recht gute Genauigkeit und Präzision der Positionsbestimmung durch das Dead Reckoning gegeben, wobei WLAN-Fingerprinting zur Steigerung der Genauigkeit und Präzision und zur zusätzlichen Stabilisierung über längere Zeiträume hinweg genutzt werden kann. Gerade wenn keine WLAN-Positionsbestimmung verfügbar ist, können die Eigenschaften des Partikelfilters dazu führen, dass sich mehrere Cluster von Partikeln bilden, die nach einem längeren Zeitraum aufgrund des Mapmatchings vermutlich zu einem einzelnen Cluster zusammenschmelzen. Die überlebenden Partikel sind einem Weg gefolgt, welcher mit hoher Wahrscheinlichkeit dem tatsächlichen Pfad des Nutzers entspricht. Daher weist der Pfad nach erfolgtem Backtracking eine deutlich höhere Genauigkeit und Präzision auf als zuvor. Diese höhere Genauigkeit beim kalibrierenden System sollte zu einer ebenso erhöhten Genauigkeit bei einem nachkalibrierten System führen. Zu diesem Zweck wird zur Ermittlung der tatsächlichen Trajektorie des Nutzer Backtracking eingesetzt, wodurch für jedes Partikel zu einem beliebigen Zeitpunkt die komplette Bewegung bis zur Initialisierung nachvollzogen werden kann. Abbildung 5.6 und Abbildung 5.7 zeigen den Vorteil von Backtracking im Vergleich zu einem herkömmlichen Partikelfilter.

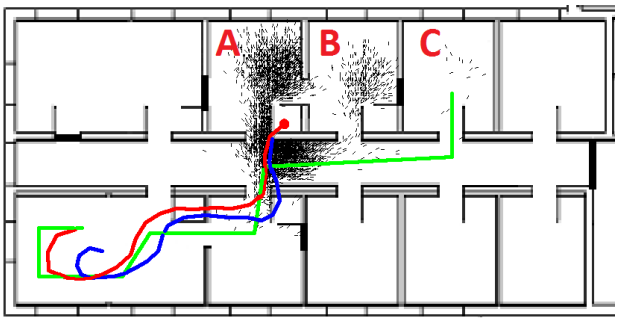


Abbildung 5.6: Tatsächlicher Pfad (hellgrün), Schätzung SIR Partikelfilter (dunkelblau), Schätzung mit Backtracking (rot) und Partikel Verteilung während der Positionierung.

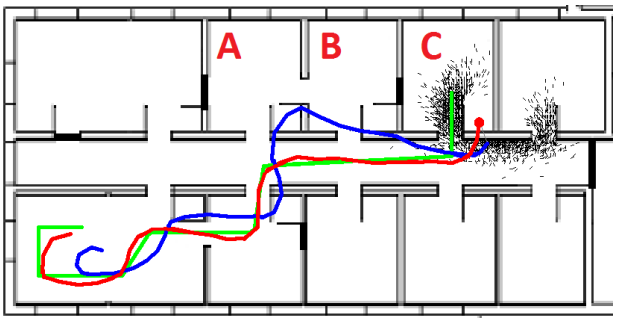


Abbildung 5.7: Tatsächlicher Pfad (hellgrün), Schätzung SIR Partikelfilter (dunkelblau), Schätzung mit Backtracking (rot) und Partikel Verteilung während der Positionierung kurze Zeit nach Abbildung 5.6.

Die dunkelblaue Linie beschreibt den Pfad, der vom Partikelfilter geschätzt wird. Hierbei werden alle Partikel in den Pfad mit einbezogen, welche an der Wand in Raum A oder B sterben (siehe Abbildung 5.6). Die rote Linie zeigt in Abbildung 5.7 den durch Backtracking aktualisierten Pfad, der alle auf dem Weg entfernten Partikel ignoriert und somit deutlich näher an dem tatsächlichen gelaufenen Pfad in hellem Grün liegt, der in Raum C endet.

Im Fall eines SIR Partikelfilters verschwinden Partikel mit einem geringen Gewicht und Partikel mit einem hohen Gewicht werden in mehrere Partikel aufgeteilt. Um dennoch die Übersicht über die vorhergehenden Partikel zu behalten, wurde das Backtracking durch Zeiger auf den Vorgänger umgesetzt. Um eine maximale Auflösung zu erhalten, wurde zudem für jedes nach der Resampling-Phase bestehende Partikel ein neues erzeugt. Im Fall von einer Aufteilung eine Partikels verweisen alle Nachfolger auf das vorhergehende Partikel. Der Vorgänger wird aus der Menge der aktiven Partikel entfernt. Partikel, die sterben, haben keine weiteren Kinder und können dementsprechend mitsamt aller Vorgänger komplett gelöscht werden. Zusätzlich zu den Partikeln wird jede Signalstärkemessung im Verlauf der Positionsbestimmung so gespeichert, dass die Messung der Menge von Partikeln zugeordnet werden kann, die zum Zeitpunkt der Messung aktiv waren.

5.2.2 Erzeugung und Kalibrierung

Für die Kalibrierung von WLAN-Fingerprint-Datenbanken muss man zwischen zwei unterschiedlichen Szenarien unterscheiden. Entweder wird eine veraltete Datenbank aktualisiert, oder es muss eine komplett neue Datenbank erzeugt werden, falls keinerlei Informationen über die WLAN-Infrastruktur verfügbar sind. Im letzteren Fall kann nur Dead Reckoning ohne WLAN-Korrektur eingesetzt werden, wobei Positionsinformationen und Signalstärkemessungen kombiniert werden, um eine neue Datenbank zu erzeugen. Im ersten Fall könnten Änderungen an der Umgebung dazu führen, dass die Daten aktualisiert werden müssen. In diesem Fall kann die Fingerprint-Datenbank immer noch für eine grobe Positionierung genutzt werden. Aktualisierte Daten müssen langsam in das System einfließen, um eine fehlerhafte Kalibrierung aufgrund von Messfehlern vorzubeugen. Dieses Problem kommt auch in vergleichbaren Szenarien des unüberwachten Lernens zum Tragen und kann dazu führen, dass die Trainingsdatenbank für die Positionsbestimmung nutzlos wird oder gar zu schlechteren Ortungsergebnissen führt als ohne Verwendung vom WLAN-Korrektur.

Dieses Problem wird in [65] statistisch gelöst, indem zwischen alter (rssi_{old}) und neuer (rssi_{new}) Signalstärke entsprechend einem Gewicht ω interpoliert wird:

$$\text{rssi} = \omega \cdot \text{rssi}_{\text{new}} + (1 - \omega) \cdot \text{rssi}_{\text{old}} \tag{5.4}$$

Auch bei Bayesischen Fingerprint-Ansätzen kann ähnlich vorgegangen werden. Da zusätzlich zum Mittelwert die Standardabweichung σ für jeden sichtbaren Access Point gespeichert wird, kann die folgende Gleichung (5.5) zur Aktualisierung der bisherigen Standardabweichung σ_{old} genutzt werden:

$$\sigma = \sqrt{(1 - \omega) \cdot \sigma_{\text{old}}^2 + \omega \cdot (\text{rssi} - \text{rssi}_{\text{new}})^2} \tag{5.5}$$

Das Gewicht ω bestimmt, wie schnell die Datenbank an neu gemessene Signalstärkewerte adaptiert wird. Signale von neuen Access Points werden ohne Gewichtung übernommen, da diese möglicherweise neu aufgestellt wurden. Die Signalstärke von Access Points in einem Fingerprint, die in der aktuellen Messung nicht auftauchen, werden kontinuierlich um $(\text{rssi}_{\text{old}} - (-100))/\omega$ verringert und ganz entfernt, wenn die Signalstärke unter -99dBm fällt, was unterhalb der Empfangsmöglichkeiten der meisten Smartphones liegt.

Im Gegensatz zu anderen Systemen werden hier Fingerprints als die Punkte im Kartenraum modelliert, an denen die Signalstärke gemessen wurde. Zusätzlich wird die Blickrichtung des Nutzers, gemessen durch den Kompass des Smartphones gemessen, zu jedem Fingerprint gespeichert. Damit kann die Abschottung des menschlichen Körpers im System durch das Messmodell aus Abschnitt 4.3.3 kompensiert werden.

Im Falle einer neuen Messung muss festgelegt werden, welche Fingerprints nach Gleichung (5.4) aktualisiert werden. Als einfache Lösung wird vorgeschlagen, dass ein Fingerprint dann aktualisiert wird, wenn er sich näher als eine bestimmte Entfernung zu der Positionsmessung der für die Kalibrierung genutzten Messung befindet. Zudem muss er in Richtung derselben Gebäudeachse zeigen, wie die Positionsmessung. Dabei sind Abweichungen bis zu 45° zulässig. Falls kein solcher Fingerprint existiert, wird eine neuer

Fingerprint an der Stelle der Positionsmessung mit der entsprechenden Ausrichtung erzeugt. Auf diese Art und Weise wird ein virtuelles Grid erzeugt, dessen Dichte leicht über den Distanz-Grenzwert reguliert werden kann. Bei der Aktualisierung einer bestehenden Datenbank wird so mit jedem Track nur ein kleiner Teil der Datenbank aktualisiert, was zu einer sehr langsamen Adaption an wechselnde Umgebungen führt und eine große Zahl von Nutzern zur schnellen Adaption benötigt. Je nach Anwendungsszenario kann es daher sinnvoll sein, alle umliegenden Fingerprints ähnlich zu [132] zu aktualisieren.

Zuletzt soll noch bemerkt werden, dass sich der vorgeschlagene Algorithmus auch auf Fälle anwenden lässt, in denen Fingerprints nicht mit Punktkoordinaten sondern ganzen Flächen verbunden sind, beispielsweise für gelabelte und trainierte Daten eines Naive Bayesischen Klassifizierers, wie er in Abschnitt 4.3.2 vorgestellt wurde. In diesem Fall müssen jedoch zu einer guten Approximation der entsprechenden Verteilungen möglichst viele Daten für jedes Label gesammelt werden.

5.2.3 Gesamtarchitektur

Die Architektur des Partikelfilters lässt sich nun wie folgt zusammenfassen (siehe auch Abbildung 5.8): Die Sensordaten aus Gyroskop und Kompass werden genutzt, um eine Richtungsschätzung zu ermitteln, und die Daten des Beschleunigungssensors werden auf Schrittmuster analysiert. Bei einem erkannten Schritt fließt die aktuelle durchschnittliche Schrittlänge und die Schrittfrequenz in die Schrittlängenschätzung mit ein (vergleiche Abschnitt 4.2.1) und wird zusammen mit der Richtungsschätzung an den Partikelfilter gemeldet. WLAN-Signalstärkemessungen werden durch den gewichteten kNN-Algorithmus mit Richtungsfilter (siehe Abschnitt 4.3.3) in eine Positionsschätzung umgewandelt, die einer Normalverteilung entspricht, deren Parameter aus der Position der nächsten Nachbarn (analog zu 4.3.5) berechnet wurden. Bei der Vorhersage durch Schritterkennung fließen die Kartendaten mit ein, um die Plausibilität einer Positionsänderung zu überprüfen und Partikel, deren Bewegung Wände durchquert, zu entfernen. Die Partikel werden in der Backtracking Komponente zusammen mit der Historie der Vorgänger abgelegt, um offline genauere Positionsinformationen herzuleiten. Zudem werden die WLAN-Signalstärkenformationen mit einem Zeitstempel abgelegt, um sie anschließend mit den genauen Positionsdaten verknüpfen zu können. Diese Daten werden zudem genutzt, um eine neue WLAN-Fingerprint-Datenbank zu erzeugen oder eine bereits bestehende Datenbank aktuell zu halten.

5.2.4 Evaluation und Diskussion der Ergebnisse

Um die vorgeschlagene Kalibrierung von WLAN-Datenbanken durch Backtracking zu evaluieren, wird zunächst die Genauigkeit des Backtrackings untersucht. Dazu wird dieselbe Testumgebung mit denselben Daten verwendet, wie in Abschnitt 5.1.2. Anschließend wird auf die eigentliche Kalibrierung eingegangen und die beiden Szenarien der Erzeugung einer neuen und der Aktualisierung einer veralteten Datenbank untersucht. Abschließend folgt eine Diskussion der erzielten Ergebnisse.

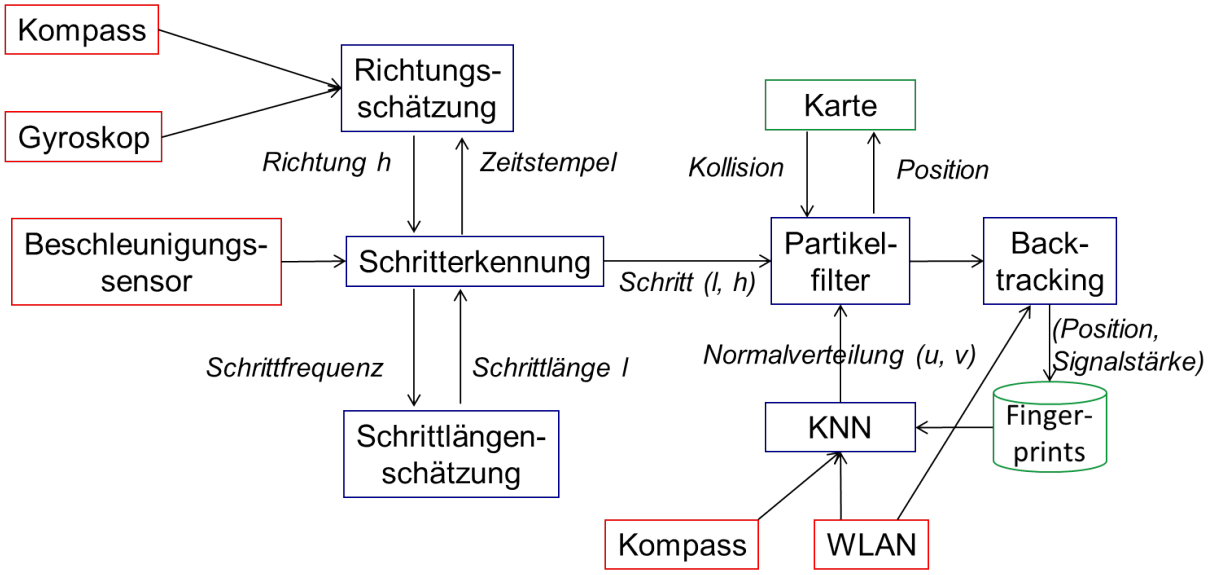


Abbildung 5.8: Überblick über die Architektur des Backtracking Partikelfilters.

Evaluation des Backtrackings

Um eine WLAN-Datenbank automatisch zu kalibrieren, benötigt man ein genaues und präzises System zu Positionsbestimmung. Solange entweder WLAN-Korrektur verfügbar ist, die Startposition bekannt ist oder der Nutzer sich einige Zeit durch das Gebäude bewegt hat, erfüllt das vorgestellte System die Anforderungen. Für den ersten Fall ist jedoch eine kalibrierte Trainingsdatenbank notwendig, womit auf eine Neukalibrierung verzichtet werden kann. Der zweite Fall ist insofern interessant, da zur Kalibrierung nur noch die Startposition eingegeben werden muss und somit eine deutliche Aufwandsreduktion im Vergleich zur manuellen Erstellung einer Trainingsdatenbank gegeben ist. Möchte man ganz ohne Aufwand eine Datenbank erzeugen und aktuell halten, so muss man in den meisten Fällen warten, bis ein einzelnes Cluster von Partikeln überlebt. In diesem Fall sind alle im bisherigen Verlauf generierten Informationen für die Kalibrierung wertlos und können aufgrund der geringen Genauigkeit für viele ortsbezogene Dienste nicht direkt genutzt werden.

Um dieses Problem zu lösen, wird das beschriebene Konzept des Backtrackings angewandt. Dazu wird an dieser Stelle die Genauigkeit und Präzision des SIR Partikelfilters mit der des Backtracking Partikelfilters verglichen. Man muss jedoch beachten, dass Backtracking die genauesten Informationen erst zum Ende der Positionsbestimmung liefert und somit nicht als Online-Tracking-Algorithmus eingesetzt werden kann. Tabelle 5.2 zeigt die Ergebnisse bezüglich mittlerer Abweichung und Standardabweichung gemittelt über alle Pfade, die am Ende des Pfades auf ein Cluster konvergiert sind. Daher sind hier nur 5 von 6 Pfaden einbezogen worden.

Wie man sehen kann, ist die mittlere Abweichung sogar niedriger als die Endabweichung des SIR Partikelfilters. Man bemerke zudem, dass sogar die Genauigkeit und Präzision im

Tabelle 5.2: Überblick über die Genauigkeit und Präzision von Backtracking

Initialisierung	mittlerer Fehler	Std.Abweichung
Gleichverteilung	1.28 m	0.62 m
Grobe WLAN-Startposition	1.23 m	0.46 m
Feste Startposition	0.94 m	0.47 m
Mit WLAN-Korrektur	0.66 m	0.38 m

Fall der verfügbaren WLAN-Korrektur gesteigert werden konnte. Ein Beispiel für einen Backtrack ist in Abbildung 5.3b zu sehen. Die ermittelten Genauigkeiten werden als genau genug für die automatisierte WLAN-Kalibrierung angesehen. Unabhängig von der Initialisierung können durch das Backtracking die vollständigen Pfade von Nutzern für diesen Zweck verwendet werden und nicht mehr die Teilpfade ab einer Konvergenz der Partikelwolke.

Automatisierte WLAN-Kalibrierung

Der Algorithmus für automatisierte WLAN-Kalibrierung wird in zwei unterschiedlichen Szenarien getestet. Im ersten wird eine alte und überholte Datenbank mit Hilfe von zwei Kalibrierungspfaden aktualisiert, wobei beim Aufnehmen der Pfade jeder Raum in der Testumgebung jeweils einmal besucht wurde. Die Zeit für die Erzeugung der beiden Tracks betrug zusammen ca. 200 Sekunden. Im zweiten Szenario wird eine Datenbank komplett neu aus den beiden Pfaddaten erzeugt. Dazu wurden wiederum sechs Testpfade aufgezeichnet, die den Pfaden aus Abbildung 5.2 entsprechen.

Die alte Datenbank war ungefähr ein halbes Jahr vor der Aufnahme der Pfaddaten erstellt worden. In der Zwischenzeit war eine Trennwand zwischen zwei Räumen abgerissen, die Abdeckung der Decke im Flur entfernt und ein paar Access Points ausgewechselt worden. Dementsprechend groß waren die Abweichungen in der Datenbank. Zudem waren in der alten Datenbank keine Fingerprints in zwei Räumen des Testgebiets aufgenommen worden, durch die allerdings zwei Testpfade führten. Mit der alten Datenbank kann das System nur noch einen mittleren Fehler mit WLAN-Korrektur von 3.31 m aufweisen, der mittlere Endfehler entspricht 2.05 m und die Standardabweichung 1.58 m. Diese Werte sind jeweils deutlich schlechter als die Performanz mit einer aktuellen manuell kalibrierten Datenbank.

Zur Aktualisierung der alten Datenbank wird $\omega = 0.33$ gewählt und beide Kalibrierungspfade werden mit einer Gleichverteilung initialisiert. Zudem wird nur die Vorhersagephase mit Resampling durchgeführt, also Dead Reckoning, und der Backtracking-Algorithmus für die Positionsschätzungen. Nach der Kalibrierung ist die Genauigkeit und Präzision des Systems mit WLAN-Korrektur leicht erhöht. Der mittlere Fehler beträgt 3.13 m, der Endfehler 1.67 m und die Standardabweichung 1.48 m, was immer noch deutlich unter den Möglichkeiten eines kalibrierten Systems liegt. Die Ergebnisse zeigen jedoch, dass man bereits mit zwei Kalibrierungspfaden und dem vorgestellten Algorithmus zur Ak-

tualisierung eine geringfügige Verbesserung der Genauigkeit und Präzision erreichen kann.

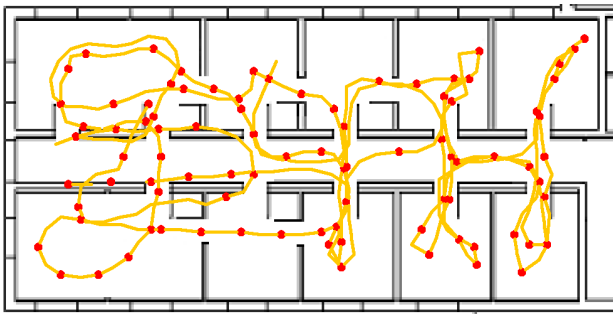


Abbildung 5.9: Das Ergebnis des Backtrackings von beiden Kalibrierungspfaden (orange) und die automatisch erzeugten Fingerprints (rot).

Um die Funktionalität des Ansatzes zu zeigen, wird auch eine komplett neue Fingerprint-Datenbank erzeugt. Zu diesem Zweck werden nur die beiden in Abbildung 5.9 gezeigten Pfade hergenommen, wobei beide mit einer Gleichverteilung im Testgebiet initialisiert werden. Zur Positionsbestimmung wird lediglich die Dead Reckoning Komponente des Partikelfilters genutzt. Anhand der erzeugten Datenbank wird die Performanz bezüglich der gleichen sechs Pfade getestet, wie im Fall der Aktualisierung einer alten Datenbank. Der durchschnittliche Fehler sinkt auf 2.03 m, der mittlere Endfehler auf 0.87 m und die Standardabweichung auf 1.25 m. Insbesondere der geringe Endfehler zeigt, dass die vorgestellte Methode genügend genau ist, um automatisiert und mit geringem zeitlichen Aufwand WLAN-Fingerprint-Datenbanken zu erzeugen. Tabelle 5.3 fasst die Ergebnisse der automatisierten WLAN-Kalibrierung zusammen.

Tabelle 5.3: Überblick über die Genauigkeit und Präzision der Kalibrierung

Szenario	**mittlerer Fehler**	**Standardabweichung**	**Endfehler**
veraltete Datenbank	3.31 m	1.58 m	2.05 m
aktualisierte Datenbank	3.13 m	1.48 m	1.67 m
Neu erzeugte Datenbank	2.03 m	1.25 m	0.87 m

Mit einer Arbeit, die etwas mehr als drei Minuten gedauert hat, wurde eine Trainingsdatenbank erzeugt, welche eine genügend hohe Genauigkeit aufweist, um als Korrekturgröße in einen Partikelfilter einzufließen. Die manuelle Erzeugung der Trainingsdatenbank erforderte im Gegensatz dazu 2 Stunden ermüdende Arbeit, bei einer dafür aber auch deutlich genaueren Positionsbestimmung. Dennoch macht der vorgestellte Ansatz die Anwendung von Positionsbestimmung innerhalb von großen und komplexen Gebäuden mit günstigen Smartphone-Sensoren zu einem tragbaren Aufwand möglich und erlaubt sogar die vollständige automatische Aktualisierung mithilfe von Nutzer-generierten Daten.

Diskussion und Bewertung

Im vorhergehenden Abschnitt wurden verschiedene Ergebnisse hinsichtlich der Performanz des Backtrackings und der WLAN-Kalibrierung durch den Partikelfilter vorgestellt. In diesem Abschnitt werden die Ergebnisse und deren Einfluss auf Anwendungsszenarien diskutiert und weitergehende Überlegungen dargestellt.

Der Hauptbeitrag aus [65] ist die Anwendung eines Backtracking Partikelfilters für die automatisierte WLAN-Kalibrierung. Dies ist a- bgesehen von Zeit-unkritischen Tracking-Szenarien - ein Anwendungsfall, in dem keine Echtzeit-Positionsbestimmung notwendig ist. Daher kann die komplette Historie der Positionsdaten genutzt werden, um einer bessere Positionsschätzung über den ganzen Zeitraum der Positionsbestimmung hinweg zu erhalten. Im Falle des Partikelfilters muss dies nicht immer zu 100 % korrekten Pfaden führen, da auch im Falle eines einzelnen überlebenden Clusters die Partikel verschiedene Wege genommen haben können. Dies kann jedoch durch Clustererkennung festgestellt werden und beispielsweise im Falle von mehreren Clustern die WLAN-Kalibrierung ausgesetzt werden. Im Fall mehrerer Cluster kann eine zusätzliche WLAN-Korrektur die Genauigkeit erheblich steigern, was jedoch gerade im Fall einer veralteten Datenbank oftmals einen negativen Einfluss auf die Positionierungsgenauigkeit hat. Aus diesem Grund wurde die Designentscheidung getroffen, zur Kalibrierung nur Dead Reckoning Informationen zu nutzen und auf die WLAN-Korrektur zu verzichten.

Um eine Trainingsdatenbank aktuell zu halten, benötigt der vorgestellte Ansatz viele Messungen innerhalb eines kurzen Zeitraums. Daher bietet sich für diesen Anwendungsfall ein System mit Nutzer-generierten Pfaden an, welches dem Nutzer Positionsbestimmung anbietet, die Daten jedoch im Nachhinein zur Kalibrierung der Datenbank einsetzt. In diesem Fall ist es sinnvoll, anstelle einer leeren Datenbank eine automatisch erzeugte Datenbank zu nutzen, wobei die Erzeugung wahlweise von einem Positionsbestimmungs-Anbieter mit dem vorgestellten Algorithmus oder durch Berechnung anhand von Ausbreitungsmodellen durchgeführt wird. Diese Daten werden anschließend von Nutzern automatisch weiter verfeinert und verbessert.

Die vorgestellten Ergebnisse wecken die Hoffnung auf ein kostengünstiges, einfaches und dennoch genaues System zur Positionsbestimmung. Eine abschließende Evaluation bezüglich der in Abschnitt 4 vorgestellten Eigenschaften neben Genauigkeit und Präzision sieht dementsprechend folgendermaßen aus:

- Die *Abdeckung*, die durch das System erreicht wird, ist aufgrund der flexiblen Initialisierung und der Möglichkeit, ohne WLAN und die Korrekturphase zu arbeiten, bei 100 %. Dabei kann jedoch die Genauigkeit zu Beginn der Positionsbestimmung leiden, weswegen für den optimalen Einsatz die Abdeckung von der Abdeckung des WLAN-Fingerprint-Systems abhängt.

- Die *Anschaffungskosten* sind gering, sämtliche Komponenten können auf dem Smartphone laufen. Im Falle der automatischen Kalibrierung ist jedoch eine Serverkomponente zumindest für die zu kalibrierende Trainingsdatenbank sinnvoll, wodurch geringe Anschaffungskosten verursacht werden.

- Die *Betriebskosten* entstehen allein durch den optionalen Betrieb der Serverkomponente.
- An sich wird keine *Infrastruktur* benötigt, wobei die Genauigkeit des Systems steigt, wenn eine WLAN-Infrastruktur zur Korrektur zur Verfügung steht. Hierbei reicht jedoch eine möglicherweise für Datenübertragungszwecke bereits verfügbare Infrastruktur vollkommen aus.
- Die *unterstützte Nutzerzahl* wird lediglich durch die Kapazität der optionalen Serverkomponente beschränkt und lässt sich gut skalieren.
- Der *Ergebnistyp* entspricht einer absoluten, geometrischen und deterministischen Positionsangabe.
- Die *Verfügbarkeit* ist gegeben, da heutige Smartphones grundsätzlich mit WLAN, Beschleunigungssensoren und einem digitalen Kompass ausgestattet sind.
- Das System weist eine große *Zuverlässigkeit* auf, da mit unbekannten, fehlenden und teilweise fehlerhaften Daten umgegangen werden kann.
- Die *Update-Rate* liegt zwischen 0,3 und 1 Sekunden für die Positionsbestimmung, je nachdem, welche Phase gerade ausgeführt wird.
- Ein *Schutz der Privatsphäre* ist nur dann gegeben, wenn das System vollständig auf dem Smartphone läuft und keine Kalibrierungsdaten in einer zentralen Serverkomponente gesammelt werden. Ist letzteres der Fall, können Nutzer selbstverständlich getrackt werden.

5.3 Plattform zur dynamischen Kombination verschiedener Sensoren

Die bisher vorgestellten Verfahren zur Positionsbestimmung gehen zum Großteil von der ständigen Verfügbarkeit bestimmter Sensoren aus. Oftmals kommt es jedoch zu Ausfällen oder Nicht-Verfügbarkeit bestimmter Sensoren oder es werden neue möglicherweise bessere Sensoren lokal verfügbar. Als Beispiel am Smartphone kann man die Verfügbarkeit von WLAN-Fingerprinting sehen. Die WLAN-Schnittstelle steht zwar die ganze Zeit über zur Verfügung, aber ob auch Signale von WLAN Access Points empfangen werden oder gar Signale empfangen werden, die eine Positionsbestimmung mithilfe der Datenbank ermöglichen, ist nicht überall sichergestellt. Betritt ein Nutzer ein Gebäude, in dem eine WLAN-Infrastruktur vorhanden ist und bereits für Positionierungszwecke ausgemessen wurde, so wird auf einmal eine neue Möglichkeit zur Positionsbestimmung verfügbar. Andere Beispiele sind GPS, aber auch spezielle Positionierungssysteme. Beispielsweise können stationäre Kamerasysteme in Gebäuden an bestimmte Orten verfügbar sein und weniger geeignete Verfahren ablösen. Zudem wurden in der Vergangenheit eine Vielzahl an verschiedensten

Verfahren und Technologien zur Positionsbestimmung entwickelt, und auch die Anforderungen an diese Verfahren können je nach Anwendungsfall variieren.

Genau auf diese Lücke zielt die Plattform zur dynamischen Kombination verschiedener Sensoren [63]. Diese wählt aus eine Menge verfügbarer Sensoren sowie Systemen und Algorithmen zur Positionsbestimmung die bestmögliche Kombination in Bezug auf verschiedene Kriterien wie Energieverbrauch, Genauigkeit oder Update-Rate aus. Zu diesem Zweck verfügt die Plattform über einen Sensor-Discovery Mechanismus, eine Komponente zur Bereitstellung von Umgebungsmodellen und dient als allgemeiner Zugangspunkt für Positionsbestimmungsdienste aller Art. Für den Zweck der Kommunikation zwischen den Sensoren und Komponenten der Plattform wurde eine Sensor-Beschreibungssprache entwickelt, welche nicht nur physikalische Sensoren beschreiben kann, sondern ebenfalls ganze Positionierungssysteme oder Algorithmen zur Sensor Fusion. In diesem Sinne wird in diesem Abschnitt ein Sensor allgemein als Einheit definiert, die einen physikalischen Effekt misst oder die Messung eines solchen weiterverarbeitet. Dadurch werden auch Sensorfusionsalgorithmen und Ortungssysteme als Sensoren bezeichnet, die aus verschiedenen Eingangsdaten eine Position berechnen.

Im Folgenden wird zunächst ein Überblick über bestehende Plattformen und bekannte Sensorbeschreibungssprachen gegeben, anschließend PositioningML, eine generische Sprache zur Beschreibung von Sensoren zur Positionsbestimmung, eingeführt und die Komponenten der Plattform beschrieben. Zuletzt folgt ein Anwendungsbeispiel als Machbarkeitsbeispiel, in welchem eine Plattform und Sprache zur Umsetzung von nahtloser Positionsbestimmung eingesetzt wird.

5.3.1 Plattformen für hybride Ortung und Sensorbeschreibungssprachen

In der Vergangenheit wurden verschiedene Ansätze für Ortungsplattformen vorgestellt. Einer der ersten Ansätze zur Ordnung und Strukturierung von Systemen zur Positionsbestimmung ist der Location Stack von Hightower et al. [50]. Die Autoren benennen Standards zur Kombination von Messwerten unterschiedlicher Sensorquellen und schlagen eine Schichtenarchitektur für die Organisation einer Plattform vor. Die unterste Schicht wird von den Sensoren gebildet, welche aus Hardware- und Software-Komponenten bestehen können. Darauf setzt die Messdatenschicht auf, welche die Rohdaten in kanonische Messdatentypen, wie beispielsweise Entfernungsmessungen, Winkelmessungen oder Positionsmessungen, überführt. Aufbauend auf diese Typen wird auf der nächsten Schicht eine mit Zeitstempel versehene probabilistische Zustandsschätzung ermittelt. Alle weiteren Schichten beschäftigen sich nicht mit positionsrelevanten Informationen und wurden in der Umsetzung des Location Stack in [39] nicht berücksichtigt. In der Umsetzung sorgt ein Partikelfilter für die Fusion von verschiedenen Positionsschätzungen unterschiedlicher Sensoren mit einem Bewegungsmodell. Obwohl der Location Stack wertvolle Informationen darüber liefert, wie Ortsinformationen in Anwendungen integriert werden, so bietet die Arbeit trotz der Umsetzung eher ein abstraktes Konzept als eine Plattform, da keine Kom-

munikationsmittel oder Schnittstellen angegeben werden. Unsicherheiten, die beispielsweise durch Messfehler verursacht werden, werden zwar bis an die Anwendungsschicht weitergereicht, allerdings wird auch hierzu kein spezielles Format festgelegt.

Ranganathan et al. stellen in [106] eine Middleware vor, die verschiedene Technologien zur Positionsbestimmung vereint und ein hybrides Umgebungsmodell mit einer symbolischen und geometrischen Beschreibung der realen Welt bietet. Dabei werden Adapter eingesetzt, um Rohdaten in eine gemeinsame Repräsentation zu überführen. Diese wird über eine Schnittstelle an einen Ortungsdienst weitergegeben, welcher die Informationen von verschiedenen Sensoren über einen geometrischen Ansatz zu einer probabilistischen Positionsschätzung verbindet. Damit ist die Architektur ähnlich zum Location Stack, wobei die Adapter den Anschluss von beliebigen Sensoren an die Middleware erlauben. Einzig die Einschränkung auf einen geometrischen Ansatz scheint ein Nachteil zu sein, da so performante Algorithmen wie Kalman oder Partikelfilter von vornherein ausgeschlossen werden.

MagicMap [20] ist ein Projekt, das auf eine Plattform für kooperative Positionsbestimmung abzielt. Ursprünglich wurde die Plattform für WLAN entworfen, allerdings wird zum jetzigen Zeitpunkt eine Vielzahl von Radiofrequenzverfahren zur Ortung und inertialen Sensoren unterstützt. Basierend auf der empfangenen Signalstärke wird die Entfernung eines Sensors zum Sender bestimmt. Um Hardware-abhängige Fehler zu vermeiden, wird die Entfernungsschätzung normalisiert und anschließend für Lateration genutzt [20]. Zur Positionsbestimmung kann zudem noch ein Partikelfilter mit einem Bewegungsmodell hinzugezogen werden, welches auch durch inertiale Daten unterstützt werden kann. Dennoch ist das System auf bestimmte Datenquellen und Algorithmen zur Positionsbestimmung angewiesen. Eine einheitliche Beschreibung von Ortungsmethoden, deren Fähigkeiten und Unsicherheitsmodellen wird nicht gegeben.

In [18] stellen Bohn und Vogt eine Sensorbeschreibung vor, welche klassische Beschreibungen von Sensoren um ganze Positionierungssysteme erweitert. Das bedeutet, dass in diesem Fall ein Sensor durchaus andere Messdaten als Eingangsgröße benötigt. Messungen werden durch einen Ereignis-getriebenen Mechanismus bei Eingang im System verarbeitet, wodurch auch Daten von neuen Sensoren im System berücksichtigt werden können. Die Sensor Fusion findet jedoch nur auf der obersten Ebene statt, was bedeutet, dass lediglich bereits geschätzte Positionen verarbeitet werden und keine unterschiedlichen Sensoren zu einer Positionsschätzung verwendet werden können. In diesem Fall kommt ein Zell-basiertes Verfahren zum Einsatz, das erst für jeden Sensor die Aufenthaltswahrscheinlichkeit für jede Zelle berechnet, diese über alle Sensoren aufaddiert und anschließend normalisiert. So bietet die Gitterstruktur schließlich eine Annäherung an eine Dichtefunktion.

Ein aktueller Ansatz auf diesem Gebiet ist MapUme [91], eine Middleware für ortsbezogene Anwendungen von Najib et al.. MapUme bietet eine Plattform für Daten verschiedenster Sensoren, Sensor Fusions Mechanismen und einen Ortungsdienst, der sowohl mit geometrischen als auch symbolischen Koordinaten umgehen kann. Die Schichtenarchitektur erinnert stark an den Location Stack [50], wobei hier verschiedene Aufgaben auch auf mehrere physikalische Maschinen verteilt werden können. Um neue Sensoren oder Fusions-Mechanismen umzusetzen, müssen Schnittstellen in der Messdaten-Komponente oder in der Fusions-Komponente implementiert werden. Dieser Ansatz ist aufgrund der Erweiter-

barkeit besonders vielversprechend, jedoch wird im Gegensatz dazu in [63] ein stärkerer Fokus auf die Dynamik verschiedener Sensoren, Methoden und wechselnder Nutzerkriterien gelegt.

Zuletzt soll ein kurzer Überblick über bestehende und bekannte Sensorbeschreibungssprachen gegeben werden. Eine der ersten Initiativen aus dem Bereich war das Projekt Sensor Web Enablement (SWE) [109] unterstützt von dem Open Geospatial Consortium (OGC). Das Ziel des Projekts ist das zur Verfügung stellen von beliebigen Sensoren über das Internet, um beispielsweise Web-basierte Sensornetze realisieren zu können. Dabei wurden Standards wie SensorML [19] und Observations & Measurements (OM) [27] definiert. SensorML ermöglicht die Definition und Beschreibung von Sensor Metadaten, Messprozessen und Transformationen der gemessenen Daten. Es unterstützt bereits eine Bewertung der Güte von Sensoren und Sensordaten. OM spezifiziert ein Modell für die Domänen-unabhängige Repräsentierung von geometrischen oder temporalen Messdaten und ist dementsprechend besonders für den Austausch von Messdaten geeignet.

Eine andere Sprache, die ein Kommunikationsmodell für Ortungsdienste beschreibt, ist die Sensor Fusion Modeling Language [116]. Diese ermöglicht die Beschreibung physikalischer und funktionaler Eigenschaften von Ortungssystemen inklusive Sensoren, Positionen und Algorithmen zur Sensor Fusion. Des Weiteren wird Unsicherheit von Positionsschätzungen durch die Angabe von Genauigkeitswerten oder Konfidenzintervallen gegeben. Die Sprache setzt den Schwerpunkt vor allem auf die Beschreibung von Ausgangsdaten und lässt bislang die geforderten Eingangsdaten für die einzelnen Ortungssysteme weitgehend außer Acht.

Der Sensor Abstraction Layer [36] erlaubt es schließlich, heterogene Sensorquellen hinter einer gemeinsamen Schnittstelle zu verbergen. Sensor Beschreibungen basieren auf SensorML, allerdings wurden verschiedene Dienstzugangspunkte und bestimmte Befehle für Sensoren hinzugefügt. Einer der großen Vorteile ist die automatische Erkennung und Konfiguration neuer Sensoren.

5.3.2 PositioningML: Beschreibung von Sensoren zur Positionsbestimmung

Für die Interoperabilität zwischen einer Vielzahl von Sensoren, Algorithmen, Positionsbestimmungssystemen, Umgebungsmodellen und Nutzern wird in [63] mit PositioningML eine standardisierte Beschreibungssprache zum Austausch und zur Verarbeitung von Informationen eingeführt. PositioningML ist in XML spezifiziert und erweitert die SWE-Initiative [109] mit Sprachen wie SensorML [19] und OM [27] um die Aufgabe der Positionsbestimmung. Weitere Elemente von MathML [8] oder UncertML [1] sorgen für eine größere Ausdrucksstärke und ermöglichen es sogar, komplexe Algorithmen oder die Unsicherheit in Ortungsverfahren auszudrücken.

Die Sprache wird in der Plattform für mehrere Aufgaben eingesetzt. Neben der Definition einer einheitlichen Kommunikationssprache ist die Beschreibung einzelner Sensoren die Hauptaufgabe der Sprache in der Plattform. Ein Sensor wird dabei als eine Einheit

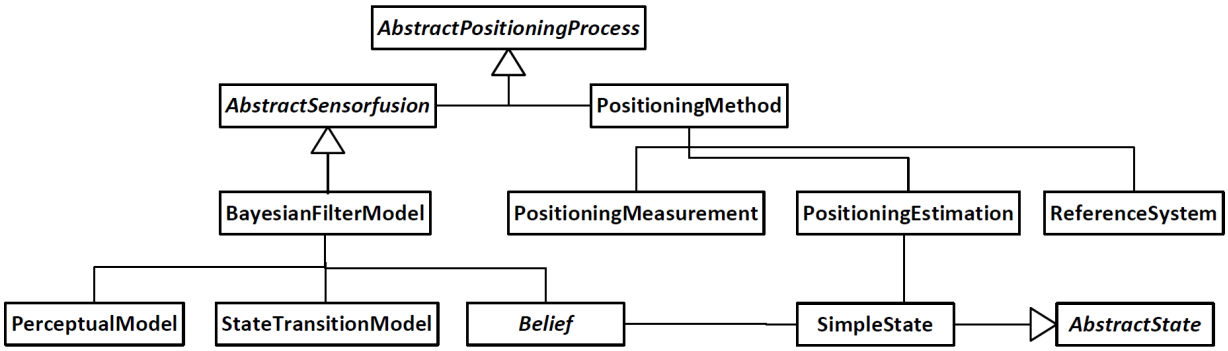


Abbildung 5.10: Assoziationen zwischen den Hauptklassen in PositioningML [63]

definiert, die eine Eigenschaft misst. Dies kann ein künstliches Signal sein, Umgebungsdaten oder relative Veränderungen, aber genauso gut die Position, Geschwindigkeit oder der Ausschluss einer Position. Dementsprechend werden auch Positionierungssysteme, Algorithmen zur Sensor Fusion und Positionsbestimmung sowie Modelle wie Bewegungs- oder Ausbreitungsmodelle mit einbezogen. PositioningML definiert einen Datentyp **AbstractPositioningProcess**, einem Erbe des SensorML-Datentyps AbstractProcess, als Basis für alle Sensoren zur Positionsbestimmung.

Es gibt zwei Arten von Sensoren zur Positionsbestimmung: Entweder steht der Sensor für eine Methode zur Positionsbestimmung, in welchem Fall er durch den Typ **PositioningMethod** beschrieben wird, oder für einen Sensor Fusions Mechanismus, welcher durch den abstrakten Datentyp **AbstractSensorfusion** dargestellt wird. Im Augenblick existiert nur eine Realisierung dieses Typs, nämlich der Typ **BayesianFilterModel**, der die üblichen Sensor Fusion Mechanismen wie Kalman und Partikelfilter umfasst. Andere Methoden können einfach hinzugefügt werden, indem die Beschreibung eines Bayesischen Filters adaptiert wird. Einfache Sensoren, wie Kompass, Beschleunigungssensor oder eine Kamera können bereits mit bestehenden SensorML Typen beschrieben werden, wodurch eine Vielzahl von Sensoren in die Positionsbestimmung mit einbezogen werden kann.

Bayesisches Filtermodell

Jedes Modell eines Bayesischen Filters wird durch eine Filtermethode des Typs **FilterMethod** beschrieben, welche nicht nur eine textuelle Definition oder einen Link zu einem ausführbaren Programm oder Quellcode in beliebiger Programmiersprache aufweisen kann, sondern sogar eine komplette Beschreibung aller Abarbeitungsschritte des Filters in MathML im Typ **ProcessMethodExtended**. Dies erlaubt der Sprache Berechnungsdetails abzubilden, welche die Übertragung von Programmlogik von der Plattform auf Nutzeranwendungen zur Endgeräte-basierten Positionsbestimmung ermöglicht.

Neben einer Beschreibung der Filtermethode wird ein **BayesianFilterModel** durch ein oder mehrere Wahrnehmungs- oder Messmodelle vom Typ **PerceptualModel** beschrieben, einem Zustandsübergangsmodell (**StateTransitionModel**), der Güte der Positionsschätzung (**Belief**) und bestimmten Eigenschaften der Fusionsmethode, welche in

Form eines Typs **FusionQualityCharacteristics** angegeben werden. Des Weiteren werden Eingangs- und Ausgabeparameter für jede Filtermethode definiert. Die Eingabe besteht aus einem Beobachtungsvektor und einem optionalen Kontrollvektor, der bekannte Charakteristika des Filters modelliert. Die Ausgabe besteht aus einem Zustandsvektor, der den Zustand des Systems angibt, nachdem die Eingangsdaten verarbeitet wurden. Dieser vereint das Wissen aus Zustandsübergangsmodell und den Wahrnehmungsmodellen.

Das **PerceptualModel** beschreibt, wie von einer Messung oder Beobachtung ausgehend ein Zustand berechnet wird. Es beinhaltet eine Liste von Sensor-Beschreibungen von den Sensoren, deren Messwerte von dem Modell verarbeitet werden. Dabei können natürlich auch Methoden oder Systeme zur Positionsbestimmung, Sensor Fusion Algorithmen oder grundlegende Sensoren gemeint sein, welche mit SensorML beschrieben sind. Wenn Abweichungen einzelner Sensoren bekannt sind, können diese Informationen als Rauschen modelliert werden, welches die Zustandsschätzung beeinflusst. Das Rauschen kann dabei mithilfe von UncertML oder MathML beschrieben werden, um komplexe Verteilungen oder einfache Funktionen abzubilden. Wahrnehmungsmodelle können einen Eingangsvektor von Messungen verarbeiten, welche als Element des Typs **AbstractState** gegeben sind. Im Augenblick existieren zwei mögliche Realisierungen dieses abstrakten Typs, nämlich der Typ **SimpleState** und der Typ **OMState**. Ersterer besteht aus ein oder mehreren Messungen mit einem Zeitstempel und einem optionalen Wert für die Verlässlichkeit des Messwertes. Alternativ zum **SimpleState** kann ein **OMState** in der standard OM-Syntax angegeben werden, um eine direkte Integration in bestehende Tools zu ermöglichen. Da es jedoch im Augenblick noch nicht möglich ist, in OM-Syntax Unsicherheiten auszudrücken, wird erstere Methode bevorzugt. Die Funktionsweise jedes Wahrnehmungsmodells kann zudem detailliert durch ein **ProcessMethodExtended**-Element angegeben werden, welches - wie bereits bei der **FilterMethod** angesprochen - eine komplette Beschreibung der Berechnungsschritte ermöglicht. Des Weiteren können Karteninformationen, wie die Lage von Wänden oder begehbare Flächen, angegeben werden.

Das **StateTransitionModel** beschreibt den Zustandsübergang über die Zeit. Es kann auch wiederum durch ein **ProcessMethodExtended**-Element im Detail angegeben werden und wird genutzt, um eine kontinuierliche Funktion des Zustandsvektors zwischen verschiedenen Messwerten anzugeben, welche von Wahrnehmungsmodellen bearbeitet werden können. Typischerweise werden Zustandsübergangsmodelle durch Bewegungsmodelle oder Vorhersagemodelle realisiert.

Das **Belief**-Element drückt die Unsicherheit eines Zustands basierend auf den Ungenauigkeiten der einzelnen Modelle aus. In Abhängigkeit des Bayesischen Filtermodells kann dies durch Mittelwerte und Varianzen für Gauß-verteilte Zustände geschehen, wie im Beispiel eines Kalman Filters, oder durch die Diskretisierung beliebiger Verteilungen, wie im Fall eines Partikelfilters. Der **Belief** wird durch ein UncertML-Element ausgedrückt, wodurch verschiedene Verteilungen ausgedrückt werden. Die Angabe der Unsicherheit einzelner Verfahren ermöglicht eine Vergleichbarkeit zwischen verschiedenen Filter-Methoden bezüglich der Genauigkeit und Präzision.

Die speziellen Eigenschaften eines Filters werden schließlich durch den Typ **FusionQualityCharacteristics** modelliert. Diese sind wichtig um Sensoren auf Sensor Fusions

Algorithmen abzubilden, was beispielsweise Informationen verlangt, welche Arten von Daten für die Fusion notwendig sind. Ein Kalman Filter beispielsweise arbeitet nur mit Gaußverteilten Daten optimal. Dementsprechend bieten diese Informationen wertvollen Input für die Entscheidungsfindung der Plattform, welche Sensoren und Algorithmen miteinander kombiniert werden. So kann beispielsweise auch die Dimension eines Zustands- oder Eingangsvektors beschränkt werden oder die Fähigkeit, multi-modale Verteilungen abbilden zu können, angegeben werden.

Modellierung von Methoden zur Positionsbestimmung

Eine **PositioningMethod** enthält eine Liste von Sensoren, beschrieben durch den SensorML Typ AbstractProcess, die als Eingangsgröße benötigt werden. Diese können beliebige andere Methoden zur Positionsbestimmung, Ortungssysteme, Sensor Fusions Mechanismen oder grundlegende Sensoren sein, die in SensorML oder PositioningML beschrieben werden. Anstelle von oder zusätzlich zu den Sensoren, können auch verschiedene Messungstypen vom Typ **PositioningMeasurements** angegeben werden. Eine bestimmte Position wird durch den Typ **PositioningEstimation** angegeben. Beide Arten von Elementen, sowohl **PositioningMeasurements** als auch **PositioningEstimations**, können in beliebigen Detailstufen beschrieben werden, die von einer textuellen Beschreibung bis zur vollständigen Berechnungsvorschrift in Form eines **ProcessMethodExtended**-Elements reichen. Jede Position referenziert das zugehörige Referenzsystem vom Typ **ReferenceSystem**. Zudem kann eine Methode zur Positionsbestimmung auf eine beliebige Anzahl an Sensor Fusions Mechanismen in Form von **AbstractSensorfusion**-Elementen verweisen, für welche diese Methode eingesetzt werden kann. Wie bei Fusions Modellen können charakteristische Eigenschaften der Methode in einem **PositioningQualityCharacteristics**-Element definiert werden, um beispielsweise Bewertungen von Genauigkeit, Präzision, Latenzen und anderen Eigenschaften zu ermöglichen. Auf dieser Basis können Methoden zur Positionsbestimmung je nach gewünschten Eigenschaften für jeden Anwendungsfall einzeln bewertet werden. Schlussendlich verfügt jedes **PositioningMethod**-Element über einen Eingangsvektor von Messdaten und einen Ausgangsvektor des Zustandes. Im Nachfolgenden werden die Sprachelemente **PositioningMeasurement**, **PositioningEstimation** und **ReferenceSystem** genauer beschrieben.

Ein **PositioningMeasurement**-Element beschreibt physikalische Beobachtungen, Anforderungen, Transformationen und Eigenschaften einer Messung. Die Beobachtungen werden dabei grundsätzlich auf dieselbe Weise charakterisiert, wie die Eingangsvektoren von Messdaten. Zusätzlich wird allerdings noch die beobachtete Eigenschaft der Messgröße angegeben, zum Beispiel ob es sich um eine Entfernungsmessung oder relative Positionsänderung handelt. Zu diesem Zweck können in einem **PositioningMeasurement**-Element ein oder mehrere Referenzpunkte angegeben werden, um die Entfernungsschätzung in Relation zur Position eines Senders zu setzen oder Trainingsdaten in einem Fingerprinting-System anzugeben. Des Weiteren kann auf bestimmte Bedingungen verwiesen werden, welche abhängig von einer Zeit, einem Wert oder einem Token sein können. Die Bedingung beschränkt die Gültigkeit der Messungen für die Positionsbestimmung. Wert-abhängige

Bedingungen können beispielsweise die gemessene Signalstärke für Distanzschätzungen auf bestimmte Wertebereiche beschränken. Token-abhängige Beschränkungen können genutzt werden, um beispielsweise textuelle Werte durch reguläre Ausdrücke einzuschränken.

Ein **PositioningEstimation**-Element beschreibt die Schlussfolgerung einer Position von einem **PositioningMeasurement**-Element. Positionsschätzungen können dabei die Beschreibung von Algorithmen, Umgebungsmodellen und Referenzpunkten beinhalten. Solch ein Element gibt einen Zustandsvektor zurück, der auf Basis eines oder mehrerer **PositioningMeasurement**-Elemente berechnet wurde und die Einträge enthält, welche von der bestimmten Methode ermittelt wurden. Solch ein **PositioningEstimation**-Element wird durch eine oder mehrere von fünf Grundtechniken zur Positionsbestimmung beschrieben. Diese sind Angulation, Lateration, Fingerprinting, Koppelnavigation und Nahbereichserkennung und können dazu eingesetzt werden, um Methoden der Positionsbestimmung zu klassifizieren und benötigte Eingangssensoren oder zumindest deren Messarten festzustellen.

Jede Position wird bezüglich eines bestimmten **ReferenceSystems** angegeben, welches in einem beliebigen Format vorliegen kann. Dieses kann symbolisch, geometrisch, global, lokal, absolut oder relativ sein und legt den Kontext fest, in welchem eine Position interpretiert wird. Um den Bezug zu anderen Referenzsystemen herzustellen, sollte jedoch eine Transformationsvorschrift gegeben werden, welche die Übersetzung in ein bekanntes und standardisiertes Referenzsystem wie WGS84 ermöglicht.

5.3.3 Plattform für hybride Positionsbestimmung

In diesem Abschnitt wird die Plattform vorgestellt. Dabei wird zuerst auf die Anforderungen eingegangen, gefolgt von einem Überblick über die verschiedenen Komponenten, welche anschließend im Detail vorgestellt werden.

Eine Plattform zur hybriden Positionsbestimmung muss mehrere Anforderungen erfüllen, welche von den verschiedenen Anwendungsszenarien und den Anforderungen bestimmter Ortungsverfahren kommen. Weitere Anforderungen entstehen durch die notwendige Interoperabilität der Plattform mit existierenden Systemen. Die folgenden Anforderungen wurden in [63] identifiziert:

- Eine Beschreibungsmöglichkeit für heterogene Sensoren ist notwendig, um spezielle Eigenschaften der verschiedenen Sensoren in die Positionsbestimmung einzubeziehen. Zudem kann so das erwartete Eingangs- und Ausgabeformat einzelner Sensoren spezifiziert werden. Hierfür und für spezielle Nutzeranforderungen und Kommunikationskanäle ist eine einheitliche Beschreibung notwendig. Diese Anforderungen werden durch die eben eingeführte Beschreibungssprache PositioningML aus Abschnitt 5.3.2 bereits erfüllt.

- Es muss möglich sein, Nutzeranforderungen und die Eigenschaften verfügbarer Sensoren in Einklang zu bringen und als Schlussfolgerung eine zu jeder Situation passende Positionsbestimmung zu ermöglichen.

- Die Plattform sollte zudem die Fähigkeit haben, mit wechselnden Umgebungen, wechselnden Sensoren und verschiedenen Nutzerkriterien umgehen zu können, auch wenn die Wechsel während der Positionsbestimmung auftreten.

- Es sollte zudem möglich sein, mehrere Sensoren dynamisch zur Laufzeit zu kombinieren, Sensor Fusionsmechanismen anzuwenden oder ungenaue Sensoren durch Wissen von genaueren Sensoren zu kalibrieren.

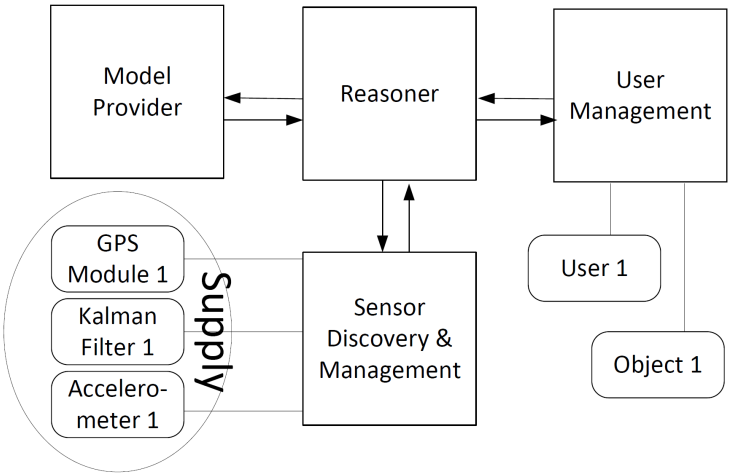


Abbildung 5.11: Ein Überblick über die Komponenten der Plattform mit Beispielsensoren und Nutzern.

Die Plattform besteht aus vier Komponenten (vergleiche Abbildung 5.11). Die erste ist für die Kommunikation mit und Verwaltung von allen Sensoren sowie das Entdecken neuer Sensoren zuständig. Die Menge aller im Augenblick bekannten und verfügbaren Sensoren wird als *Angebot* bezeichnet. Verfügbarkeit heißt in diesem Fall, dass ein Sensor zeitlich und örtlich verfügbar ist, z.B. dass ein Ortungssystem in einem Supermarkt an ist und sich der Nutzer im Bereich der Positionsbestimmung befindet. Ein Mechanismus zum Auffinden von Sensoren überwacht kontinuierlich die Verfügbarkeit bekannter und die Entdeckung neuer Sensoren. Jeder neu entdeckte Sensor stellt der Plattform seine Beschreibung in PositioningML zur Verfügung, welche ausgewertet wird und anschließend eine Kommunikationsschnittstelle zu dem Sensor etabliert.

Das Herzstück der Plattform ist jedoch der Reasoner. Dieser wertet eintreffende Messwerte aus und trifft die Entscheidung über deren Weiterverarbeitung. Dazu werden Messwerte mit den Beschreibungen der verfügbaren Sensoren verglichen und so die Fähigkeit der Verarbeitung des Messwerts in Erfahrung gebracht. Letztlich wird der Sensor zur Weiterverarbeitung ausgewählt, welcher die Daten verwerten kann und den Nutzerkriterien am nächsten kommt. Zudem kann der Reasoner auf eine Historie von Sensordaten zugreifen und so beispielsweise die letzte bekannte Position als Startpunkt für ein Dead Reckoning System einfließen lassen.

Schließlich gibt es noch die Komponenten Model Provider und User Management. Erstere verwaltet Karten und Trainingsdaten in verschiedenen Darstellungsformen und stellt diese dem Reasoner zur Verfügung. So können beispielsweise Messdaten unterschiedlicher Referenzsysteme kombiniert werden, solange dem Model Provider eine Transformation zwischen den Systemen möglich ist. Die Komponente zum User Management verwaltet die Positionsdaten der verschiedenen Nutzer sowie deren jeweilige Kriterien zur Positionsbestimmung. Zudem wird über diese Schnittstelle die Häufigkeit und Art der Positionsupdates an den Nutzer festgelegt.

Sensor Management und Discovery

Die Plattform benötigt eine Schnittstelle zur Anbindung von beliebigen externen Sensoren. An diesem können sich Sensoren registrieren und Messwerte mit der Plattform austauschen. Dies geschieht in der Komponente für Sensor Management und Discovery. Hier werden Sensoren auf ihre Verfügbarkeit hin untersucht und neue Sensoren entdeckt. Dabei hält die Komponente eine Beschreibung jedes bekannten Sensor vor, welche in Form einer Beschreibung in PositioningML vom Sensor an die Komponente weitergegeben wird. In dieser stehen die entsprechenden Fähigkeiten, benötigten Eingangsdaten und die gemessenen Ausgabegrößen. Neue Messwerte bestehender und bekannter Sensoren sowie neue Messwerte verursachen Ereignisse in der Komponente, welche dem Reasoner mitgeteilt werden, um eine weitere Bearbeitung zu ermöglichen. Zudem dient die Komponente als Sensor-Abstraktionsschicht und kümmert sich um die Kommunikation zwischen Plattform und externen Sensoren. Die Syntax von Nachrichten zwischen Sensoren und Plattform ist durch PositioningML gegeben. So ist es für Anbieter von bestehenden externen Sensoren einfach, eine Anbindung an die Plattform zu ermöglichen, indem eine Kommunikationseinheit für PositioningML Nachrichten zwischengeschaltet wird.

Das Entdecken neuer Sensoren und das Hinzunehmen zur Plattform kann über verschiedene Mechanismen erfolgen. Bekannte Ansätze aus der Diensterkennung und Vermittlung, wie Nachschlagdienste oder Agenten-basierte Systeme sind denkbar und lassen sich in die Plattform integrieren. Zudem werden bekannte Sensoren kontinuierlich auf ihre Verfügbarkeit überprüft und nicht mehr verfügbare Sensoren als solche gekennzeichnet. Alle im Augenblick zur Verfügung stehenden Sensoren werden in einer Datenbank gehalten, welche für die Verwaltung dynamischer Inhalte geeignet ist, da sich das Angebot im Verlauf der Positionsbestimmung ändern kann. Dabei können sich nicht nur die Anzahl und Art der verfügbaren Sensoren ändern, sondern auch die Eigenschaften der Sensoren selbst, wie Arbeitslast, Energieverbrauch oder Auslastung, unterliegen einer eigenen Dynamik.

User Management

Die Komponente User Management verwaltet die Nutzer der Plattform und deren Kriterien bezüglich der Ortungsverfahren. Dabei kann ein Nutzer verschiedene Kriterien, wie Energieverbrauch, Genauigkeit der Positionsbestimmung oder den Bedarf an Systemen ohne Trainingsphase oder Nutzerfeedback einstellen, wodurch der Einsatz bestimmter Sensoren

zur Positionsbestimmung beschränkt wird. Dies geschieht implizit durch die Bewertung der einzelnen Sensoren durch den Reasoner, welcher die bezüglich der Nutzerkriterien passendsten Sensoren miteinander kombiniert. Ein Beispiel bietet ein modernes Smartphone, welches eine Vielzahl von Sensoren zur Positionsbestimmung bietet. Soll bei der Positionsbestimmung auf geringen Energieverbrauch bei hoher Genauigkeit Wert gelegt werden, so sollte GPS nach einer initialen Messung durch Dead Reckoning mit inertialen Sensoren und Mapmatching ersetzt werden, solange die Genauigkeit genügend hoch ist. Sinkt die Genauigkeit unter einen geforderten Grenzwert, so wird GPS kurzzeitig aktiviert, um das Dead Reckoning System neu zu kalibrieren. Im Fall von widersprüchlichen Kriterien kann die Plattform ein Pareto-Optimum bezüglich der Kriterien finden oder der Nutzer die Widersprüche durch Prioritäten beseitigen. So könnte zum Beispiel die energieeffizienteste Methode unter allen Methoden einer gewissen Genauigkeit den Vorzug erhalten. Es gilt zu beachten, dass ein Nutzer im System auch durch ein zu ortendes Gerät gegeben sein kann, wie im Fall von Logistik und/oder Lagerhaltung, wo viele Objekte nachverfolgt werden müssen. Zudem ist es auch möglich, über die Plattform Informationen über andere Nutzer zu erhalten. Dies ist jedoch durch Zugriffkontrollregeln und Nutzereinstellungen zur Weitergabe von Positionsinformationen einzuschränken, so dass die Privatsphäre des Nutzers zu jeder Zeit sichergestellt werden kann. Die Betrachtungen des daraus folgenden Rechtemanagements werden in dieser Arbeit nicht behandelt.

Model Provider

Der Model Provider speichert und verwaltet Umgebungsinformationen wie Gebäudepläne, Kartenmaterial und Trainingsdaten, wie beispielsweise von Fingerprinting Algorithmen. Die Komponente bietet die Funktionalität, Transformationen zwischen verschiedenen Referenzsystemen und globalen Koordinatensystemen anzubieten, wie beispielsweise WGS84. Dazu müssen zu jeder Karte bestimmte Informationen wie die Verschiebung des Ursprungs, die Rotation der einzelnen Achsen zueinander und der Maßstab angegeben werden. Zudem bekommt jedes Modell und jede Karte einen eigenen eindeutigen Identifikator, welcher auch von Ortungssystemen genutzt wird, um Positionsschätzungen bezüglich eines bestimmten Referenzsystems anzugeben. Zudem kann der Modellanbieter dem Reasoner oder der Komponente für Sensormanagement Umgebungsinformationen übermitteln, damit bestimmte Algorithmen direkt mit den benötigten Modellen arbeiten können. Zu diesem Datenaustausch werden die Informationen in einem standardisierten Format bereitgestellt, welches den in Kapitel 3 beschriebenen Repräsentationen entspricht und somit entweder in Form einer annotierten Bitmap oder eines BIGML Dokuments [61] gegeben ist.

Reasoning und Kombination

Die Hauptkomponente der Plattform ist der Reasoner. Er hat die Aufgabe, neue Messwerte zu verarbeiten, indem diese mit den Nutzerkriterien und den verfügbaren Sensoren abgeglichen werden, um zu entscheiden, wie die Messwerte weiter verarbeitet werden sollen. Sollte die Messung bereits die Kriterien erfüllen, beispielsweise in Form einer Positionsschätzung,

so kann sie direkt an den Nutzer über die User Management Komponente weitergegeben werden. Zudem kann die Messung gespeichert werden und/oder zur weiteren Verarbeitung an passende Sensoren weitergegeben werden. Dazu werden die Fähigkeiten der Sensoren anhand ihrer Beschreibung ermittelt und mit den Messwerten verglichen. Wenn ein Messwert an einen Sensor übertragen wird, dessen Beschreibung die Fähigkeit zur Verarbeitung ergibt, so ist die Antwort dieses Sensors wiederum ein neuer Messwert. Der Vorgang zur Überprüfung der Fähigkeiten einzelner Sensoren kann zur besseren Automatisierung vorberechnet werden, was weiter unten beschrieben wird. Die Kommunikation mit externen Sensoren läuft über Nachrichten in PositioningML Syntax. Falls ein Sensor zur Bearbeitung weitere Daten, wie andere Messwerte oder Umgebungsinformationen, benötigt, so werden diese Daten vom Reasoner erhoben und an den entsprechenden Sensor weitergeleitet.

Da das Ereignis einer neuen Messung im Vergleich zum Wechsel von Sensoren sehr häufig auftritt, bietet es sich an, den Workflow der Abarbeitung von Messwerten für einzelne Nutzer vorzuberechnen. Um aus einer festen Menge von Sensoren eine auf die Nutzerkriterien passende Positionsschätzung zu ermitteln, sollten die folgenden Schritte ausgeführt werden, um den Prozess des Reasonings zu beschleunigen:

1. Für jeden Sensor, der eine Methode zur Positionsbestimmung darstellt, werden die Sensoren gefunden, welche die notwendigen Eingangsdaten liefern. Dies geschieht durch den Vergleich der Ausgangsvektoren von Sensoren mit dem geforderten Eingangsdatenvektor der Sensors in Betrachtung. Stimmen beide zumindest teilweise überein, wird eine Verknüpfung erstellt. Falls der Eingang nur teilweise mit den gebotenen Ausgangsdaten übereinstimmt, muss so lange weitergesucht werden, bis schließlich Sensoren für alle benötigten Eingangsdaten ermittelt wurden. Falls mehrere Sensoren gefunden werden, die als Anbieter von Eingangsdaten infrage kommen, so werden diejenigen ausgewählt, welche am besten auf die Nutzerkriterien passen. Zu diesem Zeitpunkt werden ebenfalls die in Abschnitt 5.3.2 genannten Beobachtungen und Grundtechniken zur Positionsbestimmung verglichen. So kann beispielsweise ein Sensor, der Entfernungsschätzungen liefert, für Lateration hergenommen werden. Zudem kann eine Methode zur Positionsbestimmung in diesem Schritt mit Informationen aus Umgebungsmodellen versorgt werden.

2. Für jeden Sensor, der eine Methode zur Positionsbestimmung darstellt, werden verfügbare Sensor Fusions Mechanismen gefunden und verknüpft, die für diese Art von Methode einsetzbar sind. Falls die Mechanismen in einer bestimmten Reihenfolge verarbeitet werden müssen, so werden sie gleich in dieser Reihenfolge verknüpft. Eine Besonderheit des Reasoners ist jedoch, dass auch nicht explizit angegebene Sensor Fusions Mechanismen gefunden werden. Dies wird im nächsten Schritt erreicht.

3. Für jeden Sensor, der einen Sensor Fusionsalgorithmus darstellt, werden alle Methoden zur Positionsbestimmung gefunden und verknüpft, deren Messwerte von diesem Algorithmus verarbeitet werden können. Zu diesem Zweck werden das Zustandsübergangsmodell und der Kontrolleingang des Sensor Fusionsalgorithmus mit den Ein-

und Ausgangsdaten der Sensorkandidaten überprüft. Eingangsdaten liefern Fusionsmöglichkeiten auf einer niedrigen Ebene, beispielsweise der Kombination von Rohdaten, während Ausgangsdaten genutzt werden können, um Positionsschätzungen verschiedener Sensoren auf einer höheren Ebene zu vereinen. Die Entscheidung basiert dabei zusätzlich auf den Qualitätskriterien des Fusionsalgorithmus.

4. Für jeden Sensor, der einen Sensor Fusionsalgorithmus darstellt, werden alle weiteren benötigten Sensoren gefunden, die nicht notwendigerweise mit den bereits verknüpften Sensoren übereinstimmen. Dazu wird der Eingangsvektor des Fusionsalgorithmus mit den Ausgangsdaten der Sensoren verglichen. Zudem werden in diesem Schritt benötigte Informationen aus Umgebungsmodellen den Algorithmen zur Sensorfusion bereitgestellt.

5. Schlussendlich wird jede mögliche Kette von Verknüpfungen ausgewertet und mit den Nutzerkriterien verglichen. Als Ergebnis wird die am besten auf die Kriterien passende Kette zur Positionsbestimmung ausgeführt, was zu der bestmöglichen Kombination von Sensoren aus dem Angebot zur Positionsbestimmung führt. Falls genügend Kapazität verfügbar ist und dabei keine Kriterien, wie niedriger Energieverbrauch, verletzt werden, können auch mehrere Ketten ausgeführt werden und das jeweils beste Ergebnis an den Nutzer weitergegeben werden. In diesem Fall kann neben den Charakteristiken der Sensoren auch die Qualität einzelner Messungen in die Berechnung der Qualität der Positionsschätzung mit aufgenommen werden. Neben der Rückgabe der Positionsschätzung an den Nutzer wird diese ebenfalls in der Reasoner-Komponente gespeichert, um beispielsweise anderen Positionsbestimmungsmethoden als Eingang oder Korrekturgröße zu dienen. So kann eine absolute Position beispielsweise als Startpunkt für Dead Reckoning Systeme dienen oder anhand von mehreren Positionen eine durchschnittliche Schrittlänge ermittelt werden.

Der beschriebene Prozess sollte jedes Mal ausgeführt werden, wenn ein neuer Sensor verfügbar wird, ein bestehender Sensor verschwindet oder sich die Nutzerkriterien ändern. Nachdem diese Schritte durchgeführt wurden, können eintreffende Messungen direkt den etablierten Ketten von Verknüpfungen folgen, ohne weiteres Reasoning betreiben zu müssen. Ein Beispiel für eine solche Auswertung unter Einbeziehung von Genauigkeit und Verfügbarkeit von Sensoren wird in der prototypischen Evaluation gegeben.

Theoretische Evaluation

Die dynamische Kombination verschiedener Sensoren, Positionsbestimmungssysteme und Algorithmen zur Positionsbestimmung oder Sensor Fusion wird in [63] dadurch erreicht, dass alle Sensoren mitsamt deren Eigenschaften in einem standardisierten Format beschrieben werden. So können zur Laufzeit Kombinationsmöglichkeiten ermittelt und bewertet werden. Damit ist es möglich zu jedem Zeitpunkt aus den aktuell zur Verfügung stehenden Sensoren und Messwerten die bestmögliche Positionsschätzung in Bezug auf verschiedene Nutzerkriterien zu erlangen.

Die vorgestellte Plattform hat zudem einige Vorteile gegenüber vergleichbaren Ansätzen aus verwandten Arbeiten. Die Hauptstärke liegt in der großen Flexibilität, die es erlaubt, verschiedene Sensoren in die Positionsbestimmung mit einzubeziehen und auch zur Laufzeit Änderungen zuzulassen. Zudem bietet die Bereitstellung der Plattform selbst eine gewisse Flexibilität, da diese sowohl auf mobilen Geräten wie Smartphones, als auch in einer Server-Infrastruktur betrieben werden kann. Des Weiteren kann die Plattform für eine lokale Nutzung in einzelnen Gebäuden verwaltet werden oder als skalierbare Lösung für den globalen Einsatz.

Die Plattform erlaubt die Bereitstellung von Modellen, Methoden und Algorithmen über die eigenen Grenzen hinweg. Dabei wird durch eine sehr detailierte Beschreibung erreicht, dass eine automatische Programmcode-Generierung eingesetzt werden kann, um aus der Beschreibung ausführbaren Code zu erzeugen. Das kann beispielsweise genutzt werden, um die Privatsphäre von Nutzern zu schützen, indem alle benötigten Algorithmen und Modelle direkt auf das mobile Gerät geladen werden und keine Kommunikation mit der Plattform oder anderer Infrastruktur notwendig ist. Auf der anderen Seite können komplexe Algorithmen in eine leistungsstarke Infrastruktur ausgelagert werden, um auf dem mobilen Gerät Strom zu sparen.

5.3.4 Anwendungsszenario und Evaluation

In diesem Abschnitt wird die Evaluation der prototypischen Implementierung der Plattform in einem Gebäude der LMU München vorgestellt. Die Evaluation zeigt die Fähigkeit der Plattform, verschiedene heterogene Sensoren und deren Daten zu verwalten. Zudem wird gezeigt, dass die Plattform mit einem wechselnden Angebot von verfügbaren Sensoren umgehen kann, und einzig auf Basis der verfügbaren Sensoren und deren Genauigkeiten ein Szenario der nahtlosen Positionsbestimmung in Außen- wie Innenbereichen ermöglicht.

Szenario

Um ein paar Möglichkeiten der Plattform zu präsentieren und die Machbarkeit des Ansatzes zu zeigen, wurde die Plattform in einem Szenario der nahtlosen Positionsbestimmung eingesetzt. Es wird davon ausgegangen, dass ein Student die Plattform und sein Smartphone zur Positionsbestimmung und Navigation zu einem Hörsaal in der Universität einsetzt. Dementsprechend sind die Nutzerkriterien des Studenten, dass die Positionsbestimmung eine genügend hohe Genauigkeit für eine Navigationsanwendung aufweisen soll, was im Bereich von Straßennetzen ca. 10 Meter, in Innenbereichen unter 2 Metern entspricht. Zudem darf die Positionsberechnung maximal drei Sekunden dauern, um Echtzeitnavigation zu ermöglichen.

Die Plattform erkennt die Nutzerkriterien und das mobile Gerät mit GPS, WLAN, Beschleunigungssensor und Kompass, da diese Sensoren in PositioningML beschrieben sind. Die Plattform selbst verfügt über einen Algorithmus zur Schritterkennung. Anschließend wartet die Plattform auf Messwerte. Da keine WLAN-Positionsbestimmungsmethode verfügbar ist und Dead Reckoning mit Beschleunigungssensor und Kompass eine initiale

Abbildung 5.12: Zwei gelaufene Pfade (schwarz) mitsamt den geschätzten Pfaden (links in rot und rechts in blau).

Position benötigt, wartet die Plattform auf eine absolute Positionsschätzung, die vom GPS-Sensor zur Verfügung gestellt wird. Messungen der anderen Sensoren können noch nicht sinnvoll zu einer Positionsschätzung verarbeitet werden. Nach der ersten GPS-Messung wird die Positionsschätzung den Kriterien entsprechend akzeptiert und die Navigation kann beginnen. Da die Plattform über einen Sensor zur Schritterkennung verfügt, werden die Daten des Beschleunigungssensors von der Plattform verarbeitet, um eine mittlere Schrittlänge für den Nutzer für die spätere Verarbeitung zu schätzen.

Nach einiger Zeit betritt der Student das Universitätsgebäude, wo die Plattform eine bestehende WLAN-Infrastruktur erkennt, zu der im Model Provider Trainingsdaten für einen Fingerprinting-Algorithmus vorliegen. Zudem wird von der Universität der kNN-Algorithmus aus Abschnitt 4.3.2 und ein Partikelfilter aus Abschnitt 4.2.2 basierend auf Kompass und - optional - Gyroskop, Beschleunigungssensor und Kartenmaterial bereit gestellt, welche von der Plattform entdeckt und aufgrund der Beschreibung in PositioningML zu den dem Studenten zur Verfügung stehenden Sensoren hinzugefügt werden. Sobald die Genauigkeit der WLAN-Positionsbestimmung höher ist als die von GPS, spätestens jedoch nachdem nach Betreten des Gebäudes drei Sekunden lang kein GPS Signal mehr gemessen wird, werden die WLAN-Signale von der Plattform an den kNN-Algorithmus weitergegeben, um eine absolute Position im Gebäude zu schätzen. Diese wird wiederum nicht nur dem Nutzer mitgeteilt, sondern ebenfalls verwendet, um den Partikelfilter zu initialisieren.

Während sich der Student durch das Gebäude bewegt, wechselt sowohl die Zahl der sichtbaren WLAN-Zugangspunkte wie auch deren empfangene Signalstärke. Um die Stabilität der Positionsbestimmung zu erhöhen und die Genauigkeitsanforderungen des Nutzers

zu gewährleisten, wird neben der Positionsbestimmung per WLAN auch eine kontinuierliche Positionsbestimmung mit Hilfe des Partikelfilters durchgeführt. Dazu werden die Daten des Beschleunigungssensors und des Kompass vom mobilen Gerät an die Plattform geschickt, wo sie vom Reasoner an die Schnittstelle des Partikelfilters weitergereicht werden. Dort werden die Daten verarbeitet und zu einer Positionsschätzung kombiniert. Zudem ist die Plattform in der Lage, dem Partikelfilter die mittlere Schrittlänge des Studenten mitzuteilen, die mithilfe der Schritterkennung und vorherigen Positionsschätzungen ermittelt werden konnte, um so die Genauigkeit des Partikelfilters erheblich zu erhöhen. Die Ergebnisse des Partikelfilters werden wiederum als neue Messwerte der Plattform zur Verfügung gestellt und dem Nutzer mitgeteilt, solange die erreichte Genauigkeit den Kriterien entspricht und genauer ist als die zeitgleich berechnete WLAN-Position. Ist hingegen die WLAN-Position genauer, so wird diese nicht nur dem Nutzer mitgeteilt, sondern der Partikelfilter mit der neuen genaueren Position neu initialisiert.

Fähigkeiten der Plattform

Das Szenario betont einige der besonderen Eigenschaften der Plattform. Es zeigt nicht nur die Fähigkeit, mit heterogenen Sensordaten umzugehen, sondern ebenfalls die verfügbaren Methoden zur Positionsbestimmung und Algorithmen zu Sensorfusion mit den benötigten Eingangsdaten zu versorgen. Zudem können Positionsschätzungen von verschiedenen Sensoren ausgewertet und in Bezug auf die Nutzerkriterien und die Vertrauenswürdigkeit bewertet werden, wodurch die Plattform entscheiden kann, wie mit den Daten weiter vorzugehen ist.

Im Szenario wird die berechnete Genauigkeit der Positionsschätzung und die Verfügbarkeit von Sensoren dazu genutzt, um zu entscheiden, welche Positionsschätzung mehrerer Sensoren an den Nutzer weitergegeben werden soll. Solange GPS verfügbar ist und eine genügend hohe Genauigkeit aufweist, wird es zur Positionsbestimmung genutzt. Zudem wird gleichzeitig eine Schritterkennung genutzt, um die Schrittlänge des Nutzers aus der Zahl der erkannten Schritte und der zurückgelegten Distanz zu ermitteln. Sobald GPS nicht mehr verfügbar ist, die geforderten Genauigkeiten verletzt oder eine genauere Methode zur Positionsbestimmung verfügbar ist, wird eine alternative Methode zur Positionsbestimmung genutzt. In diesem Fall ist als zusätzliches Kriterium die Zeit seit dem letzten Positionsupdate gegeben, um einen Echtzeitdienst zu unterstützen. Zudem wird die Fähigkeit der Plattform klar, benötigte Informationen zwischen verschiedenen Komponenten auszutauschen. Dabei wird im Beispielszenario die WLAN-Positionsbestimmung mit einer dem Model Provider bekannten Fingerprint-Datenbank versorgt, der Partikelfilter mit einer Karte der Umgebung, einer initialen Position und mit der mittleren Schrittlänge des Nutzers. In diesem Fall werden WLAN-Positionsbestimmung und der Partikelfilter gleichzeitig ausgeführt, um anhand der Genauigkeit der einzelnen Messungen das jeweils bessere Ergebnis nutzen zu können. Alternativ hätte auch der Partikelfilter aufgrund der durchschnittlich höheren Genauigkeit ausgeführt werden können, falls aufgrund von Energieeffizienz nur eine Methode ausgeführt werden sollte, oder gar nur WLAN-Positionsbestimmung, falls ein geringer Energieverbrauch wichtiger als die Genauigkeit der Positionsbestimmung gewesen

wäre. Die Genauigkeit der jeweiligen Methode wird hierbei zunächst in der Sensorbeschreibung angegeben, jedoch zusätzlich sofern möglich für jeden Messwert einzeln ermittelt. Dabei kann bei WLAN analog zu Abschnitt 4.3.5 die Varianz der nächsten Nachbarn als Genauigkeitsmaß der Messung angesehen werden, während beim Partikelfilter die Varianz der Partikel ein Maß für die Verlässlichkeit der Positionsschätzung liefert.

Die Plattform stellt direkt die benötigten Umgebungsinformationen bereit, wobei diese von der Plattform anhand der aktuellen Position oder der Messwerte des Nutzers zur Laufzeit ermittelt werden. Im Fall der WLAN-Fingerprint-Datenbank konnte diese aus verschiedenen Messungen als relevant ermittelt werden. Zum ersten kann die zuletzt bekannte Position (falls verfügbar und falls zeitlich aktuell) in ein globales Koordinatensystem übertragen werden und die Distanz zu lokalen Gebäudemodellen und Plänen ermittelt werden. Falls ein Modell örtlich nahegelegen ist und/oder mindestens drei Zugangspunkte der aktuellen WLAN-Messung mit einer Messung aus der Trainingsdatenbank übereinstimmen, befindet sich der Nutzer mit großer Wahrscheinlichkeit in der Nähe zu diesem Modell. Sollten sich mehrere Kandidaten in der Nähe befinden, kann die Auswahl durch die Ähnlichkeit der Messungen oder die Varianz der Positionsschätzung bezüglich aller Modelle als Maß für die Wahrscheinlichkeit des Aufenthalts im Gebiet eines bestimmten Modells genommen werden.

Zu guter Letzt kann die Plattform Positionsinformationen relativ zu jedem bekannten Modell berechnen und die Koordinaten zwischen verschiedenen Referenzsystemen und Repräsentation hin und her konvertieren. Zudem erlaubt sie es, die Positionsschätzungen in der jeweils vom Nutzer oder einem Dienst benötigten Form anzubieten. So können beispielsweise bestehende ortsbezogene Anwendungen für den Außenbereich auch mit genauen Positionsdaten in Gebäuden genutzt werden, indem diese in ein globales Koordinatensystem überführt werden.

Prototypische Evaluation

Die Fähigkeiten der Plattform wurden anhand einer prototypischen Implementierung getestet und anhand zweier Pfade bewertet. Beide starten analog zum Szenario in einiger Entfernung zum Gebäude und enden innerhalb des Gebäudes. Die Wechsel zwischen den verschiedenen Sensoren und die Weiterleitung von Messwerten an weitere Sensoren zur Verarbeitung funktionierte zuverlässig. Es wurde allerdings festgestellt, dass die Verfügbarkeit von GPS jeweils schon kurz vor dem Betreten des Gebäudes nachließ. Gerade im Fall des blauen Pfades auf der rechten Seite von Abbildung 5.12 wurde die GPS Positionsbestimmung bereits vor dem Gebäude immer ungenauer und versagte ca. 15 Meter vor dem Eingang. Dennoch war ein Übergang möglich, ohne dass Übergänge zwischen Innen- und Außenbereichen explizit modelliert wurden. Der große Abstand zwischen echtem und geschätztem Pfad an dieser Stelle resultiert aus der Tatsache, dass die WLAN-Trainingsdatenbank nur Daten im Inneren des Gebäudes beinhaltet, wodurch die Position etwas zu früh in das Gebäude versetzt wurde. Bereits nach kurzer Zeit hatte sich die Positionsbestimmung soweit stabilisiert, dass genügend genaue Daten für die Indoor-Navigation zur Verfügung standen.

Allerdings zeigt das Szenario auch eine Schwäche der allgemeinen Modellierung der Plattform bei Übergängen zwischen den von verschiedenen Sensoren abgedeckten Bereichen. Viele Sensoren, wie beispielsweise WLAN, neigen dazu, in den Randbereichen des Abdeckungsgebietes eine hohe Genauigkeit zu suggerieren, obwohl das Gegenteil der Fall ist. Im Falle eines impliziten Handovers, wie er bei der Plattform durchgeführt wird, kann es dann zu Ungenauigkeiten kommen, welche die Genauigkeitskriterien verletzten. Aus diesem Grund ist zudem ein Ansatz zur expliziten Erkennung von Übergängen zwischen Außen- und Innenbereichen auf Basis der Umgebungsmodelldaten aus Abschnitt 3.2 entwickelt worden, der im folgenden Abschnitt genauer vorgestellt wird.

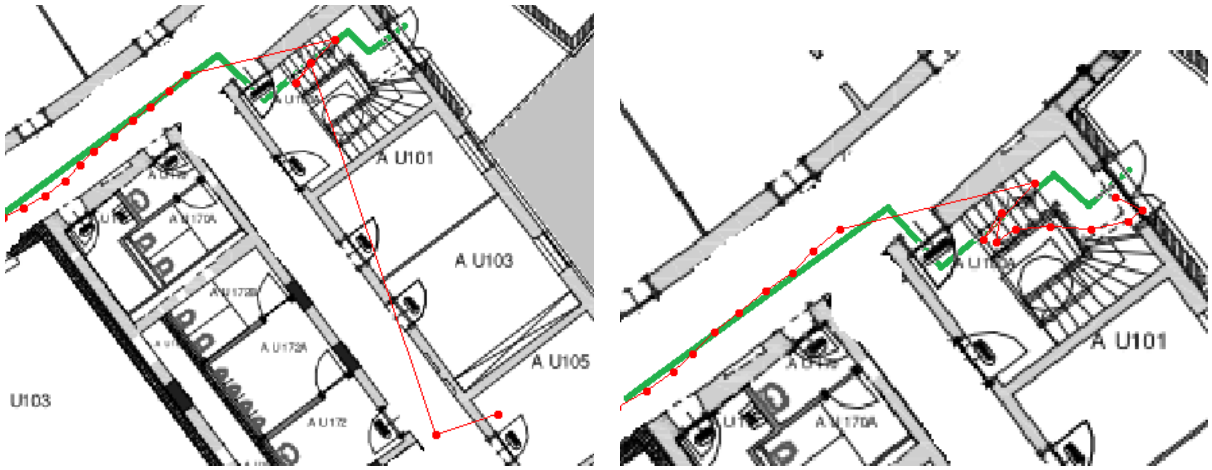
5.3.5 Explizite Übergangserkennung

In der Literatur findet man einige Arbeiten, die sich mit der nahtlosen Positionsbestimmung beschäftigen [122, 102, 24, 71, 46]. Thomas Springer verwendet in [122] GPS und WLAN zur Positionsbestimmung, wobei aus Gründen der Energieeffizienz WLAN den Vorzug vor GPS erhält, solange es verfügbar ist. Das LocateMe System von Pereira et al. nutzt hingegen immer GPS und wechselt zu WLAN oder GSM, falls GPS nicht mehr verfügbar ist [102]. Hansen identifiziert Gebäude anhand der MAC-Adressen der im Gebäude angebrachten Access Points [46]. Neben den beiden oben genannten Alternativen zum Wechsel zwischen GPS und WLAN werden zudem noch die beiden folgenden Möglichkeiten vorgestellt: Um kurzzeitigen qualitativ schlechten GPS Empfang, beispielsweise beim Passieren von Fenstern, zu vermeiden, wird GPS nur solange bevorzugt, bis das Signal verloren geht. Anschließend wird WLAN verwendet, bis wiederum kein WLAN-Signal mehr verfügbar ist. Die letzte Methode schließlich bevorzugt aus demselben Grund GPS, solange ein ununterbrochenes GPS-Signal von mindestens 5 Sekunden besteht. In seiner Dissertation entwickelt Weyn ein opportunistisches Ortungssystem, welches durch die Integration von GPS, GSM, WLAN und Schritterkennung auch zur nahtlosen Positionsbestimmung eingesetzt werden kann [139]. Dabei kommt ein Partikelfilter zum Einsatz, dessen Partikel je nach eintreffenden Sensordaten bewegt, neu gewichtet oder sonstige Eigenschaften aktualisiert werden. Bei allen genannten Ansätzen wird zwar ein Übergang zwischen Innen- und Außenbereichen erreicht, jedoch wird keine explizite Erkennung von Übergängen umgesetzt, sondern die nahtlose Positionsbestimmung allein anhand von verfügbaren Sensordaten umgesetzt.

Im Gegensatz zu den genannten Arbeiten beschreiben Gallagher et al. in [34] sogenannte Übergangszonen im Umgebungsmodell. Sie nutzen zur nahtlosen Positionsbestimmung ebenfalls WLAN und GPS. Der Übergang von innen nach außen wird dann angenommen, wenn ein Nutzer eine gewisse Zeit in einer Übergangszone lokalisiert wurde, in diesem Fall wird GPS für eine gewisse Zeit eingeschaltet. Hat der Nutzer die Zone passiert, ohne nach außen zu wechseln, so wird GPS wieder deaktiviert. Der Übergang von außen nach innen wird dadurch gehandhabt, dass WLAN aktiviert wird, sobald GPS nicht mehr verfügbar ist.

Allerdings ist der Übergang von außen nach innen für ortsbezogene Dienste deutlich wichtiger, als andersherum. Durch Assisted GPS wird innerhalb von wenigen Sekunden nach Verlassen eines Gebäudes eine hinreichend genaue GPS-Position für ortsbezogene

Dienste in Außenbereichen verfügbar. Zudem kann die Zeit dazwischen mit aktuellen Koppelnavigationsansätzen überwunden werden, ohne dass die Fehler zu sehr überhand nehmen. Wechselt man hingegen in einen Innenbereich, so ist nicht nur die Genauigkeit von GPS nicht mehr ausreichend, um viele Dienste wie Navigation zu betreiben, sondern auch die Genauigkeit der Indoor-Ortung kann zunächst eingeschränkt sein, wie die obigen Testläufe zeigen. Aus diesem Grund werden die Koordinaten der Eingänge aus dem Umgebungsmodell genutzt, um Übergänge explizit zu erkennen und entsprechend zu modellieren.



(a) Ohne Eingangserkennung (b) Mit Eingangserkennung

Abbildung 5.13: Vergleich zwischen expliziter und impliziter Übergangserkennung. Der echte Pfad (grün) und der geschätzte Pfad (rot).

Die explizite Übergangserkennung nutzt ein vierstufiges Modell:

- Ist die GPS-Position weiter als die doppelte Genauigkeit von bekannten Gebäuden entfernt, so sind Übergänge ausgeschlossen. Die Positionsbestimmung erfolgt wie gehabt.
- Liegt die GPS-Position näher am Gebäude als die doppelte Genauigkeit, so wird zusätzlich GPS mit Schritterkennung durch einem Kalmanfilter gekoppelt, um die Genauigkeit in der Nähe von Gebäuden zu erhöhen. Die Position des Kalmanfilters wird nun genutzt, um den Abstand zu den Eingängen des Gebäudes zu überprüfen.
- Befindet sich ein Eingang im Kreis um die Position mit eineinhalbfacher Standardabweichung als Radius, so wird der Partikelfilter mit den Koordinaten des Eingangs und der eineinhalbfachen Standardabweichung initialisiert und ab diesem Zeitpunkt bereits eine Indoor-Ortung mit dem Partikelfilter im Hintergrund durchgeführt. Diese wird abgebrochen, wenn sich die Position des Kalmanfilters wieder vom Eingang entfernt oder entsprechend der dynamischen Kombination neu initialisiert, sobald ein anderes System eine höhere Genauigkeit aufweist.

- Sobald GPS nicht mehr verfügbar ist, so wird GPS deaktiviert, bis sich die Indoor-Position wiederum näher als die zweimalige Standardabweichung an einem Eingang befindet. Die Indoor-Positionsbestimmung wird weitergeführt, bis eine GPS-Position mit einer höheren Genauigkeit zur Verfügung steht.

Abbildung 5.13 zeigt die Genauigkeiten beim Übergang von Außen nach Innen. Während es immer noch zu leichten Ungenauigkeiten kommen kann, sind diese doch im Vergleich zur herkömmlichen Methode deutlich gesunken, und auch beim Übergang werden die Genauigkeitskriterien eingehalten.

5.4 Zusammenfassung

In diesem Kapitel wurde insbesondere die Fragestellung erläutert, wie genaue Positionsdaten für I-LBS ohne großen Aufwand möglichst flexibel bereitgestellt werden können und wie eine nahtlose Positionsbestimmung in Innen- und Außenbereichen realisiert werden kann. Dazu wurden zuerst die einzelnen Komponenten aus Kapitel 4 in einem Partikelfilter kombiniert und anschließend das Problem des hohen Kalibrierungsaufwands von Fingerprint Ansätzen durch Backtracking gelöst. Zuletzt wurde eine Plattform zur Kombination beliebiger Sensoren zur Positionsbestimmung vorgestellt, in einem Szenario der nahtlosen Positionsbestimmung evaluiert und durch explizite Eingangserkennung weiter verbessert.

Zur automatischen Kalibrierung wurde eine Methode entwickelt, wie mithilfe von Dead Reckoning und Mapmatching durch Backtracking eine Fingerprint Datenbank automatisch ohne Interaktion von Nutzern erzeugt und durch die Nutzer aktuell gehalten werden kann. Die durch das Backtracking verfügbar werdenden genauen Positionsdaten werden genutzt, um Signalstärkemessungen automatisch mit einem Ort auf der Karte zu verknüpfen. Dadurch können mithilfe der Nutzer des Systems automatisch große Datenmengen gesammelt werden. Diese bieten durch die Kombinationsmöglichkeit von Koppelnavigation mit Fingerprinting in einem Partikelfilter nachweislich genauere Positionsdaten.

Die Plattform zur Positionsbestimmung erlaubt die Nutzung beliebiger bestehender Verfahren zur Positionsbestimmung. Die Verfahren werden durch eine Beschreibungssprache so genau beschrieben, dass sie sich mit dieser Beschreibung an der Plattform anmelden können. Die Plattform übernimmt sowohl die Aufgaben der Dienstvermittlung, welche anhand der vom Nutzer gewünschten Kriterien und der Eigenschaften der Verfahren ermittelt werden, als auch die Rolle einer Middleware, die die Sensordaten entgegen nimmt und diese zur Verarbeitung durch die einzelnen Verfahren weiterreicht. Zudem agiert die Plattform als Anbieter von Umgebungsmodellen, welche ebenfalls über die Plattform verwaltet und zur Verfügung gestellt werden. Ein Einsatzgebiet der Plattform ist die nahtlose Positionsbestimmung, da sich bekannte Ortungsverfahren in und außerhalb von Gebäuden an der Plattform anmelden können. Allerdings leidet die Genauigkeit der Plattform beim Betreten eines Gebäudes. Um dieser Problematik zu begegnen, wurde zudem ein Verfahren zur expliziten Übergangserkennung entwickelt, welches auf Basis fester Sensoren zuverlässig das Betreten von Gebäuden erkennt und somit eine qualitativ hochwertigere nahtlose Positionsbestimmung erlaubt.

Kapitel 6

Zusammenfassung und Ausblick

Ortsbezogene Dienste genießen aktuell eine hohe Verbreitung in unserem alltäglichen Leben. Doch obwohl dieses zum Großteil in Gebäuden stattfindet, gibt es nur wenige Anbieter von ortsbezogenen Diensten in Gebäuden. In dieser Arbeit wurden die Probleme in Form von nicht verfügbaren semantischen Gebäudemodellen und teurer oder für viele Dienste zu ungenauer Positionsbestimmung identifiziert und Lösungsmöglichkeiten entwickelt und evaluiert.

Das erste Ziel der Arbeit, die **automatische Erzeugung und effiziente Bereitstellung von Umgebungsinformationen** wurde durch eine automatische Erzeugung von geometrischen, topologischen und semantischen Informationen aus bestehenden CAD-Daten oder einfachen Bitmaps erreicht. Eine Hybridisierung beider Ansätze erlaubt die effiziente Anfragebeantwortung für ortsbezogene Dienste. Durch die Flexibilität, die automatische Erzeugung und die Definition weitgehend standardisierter Austauschformate ist eine flächendeckende Verbreitung von Umgebungsinformationen möglich, die für eine Vielzahl von I-LBS genutzt werden können. Das hybride Modell ist dabei bis zu fünf Mal schneller als jedes der Modelle für sich alleine und bis zu 1000-mal schneller als das für bestimmte Anfragen schlechter geeignete Modell.

Das zweite Ziel der **einfachen und flexiblen Positionsbestimmung in Gebäuden von genügender Genauigkeit und ohne zusätzliche Kosten** für I-LBS wurde durch die Entwicklung eines Partikelfilters erreicht, der eine infrastrukturlose Koppelnavigation durchführt. Bei bekannter Startposition beträgt die Abweichung von der tatsächlichen Position weniger als 2 Meter, bei unbekannter Startposition wird eine solche Genauigkeit erst nach einiger Zeit durch Mapmatching erreicht. Die Positionsbestimmung kann durch Hinzunahme von WLAN-Signalstärkeinformationen einer existierenden WLAN-Infrastruktur weiter verbessert und stabilisiert werden. Dazu wird die WLAN-Positionsbestimmung als Korrekturmodell in den Partikelfilter integriert, wodurch der Fehler der Positionsbestimmung durchgehend auf unter 2 Meter gesenkt wird. Der Aufwand zur Erzeugung von Trainingsdaten für die WLAN-Positionsbestimmung konnte durch ein Backtracking-Verfahren auf dem Partikelfilter auf ein Minimum reduziert werden. In diesem Fall werden aus den Positionsschätzungen Fingerprints erzeugt, wodurch die Trainingsdaten ohne Nutzerinteraktion automatisch generiert werden. Die Nutzung solcher automatisch generierter Trai-

ningsdaten verschlechtert die Genauigkeit der Positionsbestimmung nur geringfügig, reduziert allerdings deutlich den Zeit- und Kostenaufwand.

Zuletzt wurden **Dienstbrüche bei Übergängen zwischen Innen- und Außenbereichen beseitigt**. Dazu wurde zuerst eine Plattform zur dynamischen Auswahl und Kombination von Sensoren und Verfahren zur Positionsbestimmung analog zu einer Dienstvermittlung konzipiert. Eine Sensorbeschreibungssprache erlaubt die Charakterisierung von Dienstgüteparametern und somit eine Bewertung der zu Verfügung stehenden Verfahren. Obwohl sich dadurch - wie prototypische gezeigt wurde - Dienstbrüche beseitigen lassen, kann es gerade bei Übergängen zu einer verminderten Dienstgüte kommen. Für den wichtigen Spezialfall des Übergangs von außen nach innen wurde zu diesem Zweck ein Verfahren zur expliziten Übergangserkennung auf Basis von Kartendaten entwickelt, welches eine zuverlässige Eingangserkennung durchführt und gleichzeitig eine Initialisierungsmöglichkeit für den oben genannten Partikelfilter für Innenbereiche darstellt.

In dieser Arbeit wurden einige neue Methoden entwickelt, welche die Grundlage für die künftige Verbreitung ortsbezogene Dienste in Gebäuden bieten. Mit den vorgestellten Methoden zur automatischen Modellerzeugung ist es möglich, dass Gebäudeverwalter einem I-LBS-Anbieter Schnittstellen zu den semantischen Kartendaten anbieten, allerdings weiter die Kontrolle über die eigenen Daten behalten. Trotz der Anstrengung internationaler Konzerne, wie Google oder Nokia, ist eine einheitlich verwaltete Kartenbasis von Gebäudedaten in Zukunft nicht zu erwarten. Vereinzelte Gebäudedaten von Shoppingzentren oder Flughäfen werden zwar verfügbar sein, viele Verwaltungseinheiten zögern jedoch, ihre Daten preiszugeben. Durch diese Arbeit haben sie die Möglichkeit, selbst zum Anbieter von I-LBS in ihren Gebäuden zu werden, wobei eine komplette Lösung bis hin zur Positionsbestimmung angeboten werden kann.

Im Umfeld der ortsbezogenen Dienste in Gebäuden existieren zahlreiche Weiterentwicklungsmöglichkeiten und zusätzliche Herausforderungen. Bereits in Abschnitt 5.2.4 wurde die Notwendigkeit weiterer Versuche in Bezug auf die WLAN-Kalibrierung angesprochen. Weitere Tests in anderen Umgebungen werden dabei helfen, den Partikelfilter noch robuster und flexibler zu gestalten. Museen und Flughäfen zeichnen sich im Gegensatz zu Büroumgebungen beispielsweise durch große Räume und Hallen aus. Fabrikhallen bestehen aus Eisen und Beton, Baustoffen die einen großen Einfluss auf die genutzten Sensoren zur Positionsbestimmung haben.

Dem Vorhersagemodell des Partikelfilters liegt die Annahme zugrunde, dass der Nutzer geht. Rennt dieser, bewegt er sich auf Rädern fort oder ist auf eine Gehhilfe angewiesen, so ist eine exakte Vorhersage nicht mehr möglich. Hierfür sind Aktivitätserkennungsalgorithmen in die Vorhersage mit einzubeziehen, um in den Spezialfällen die Genauigkeit zu halten. Damit ist auch eine Zustandsüberwachung älterer Personen möglich, die Stürze erkennt und automatisch Notdienste alarmiert. Die Positionsbestimmung mithilfe des Partikelfilters kann auch visuell eingeschränkten Personen eine sichere Navigation zum Ziel ermöglichen. Dazu müssen allerdings Methoden zur audiobasierten Routenführung entwickelt werden, damit die Navigationsanweisungen auch Blinden zur Verfügung stehen.

Ein besonderer Forschungszweig beschäftigt sich mit der ortsabhängigen Zugriffskontrolle. Dabei werden manche Dienste nur an bestimmten Orten erlaubt. Die Autorisierung

erfolgt entsprechend erst, nachdem per Positionsbestimmung festgestellt wurde, dass sich der Nutzer an einem erlaubten Ort befindet. Hierbei ermöglicht die kontinuierliche Positionsbestimmung durch den Partikelfilter, eine Zugriffskontrolle nicht nur für einzlne Positionsabfragen zu definieren, sondern Einschränkungen an die erfolgte Bewegung und die zurückgelegten Pfade zu überprüfen. In sicherheitskritischen Fällen sollte hier jedoch das Vorhersagemodell so angepasst werden, dass ein Angreifer nicht durch passende Handbewegungen eine Bewegung simulieren kann, und dadurch unberechtigterweise Zugang zum gesicherten Dienst erlangt.

Neben den vielen Anwendungsgebieten gibt es jedoch auch eine Reihe von Herausforderungen bei der Positionsbestimmung, die in dieser Arbeit nur am Rande betrachtetet wurden. Ein Herausforderung, stellt der große Energieverbrauch der Positionsbestimmung dar. Je aufwändiger die Berechnungen des Partikelfilters werden, desto höher ist auch der Energieverbrauch des Prozessors. Eine ununterbrochene Positionsbestimmung mit dem mobilen Partikelfilter verbraucht grob dreimal mehr Strom als die Nutzung von GPS. Zwar sind die in Gebäuden zurückgelegten Strecken deutlich kürzer, als in Außenbereichen, und es ist meist eine Stromquelle in Form von Steckdosen verfügbar, jedoch müssen dennoch stromsparendere Mechanismen gefunden werden, um auch einen dauerhaften Einsatz der Ortsbestimmung zu ermöglichen. Neben Stromsparmechanismen muss zur breiten Nutzung unterschiedlicher Smartphones bei der WLAN-Positionbestimmung die unterschiedliche Empfangsleistung berücksichtigt werden, welche einen wesentlichen Einfluss auf die Qualität der Positionsbestimmung hat. Auf diesem Gebiet gibt es in letzter Zeit einige Fortschritte [74], so dass unterschiedliche Geräte in Zukunft die gleichen Referenzdaten zur Positionsbestimmung nutzen können.

Ähnlich zu herkömmlichen ortsbezogenen Diensten sind auch bei I-LBS die Fragen des Datenschutzes und der Sicherheit der Privatsphäre offen. Die Antwort liegt dabei nach eigener Einschätzung weniger in einer globalen Lösung als im Bewusstsein jedes einzelnen Nutzers, dass mit der Nutzung von ortsbezogenen Diensten natürlich auch persönliche Daten an Dienstanbieter weitergegeben werden müssen. Diese Thematik kann auf Bezahlvorgänge im Internet abgebildet werden, bei denen durch die Angabe von Kontodaten oder Kreditkartennummern die Preisgabe persönlicher Daten im Gegenzug zur Diensterbringung erfolgen muss.

Allen Herausforderungen zu Trotz stehen ortsbezogene Dienste in Gebäuden vor einem Durchbruch, der unser Leben ähnlich positiv beeinflussen wird, wie die ortsbezogenen Dienste selbst.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] UncertML: Describing and Exchanging Uncertainty. online, 2013. `http://www.uncertml.org/` , zuletzt besucht März 2013.

[2] Website der OpenStreetMap. online, 2013. `http://www.openstreetmap.org/`, zuletzt besucht März 2013.

[3] O. Aichholzer, F. Aurenhammer, D. Alberts, and B. G. Artner. A Novel Type of Skeleton for Polygons. In *Journal of Universal Computer Science, vol. 1, no. 12*, pages 752–761, 1995.

[4] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[5] A. Asthana, M. Cravatts, and P. Krzyzanowski. An Indoor Wireless System for Personalized Shopping Assistance. Technical report, AT&T Bell Laboratories, 1994.

[6] M. Atia, A. Noureldin, and M. Korenberg. Dynamic Online-Calibrated Radio Maps for Indoor Positioning In Wireless Local Area Networks. *IEEE Transactions on Mobile Computing*, 99(99):1–14, 2012.

[7] F. Aubeck, C. Isert, and D. Gusenbauer. Camera Based Step Detection on Mobile Phones. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[8] R. Ausbrooks, S. Buswell, D. Carlisle, G. Chavchanidze, S. Dalmas, S. Devitt, A. Diaz, S. Dooley, R. Hunter, P. Ion, M. Kohlhase, A. Lazrek, P. Libbrecht, B. Miller, R. Miner, C. Rowley, M. Sargent, B. Smith, N. Soiffer, R. Sutor, and S. Watt. Mathematical Markup Language (MathML) Version 3.0, 2010.

[9] Autodesk. DXF Reference. Technical report, Autodesk Inc., 2007.

[10] P. Bahl and V. N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00)*, volume 2, pages 775–784, 2000.

[11] P. Bahl, V. N. Padmanabhan, and A. Balachandran. A Software System for Locating Mobile Users: Design, Evaluation, and Lessons. Technical report, Mircosoft Research, 2000.

[12] R. Battiti, T. L. Nhat, and A. Villani. Location-Aware Computing: A Neural Network Model For Determining Location In Wireless LANs. Technical report, University of Trento, 2002.

[13] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110:346–359, 2008.

[14] C. Becker and F. Dürr. On Location Models for Ubiquitous Computing. *Personal and Ubiquitous Computing*, 9(1):20–31, 2005.

[15] C. Beder, A. McGibney, and M. Klepal. Predicting the expexted accuracy for fingerprinting based WiFi localisation systems. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[16] P. Bellavista, A. Küpper, and S. Helal. Location-Based Services: Back to the Future. *IEEE Pervasive Computing*, 7(2):85–89, 2008.

[17] A. M. Bernardos, J. R. Casar, and P. Tarrío. Real time calibration for RSS indoor positioning systems. In *Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[18] J. Bohn and H. Vogt. Robust Probabilistic Positioning based on High-Level Sensor-Fusion and Map Knowledge. Technical Report 421, Swiss Federal Institute of Technology, ETH Zurich, Switzerland, April 2003.

[19] M. Botts. OpenGIS Sensor Model Language (SensorML) Implementation Specification, 2007.

[20] S. Brüning, J. Zapotoczky, P. Ibach, and V. Stantchev. Cooperative Positioning with MagicMap. In *Proceedings of Workshop on Positioning, Navigation and Communication (WPNC'07)*, 2007.

[21] J. Canny. A Computational Approach to Edge Detection. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[22] E. C. L. Chan, G. Baciu, and S. C. Mak. Orientation-Based Wi-Fi Positioning on the Google Nexus One. In *6th International Conference on Wireless and Mobile Computing, Networking and Communications*, WiMob'10, pages 392–397, 2010.

[23] G. Chen and D. Kotz. A Survey of Context-Aware Mobile Computing Research. Technical report, Department of Computer Science Dartmouth College, 2000.

[24] Y. Chen, R. Chen, X. Chen, W. Chen, and Q. Wang. Wearable electromyography sensor based outdoor-indoor seamless pedestrian navigation using motion recognition method. In *Proceedings of the 2nd International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[25] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor Localization Without the Pain. In *Proceedings of the 16th International Conference on Mobile Computing and Networking (MobiCom'10)*, pages 173–184, 2010.

[26] R. Cover. xNAL Name and Address Standard. Technical report, Organization for the Advancement of Structured Information Standards, 2002. http://xml.coverpages.org/xnal.html, last visited 15.01.2010.

[27] S. Cox. Observations and Measurements-XML Implementation, 2011.

[28] B. Eissfeller, A. Teuber, and P. Zucker. Indoor-GPS: Ist der Satellitenempfang in Gebäuden möglich? Technical report, Universität der Bundeswehr München, 2006.

[29] F. Evennou and F. Marx. Advanced Integration of WIFI and Inertial Navigation Systems for Indoor Mobile Positioning. *EURASIP Journal on Advances in Signal Processing*, 2006:1–11, 2006.

[30] S.-H. Fang, J.-C. Chen, H.-R. Huang, , and T.-N. Lin. Is FM a RF-based positioning solution in a metropolitan-scale environment? A probabilistic approach with radio measurements analysis. *IEEE Transactions on Broadcasting*, 55(3):577–588, 2009.

[31] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo loalization: efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence*, pages 343–349, 1999.

[32] D. Fox, W. Burgard, and S. Thrun. Active Markov Localization for Mobile Robots. *Robotics and Autonomous Systems*, 25:195–207, 1998.

[33] E. Foxlin. Pedestrian Tracking with Shoe-Mounted Inertial Sensors. *IEEE Computer Graphics and Applications*, 25(6):38–46, 2005.

[34] T. Gallagher, B. Li, A. G. Dempster, and C. Rizos. Power efficient indoor/outdoor positioning handover. In *Proceedings of the 2nd International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[35] M. Garcia Puyol, P. Robertson, and O. Heirich. Complexity-reduced FootSLAM for Indoor Pedestrian Navigation. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, 2012.

[36] G. Gigan and I. Atkinson. Sensor Abstraction Layer: a unique software interface to effectively manage sensor networks. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP'07)*, pages 479–484, 2007.

[37] M. Goetz and A. Zipf. Extending OpenStreetMap to Indoor Environments: Bringing Volunteered Geographic Information to the Next Level. In *Proccedings of the 28th Urban Data Management Symposium*, 2011.

[38] D. Graumann and J. Hightower. Real-World Implementation of the Location Stack: The Universal Location Framework. In *Proceedings of the Fifth IEEE Workshop on Mobile Computing Systems and Applications*, pages 122–128, 2003.

[39] D. Graumann, J. Hightower, W. Lara, and G. Borriello. Real-world Implementation of the Location Stack: The Universal Location Framework. In *Proceedings of the Fifth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'03)*, pages 122–128, 2003.

[40] G. Gröger, T. Kolbe, A. Czerwinski, and C. Nagel. City Geography Encoding Standard. Technical report, Open Geospatial Consortium Inc., 2008.

[41] D. Grotzky. An indoor dead reckoning system at room level accuracy using cell phone sensors. Bachelorarbeit, Ludwig-Maximilians-Universität München, 2012.

[42] D. Gusenbauer, C. Isert, and J. Krösche. Self-Contained Indoor Positioning on Off-The-Shelf Mobile Devices. In *Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[43] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, K. R., and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, 2002.

[44] D. Hall and S. McMullen. *Mathematical Techniques in Multisensor Data Fusion*. Artech House Information Warfare Library. Artech House, 2004.

[45] J. Hallberg, M. Nilsson, and K. Synnes. Positioning with Bluetooth. In *Proceedings of the 10th International Conference on Telecommunications (ICT'03)*, volume 2, pages 954–958, 2003.

[46] R. Hansen, R. Wind, C. S. Jensen, and B. Thomsen. Seamless indoor/outdoor positioning handover for location-based services in streamspin. In *Proceedings of the 10th International Conference on Mobile Data Management: Systems, Services and Middleware (MDM'09)*, pages 267–272, 2009.

[47] C. Harr. Mobiles Navigationssystem für MedioVis. Master's thesis, Universität Konstanz, 2006.

[48] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey Vision Conference*, pages 147–152, 1988.

[49] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *Computer*, 34(8):57–66, 2001.

[50] J. Hightower, B. Brumitt, and G. Borriello. The Location Stack: a Layered Model for Location in Ubiquitous Computing. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, pages 21–28, 2002.

[51] J. Hightower, R. Want, and G. Boriello. SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength. Technical report, University of Washington, 2000.

[52] H. Hile and G. Borriello. Information Overlay for Camera Phones in Indoor Environments. In *Location-and Context-Awareness: Third International Symposium*, LoCA, pages 68–84, 2007.

[53] B. Hoffmann-Wellenhof, H. Lichtenegger, and J. Collins. *GPS – Theory and Practice*. Springer, 2001.

[54] V. Honkavirta, T. Perälä, S. Ali-Löytty, and R. Piché. A comparative survey of WLAN location fingerprinting methods. In *Proceedings of the 6th Workshop on Positioning, Navigation and Communication (WPNC'09)*, pages 243–251, 2009.

[55] S. Hotta, Y. Hada, and Y. Yaginuma. A robust room-level localization method based on transition probability for indoor environments. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, 2012.

[56] K. Kaemarungsi and P. Krishnamurthy. Properties of Indoor Received Signal Strength for WLAN Location Fingerprinting. In *1st Annual International Conference on Mobile and Ubiquitous Systems*, MobiQuitous'04, pages 14–23, 2004.

[57] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME-Journal of Basic Engineering*, Series D(82):35–45, 1960.

[58] H. Kawaji, K. Hatada, T. Yamasaki, and K. Aizawa. Image-Based Indoor Positioning System: Fast Image Matching Using Omnidirectional Panoramic Images. In *1st ACM International Workshop on Multimodal Pervasive Video Analysis*, MPVA, pages 1–4, 2010.

[59] J. Kemper, N. Kirchhof, M. Walter, and H. Linde. Human-Assited Calibration of an Angualtion based Location Indoor System with Preselection of Measurements. *International Journal on Advances in Networks and Services*, 3(1&2):82–91, 2010.

[60] M. Kessel. *Umgebungsmodelle für Ortsbezogene Dienste innerhalb von Gebäuden*. Verlag Dr. Müller, 2010.

[61] M. Kessel, P. Ruppel, and F. Gschwandtner. BIGML: A Location Model with Individual Waypoint Graphs for Indoor Location-based Services. *Praxis der Informationsverarbeitung und Kommunikation*, 33(4):261–267, 2010.

[62] M. Kessel, P. Ruppel, and F. Gschwandtner. Waypoint Graph Creation for Indoor Location-based Services. In *Proceedings of the 7th GI/ITG KuVS-Fachgespräch Ortsbezogene Anwendungen und Dienste*, pages 93–100, 2010.

[63] M. Kessel and S. Schreier. Platform for Hybrid Positioning based on a Sensor Description Language. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, 2012.

[64] M. Kessel and M. Werner. SMARTPOS: Accurate and Precise Indoor Positioning on Mobile Phones. In *Proceedings of the International Conference on Mobile Services, Resources, and Users (MOBILITY'11)*, pages 158–163, 2011.

[65] M. Kessel and M. Werner. Automated WLAN Calibration with a Backtracking Particle Filter. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, 2012.

[66] M. Kessel, M. Werner, and C. Linnhoff-Popien. Compass and WLAN Integration for Indoor Tracking on Mobile Phones. In *The Sixth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'12)*, pages 1–7, 2012.

[67] C. Kessler, M. Wankerl, and G. Trommer. Dual imu indoor navigation with particle filter based map-matching on a smartphone. In *Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[68] T. King, S. Kopf, T. Haenselmann, C. Lubberger, and W. Effelsberg. COMPASS: A Probabilistic Indoor Positioning System Based on 802.11 and Digital Compasses. In *Proceedings of the International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH'06)*, pages 34–40, 2006.

[69] M. B. Kjærgaard and C. V. Munk. Hyperbolic Location Fingerprinting: A Calibration-Free Solution for Handling Differences in Signal Strength. In *Proceedings of the 2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM '08)*, pages 110–116, 2008.

[70] T. K. Kohoutek, R. Mautz, and A. Donaubauer. Real-time Indoor Positioning Using Range Imaging Sensors. In *Proceedings of Real-Time Image and Video Processing*, 2010.

[71] N. Kohtake, S. Morimoto, S. Kogure, and D. Manandhar. Indoor and outdoor seamless positioning using indoor messaging system and gps. In *Proceedings of the 2nd International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[72] N. Kothari, B. Kannan, and M. B. Dias. Robust indoor localization on a commercial smart-phone. Technical report, Carnegie-Mellon University, 2011.

[73] A. Küpper. *Location-Based Services: Fundamentals and Operation*. John Wiley and Sons Ltd, 2005.

[74] C. Laoudias, R. Piché, and C. G. Panayiotou. Device signal strength self-calibration using histograms. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, 2012.

[75] J. Ledlie, J.-g. Park, D. Curtis, A. Cavalcante, L. Camara, A. Costa, and R. Vieira. Molé: A Scalable, User-Gnerated WiFi Positioning Engine. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[76] H. Lemelson, M. B. Kjærgaard, R. Hansen, and T. King. Error Estimation for 802.11 Location Fingerprinting. In *Proceedings of the 4th International Symposium on Location and Context Awareness (LoCA'09)*, pages 138–155, 2009.

[77] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 14th International Conference on Ubiquitous Computing (Ubicomp'12)*, pages 421–430, 2012.

[78] T. Liebich. IFC4 RC4 Documentation. online, September 2012. http://www.iai-tech.org/ifc/IFC2x3/ TC1/html/index.htm.

[79] J. A. B. Link, P. Smith, N. Viol, and K. Wehrle. FootPath: Accurate Map-based Indoor Navigation Using Smartphones. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[80] H. Liu, H. Darabi, P. Banerjee, and J. Liu. Survey of Wireless Indoor Positioning Techniques and Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(6):1067–1080, 2007.

[81] J. Liu, R. Chen, L. Pei, W. Chen, T. Tenhunen, H. Kuusniemi, T. Kroeger, and Y. Chen. Accelerometer Assisted Robust Wireless Signal Positioning Based on a Hidden Markov Model. In *Proceedings of the Position Location and Navigation Symposium (IEEE/ION PLANS'10)*, pages 488–497, 2010.

[82] D. G. Lowe. Object Recognition from Local Scale-Invariant Features. In *International Conference on Computer Vision*, ICCV, pages 1150–1157, 1999.

[83] J. Machaj and P. Brida. Optimization of rank based fingerprinting localization algorithm. In *Proceedings of the 2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN'12)*, 2012.

[84] P. Marcus, M. Kessel, and C. Linnhoff-Popien. Securing mobile device-based machine interactions by employing user location histories. In *Security and Privacy in Mobile*

Information and Communication Systems Social Informatics and Telecommunications Engineering (MOBISEC'12), pages 81–92, 2012.

[85] P. Marcus, M. Kessel, and M. Werner. Dynamic nearest neighbors and online error estimation for smartpos. *zu erscheinen in: International Journal On Advances in Internet Technology*, 6(1&2), 2013.

[86] R. Mautz and S. Tilch. Optical Indoor Positioning Systems. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[87] C. Mercer. Data Smoothing: RC Filtering and Exponential Averaging. Prosig Signal Processing Tutorials, 2003. http://blog.prosig.com/2003/04/28/data-smoothing-rc-filtering-and-exponential-averaging/.

[88] A. Miu. Design and Implemantation of an Indoor Mobile Navigation System. Master's thesis, University of Califonia at Berkeley, 2002.

[89] M. Muffert, J. Siegemund, and W. Förstner. The Estimation of Spatial Positions by Using an Omnidirectional Camera System. In *Proceedings of the 2nd International Conference on Machine Control and Guidance*, pages 95–104, 2010.

[90] A. Mulloni, D. Wagner, D. Schmalstieg, and I. Barakonyi. Indoor Positioning and Navigation with Camera Phones. *IEEE Pervasive Computing*, 8:22 – 31, 2009.

[91] W. Najib, M. Klepal, and S. B. Wibowo. MapUme: Scalable Middleware for Location Aware Computing Applications. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[92] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. *Wireless Networks*, 10(6):701–710, 2004.

[93] D. Nicklas. *Ein umfassendes Umgebungsmodell als Integrationsstrategie für ortsbezogene Daten und Dienste*. PhD thesis, Universität Stuttgart, 2005.

[94] D. Nicklas. Nexus - A Global, Active, and 3D Augmented Reality Model. In *Photogrammetic Week '07*, pages 325–334, 2007.

[95] E. Nießner. Entwicklung eines hybriden Indoor-Umgebungsmodells für mobile Geschäftsanwendungen. Masterarbeit, Ludwig-Maximilians-Universität München, 2013.

[96] J. Nykopp. Location Based Services. Technical report, GSM Association, 2003.

[97] R. Oppermann and M. Specht. A Context-sensitive Nomadic Information System as an Exhibition Guide. Technical report, German National Research Center for Information Technology - Institute for Applied Information Technology, 2000.

[98] J. O'Rourke. *Computational Geometry in C*. Cambridge University Press, 1998.

[99] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara. Accurate GSM Indoor Localization. In *Proceedings of the 1st International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[100] R. Palaniappan, P. Mirowski, T. K. Ho, H. Steck, P. Whiting, and M. MacDonald. Autonomous RF Surveying Robot for Indoor Localization and Tracking. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[101] J.-F. Pascual-Sánches. Introducing Relativity in Global Navigation Satellite Systems. *Annalen der Physik*, 16(4):258–273, 2007.

[102] C. Pereira, L. Guenda, and N. B. Carvalho. A smart-phone indoor/outdoor localization system. In *Proceedings of the 2nd International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[103] C. Portele. OpenGIS Geography Markup Language (GML) Encoding Standard. Technical report, Open Geospatial Consortium Inc., 2007.

[104] J. Preis, M. Kessel, M. Werner, and C. Linnhoff-Popien. Gait Recognition with Kinect. In *Proceedings of the First Workshop on Kinect in Pervasive Computing*, pages 1–4, 2012.

[105] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-Effort Crowdsourcing for Indoor Localization. In *Proceedings of the 18th International Conference on Mobile Computing and Networking (MobiCom'12)*, pages 293–304, 2012.

[106] A. Ranganathan, J. Al-Muhtadi, S. Chetan, R. Campbell, and M. D. Mickunas. MiddleWhere: a Middleware for Location Awareness in Ubiquitous Computing Applications. In *Proceedings of the 5th International Conference on Middleware (Middleware'04)*, pages 397–416, 2004.

[107] C. Rizos, G. Roberts, J. Barnes, and N. Gambale. Locata: A New High Accuracy Indoor Positioning System. In *Proceedings of the 1st International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[108] P. Robertson, M. Angermann, and B. Krach. Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors. In *Proceedings of the 11th International Conference on Ubiquitous Computing (Ubicomp'09)*, pages 93–96, 2009.

[109] A. Robin. OGC SWE Common Data Model Encoding Standard, 2011.

[110] K. Rothermel, T. Ertl, D. Fritsch, P. Kühn, B. Mitschang, E. Westkämper, C. Becker, D. Dudkowski, A. Gutscher, C. Hauser, L. Jendoubi, D. Nicklas, S. Volz, and M. Wieland. SFB 627 - Umgebungsmodelle für mobile kontextbezogene Systeme. Technical report, Universität Stattgart, 2006.

[111] A. R. J. Ruiz, F. S. Granja, J. C. P. Honorato, and J. I. G. Rosas. Accurate Pedestrian Indoor Navigation by Tightly Coupling Foot-Mounted IMU and RFID Measurements. *IEEE Transactions on Instrumentation and Measurement*, 61(1):178–189, 2012.

[112] P. Ruppel. *Umgebungsmodelle und Navigationsdaten für ortsbezogene Dienste in Gebäuden*. PhD thesis, Ludwig-Maximilians-Universität München, 2011.

[113] P. Ruppel, F. Gschwandtner, C. K. Schindhelm, and C. Linnhoff-Popien. Indoor Navigation on Distributed Stationary Display Systems. In *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC'09)*, 2009.

[114] S. Saha, K. Chaudhuri, D. Sanghi, and P. Bhagwat. Location Determination of a Mobile Device using IEEE 802.11 Access Point Signals. In *Proceedings of the Wireless Communications and Networking Conference (WCNC'02)*, pages 1987–1992, 2002.

[115] M. Schäfer, C. Knapp, and S. Chakraborty. Automatic Generation of Topological Indoor Maps for Real-Time Map-Based Localization and Tracking. In *Proceedings of the 2nd International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[116] T. Schardt and C. Yuan. A Dynamic Communication Model for Loosely Coupled Hybrid Tracking Systems. In *Proceedings of the 5th International Conference on Information Fusion (ISIF'02)*, pages 1236–1242, 2002.

[117] C. Schindhelm, F. Gschwandtner, and M. Banholzer. Usability of Apple iPhones for Inertial Navigation Systems. In *Proceedings of the 22nd International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC'11)*, pages 1254 –1258, 2011.

[118] T. Schwartz, C. Stahl, C. Müller, V. Dimitrov, and H. Ji. UbiSpot - A User Trained Always Best Positioned Engine for Mobile Phones. In *Proceedings of Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS'10)*, 2010.

[119] D. Serant, O. Julien, P. Thevenon, L. Ries, and M. Dervin. Testing OFDM-based Positioning Using the Digital TV Signals. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO'12)*, pages 539–543, 2012.

[120] S. A. Shafer. A Framework for Creating and Using Maps of Privately Owned Spaces. In *LoCA*, pages 174–191, 2009.

[121] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking Moving Devices with the Cricket Location System. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys'04)*, pages 190–202, 2004.

[122] T. Springer. Mapbiquitous - an approach for integrated indoor/outdoor location-based services. In *Mobile Computing, Applications, and Services*, volume 95 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 80–99. 2012.

[123] C. Stahl and J. Haupert. Taking Location Modelling to New Levels: A Map Modelling Toolkit for Intelligent Environments. In *Proceedings of the International Workshop on Location- and Context-Awareness (LoCA'06)*, pages 74–85, 2006.

[124] C. Stahl and T. Schwartz. Modeling and simulating assistive environments in 3-d with the yamamoto toolkit. In *Proceedings of the first International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[125] M. N. Stefan Steiniger and A. Edwardes. Foundations of Location Based Services. Vorlesungsskript, Department of Geography, University of Zürich, 2006.

[126] W. Storms, J. Shockley, and J. Raquet. Magnetic Field Navigation in an Indoor Environment. In *Proceedings of the 1st Ubiquitous Positioning, Indoor Navigation and Location Based Service Conference (UPINLBS'10)*, pages 1–10, 2010.

[127] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik. Indoor Localization without Infrastructure using the Acoustic Background Spectrum. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services (MobiSys'11)*, pages 155–168, 2011.

[128] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2006.

[129] P. Tozour. Search space representations. *AI Game Programming Wisdom*, 2:85–102, 2004.

[130] S. Van den Berghe, M. Weyn, V. Spruyt, and A. Ledda. Fusing Camera and Wi-Fi Sensors for Opportunistic Localization. In *Proceedings of the Fifth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM'11)*, pages 169–174, 2011.

[131] U. Walder and T. Bernoulli. Context-Adaptive Algorithms to Improve Indoor Positioning with Inertial Sensors. In *Proceedings of the 2010 International Conference on Indoor Positioning and Indoor Navigation (IPIN'10)*, 2010.

[132] H. Wang, L. Ma, Y. Xu, and Z. Deng. Dynamic Radio Map Construction for WLAN Indoor Location. In *Proceedings of the 2011 Third International Conference on Intelligent Human-Machine Systems and Cybernetics - Volume 02 (IHMSC '11)*, pages 162–165, 2011.

[133] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992.

[134] A. Ward, A. Jones, and A. Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4:42–47, 1997.

[135] M. Werner. *Ubiquitous Navigation: Skalierbare ortsbezogene Dienste in Gebäuden.* PhD thesis, Ludwig-Maximilians-Universität München, 2012.

[136] M. Werner and M. Kessel. Organisation of Indoor Navigation Data from a Data Query Perspective. In *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS'10)*, pages 1–6, 2010.

[137] M. Werner and M. Kessel. A Bitmap-Centric Environmental Model for Mobile Navigation Inside Buildings. *International Journal On Advances in Networks and Services*, 5(1 & 2):91–101, 2012.

[138] M. Werner, M. Kessel, and C. Marouane. Indoor Positioning Using Smartphone Camera. In *Proceedings of the 2011 International Conference on Indoor Positioning and Indoor Navigation (IPIN'11)*, 2011.

[139] M. Weyn. *Opportunistic Seamless Localization.* PhD thesis, University of Antwerpen, 2011.

[140] Widyawan. *Learning Data Fusion for Indoor Localisation.* PhD thesis, Cork Institute of Technology, 2009.

[141] Widyawan, M. Klepal, and S. Beauregard. A Backtracking Particle Filter for Fusing Building Plans with PDR Displacement Estimates. In *Proceedings of the 5th Workshop on Positioning, Navigation and Communication (WPNC'08)*, pages 207–212, 2008.

[142] Widyawan, M. Klepal, and D. Pesch. A Bayesian Approach for RF-based Indoor Localisation. In *Proceedings of the 4th International Symposium on Wireless Communication Systems (ISWCS'07)*, pages 133–137, 2007.

[143] T. Wilson. Keyhole Markup Language 2.2, 2008.

[144] O. Woodman. *Pedestrian localisation for indoor environments.* PhD thesis, University of Cambridge, 2010.

[145] O. Woodman and R. Harle. Pedestrian Localisation for Indoor Environments. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp'08)*, pages 114–123, 2008.

[146] O. Woodman and R. Harle. RF-based Initialisation for Inertial Pedestrian Tracking. In *Proceedings of the 7th International Conference on Pervasive Computing (Pervasive'09)*, pages 238–255, 2009.

[147] Y. Xuan, R. Sengupta, and Y. Fallah. Crowd sourcing indoor maps with mobile sensors. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, volume 73 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 125–136, 2012.

[148] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys'05)*, pages 205–218, 2005.

[149] M. Youssef and A. Agrawala. The Horus Location Determination System. In *Wireless Networks*, volume 14, pages 357–374, 2007.

[150] K. Zickuhr. Three-quarters of smartphone owners use location-based services, 2012. available at `http://pewinternet.org/~/media/Files/Reports/2012/PIP_Location_based_services_2012_Report.pdf`.