EBERHARD-KARLS-UNIVERSITÄT TÜBINGEN
Wilhelm-Schickard-Institut für Informatik
Lehrstuhl Rechnerarchitektur

Diplomarbeit

Multimodale Indoor-Lokalisierung für Android basierte mobile Endgeräte

Stephan Linzner

Betreuer: Prof. Dr. rer. nat. Andreas Zell
Wilhelm-Schickard-Institut für Informatik

Dr. -Ing. Christian Hoene
Wilhelm-Schickard-Institut für Informatik

Begonnen am: 1. September 2009

Beendet am: 25. Mai 2010

Erklärung

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

Tübingen am 25. Mai 2010

_____

Stephan Linzner

Kurzfassung. Die Allgegenwart von Computern und Kommunikationsnetzen und die voranschreitende Miniaturisierung im Bereich der mobilen Endgeräte vollzieht sich in einem signifikanten Tempo. Mobile Endgeräte der neusten Generation sind das Ergebnis einer Konvergenz von Internet, Mobilfunk und Sensortechnologien. Der Aufenthaltsort mobiler Endgeräte ist dabei von immer größerer Relevanz. Vor allem im Bereich der standortbezogenen Dienste (Location-Based-Services - LBS) und der kontextsensitiven Anwendungen stellt der Aufenthaltsort eines mobilen Endgerätes einer der wichtigsten Informationen überhaupt dar. Die Lokalisierung von mobilen Endgeräten wird dabei im Regelfall über das Global Positioning System (GPS) oder Wireless Positioning Services (WPS) bereitgestellt. Bis heute gibt es keine allgemeine Lösung zur Lokalisierung in geschlossenen Räumen. In der Vergangenheit wurden zahlreiche Ansätze zur Indoor-Lokalisierung implementiert, jedoch verwenden die meisten dieser Systeme spezialisierte Hardware. Eine neue Generation von intelligenten mobilen Endgeräten, ausgestattet mit einer Vielzahl an Funk- und Sensorentechnologien, ergänzt durch die allgegenwärtige Funk-Infrastruktur ermöglicht es, ein fein granulares Sensor-Profil einer Umgebung zu erstellen und somit einen adäquaten Ansatz zur Implementierung von Indoor-Lokalisierungssystemen (ILS) bereitzustellen.

Aufgabe und Ziel dieser Diplomarbeit ist es, Möglichkeiten und Potentiale einer multimodalen Indoor-Lokalisierung auf Android basierten mobilen Endgeräte zu evaluieren, die besten Konzepte zu identifizieren und in einem ILS-Framework zu implementieren. Die Arbeit gibt einen detaillierten Überblick über den aktuellen Stand der Wissenschaften im Bereich der Indoor-Lokalisierungssysteme. In einer umfassenden Literaturrecherche wurden über zweihundert der wichtigsten wissenschaftlichen Veröffentlichungen im Bereich von Indoor-Lokalisierungssystemen evaluiert und auf ihre Potentiale bezüglich einer adäquaten Indoor-Lokalisierung mobiler Endgeräte bewertet. Im theoretischen Teil der Arbeit wird eine detaillierte Einführung und Gegenüberstellung von Indoor-Lokalisierungs: Technologien, Techniken, Verfahren und bereits implementierten Systemen gegeben. Weiterhin wird anhand der Einschätzung der Potentiale ein geeignetes Konzept zur Realisierung eines ILS in einem theoretischen Modell abgebildet. Im praktischen Teil werden ausgehend von dem theoretischen Modell die Anforderungen und Konzepte in ein Design und eine Framework-Implementierung überführt. Im Anschluss wird eine Analyse der Lokalisierung bezüglich des Leistungsverhaltens: Genauigkeit, Zuverlässigkeit und Geschwindigkeit in Bezug auf verschiedene Technologien und Umgebungen durchgeführt. Das entwickelte Smart-Space-Framework (SSF) implementiert eine Indoor-Lokalisierung anhand des Location Fingerprinting Verfahrens für WLAN und GSM. Es ermöglicht die Umsetzung standortbezogener Lokalisierungsdienste in geschlossenen Räumen auf Android basierten mobilen Endgeräten. Es nutzt dabei ausschließlich in der Umgebung bereits verfügbare WLAN-, GSM- und Marker-Infrastruktur, unterstützt von einer Komponente zur Bewegungserkennung, zum transkribieren des aktuellen Aufenthaltsorts aus allen verfügbaren Sensordaten. Das SSF ist als Open-Source Implementierung frei verfügbar und umfasst derzeit ca. 9950 Lines of Code (LOC) davon etwa 6000 LOC Produktivcode und 4000 LOC Testcode.

Die Ergebnisse der durchgeführten Messreihen entsprechen in etwa den in der Literatur beschriebenen Genauigkeiten von 2-5 Metern. Die in der Arbeit durchgeführte Messreihe zum WLAN-Location Fingerprinting Verfahren in einem realistischen Alltagsszenario in einer einstöckigen Installation in mehreren Räumen ergab eine Zuverlässigkeit bei der Lokalisierung

von über 80% bei einer mittleren Abweichung der Genauigkeit von $1,41$ Metern. Bei der Stockwerkerkennung, der WLAN-Messreihe in einer mehrstöckigen Umgebung, konnte sogar eine Zuverlässigkeit von 95% erreicht werden.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mir bei der Erstellung der vorliegenden Arbeit mit Rat und Tat zur Seite standen. Danke an:

An Herrn Prof. Dr. Andreas Zell für die Möglichkeit meine Diplomarbeit am Lehrstuhl für Rechnerarchitektur am Wilhelm-Schickard Institut durchführen zu können und für die Vergabe dieses interessanten Themas.

An Herrn Prof. Dr. Wolfgang Rosenstiel der freundlicherweise die Zweitkorrektur dieser Arbeit übernommen hat.

Zu besonderem Dank verbunden bin ich Dr. Christian Hoene für die engagierte Betreuung und Unterstützung und für die vielen Diskussionen während meiner Diplomarbeit. Sehr freundliche Unterstützung habe ich ebenfalls von den Mitarbeitern des Ambisense Projektes erhalten. Für ihre große Bereitschaft mich weiterzubringen und mir wichtige Einschätzungen und Rückmeldungen zu geben möchte ich mein Dankeschön ausdrücken.

An meine Eltern, die meinen Studienweg stets unterstützt haben, nicht nur finanziell, sondern auch moralisch.

An meinen Kommilitonen Daniel Kersting, mit dem ich manches inhaltliches und fachliches Problem diskutieren konnte und dessen Meinung beziehungsweise Einschätzung mir stets wichtig und hilfreich war.

Inhaltsverzeichnis

1 Einführung

„Location changes everything. This one input – our coordinates – has the potential to change all the outputs. Where we shop, who we talk to, what we read, what we search for, where we go – they all change once we merge location and the Web.“

(Mathew Honan, WIRED magazine, 19.1.2009)

1.1 Indoor Lokalisierung

Die Art und Weise, wie wir in der globalisierten und hochvernetzten Welt mit Computern und Informationstechnologien umgehen, hat sich in den letzten Jahrzehnten grundlegend verändert. Betrachtet man die Entwicklung unserer Gesellschaft, vor allem in den letzten Jahren, ist ein deutlicher Trend hin zu Mobilität, Konnektivität, Kommunikation und Information zu erkennen. Wachsende Bedeutung bekommen in diesem Kontext neue Techniken und Technologien. Mit dem Aufkommen von Cloud Computing- und standortbezogenen Diensten, dem Internet der Dinge („Internet of Things“) und neuartigen Techniken zur Informationsdarstellung wie Augmented Reality, besteht eine immer größere Notwendigkeit hin zu kontextsensitiven Anwendungsprogrammen.

Den aktuellen Aufenthaltsort eines Nutzers zu kennen, erlangt wachsende Bedeutung und ist die wichtigste Information in kontextsensitiven Computersystemen. Während heutzutage im Freien, dank des Global Positioning System (GPS) oder Wireless Positioning Services (WPS), jederzeit der Aufenthaltsort mobiler Endgeräte bestimmt werden kann, gibt es derzeit keine allgemein gültige Lösung zur Lokalisierung in geschlossenen Räumen. In der Vergangenheit wurden zahlreiche Ansätze zur multimodalen Indoor-Lokalisierung untersucht und umgesetzt, jedoch verwenden die meisten dieser Ansätze spezialisierte Hardware, die nicht im Massenmarkt erhältlich ist. Darüber hinaus sind diese Systeme meist proprietär und nicht als freie Software erhältlich. Aber, eine neue Generation von intelligenten mobilen Endgeräten, ausgestattet mit einer Vielzahl an Funk- und Sensorentechnologien, ergänzt durch die allgegenwärtige Funk-Infrastruktur, vor allem im innerstädtischen Bereich, ermöglicht es, ein fein granulares Sensor-Profil einer Umgebung zu erstellen und somit einen adäquaten Ansatz zur Implementierung von Indoor-Lokalisierungssystemen (ILS) bereitzustellen.

1.1.1 Mobile Endgeräte

Waren Computer Anfang dieses Jahrhunderts zwar schon in den meisten Haushalten vorhanden, so waren diese zumeist stationär. Mussten Menschen früher „zu den Computern“ gehen, tragen zunehmend mehr Menschen diese heute in Form von mobilen Endgeräten (ME), wie Smartphones oder Netbooks[1], mit sich herum. Die Minituasierung der Computer geht

[1] Als Netbook wird eine Klasse von Computern bezeichnet, deren Rechenleistung kleiner als die üblicher Notebooks ausgelegt ist. Die tragbaren Geräte sind vor allem zur mobilen Internetnutzung gedacht. Kommunikation

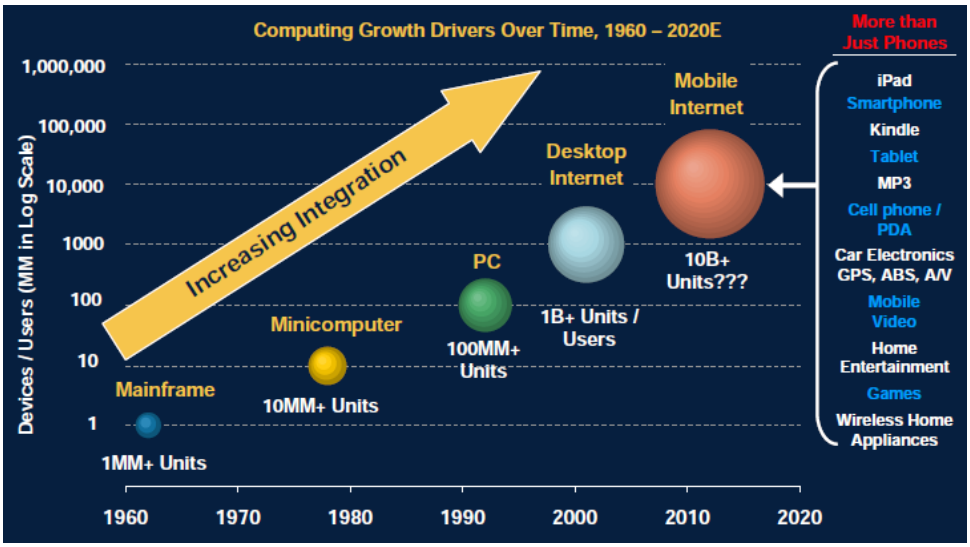


Abbildung 1.1: Chronologische Darstellung der Absatzzahlen der verschiedenen Computer Generationen von von 1960 (Mainframe) bis heute (Mobile Endgeräte), in Millionen verkaufter Geräte (Quelle: Morgan Stanley [Staa]).

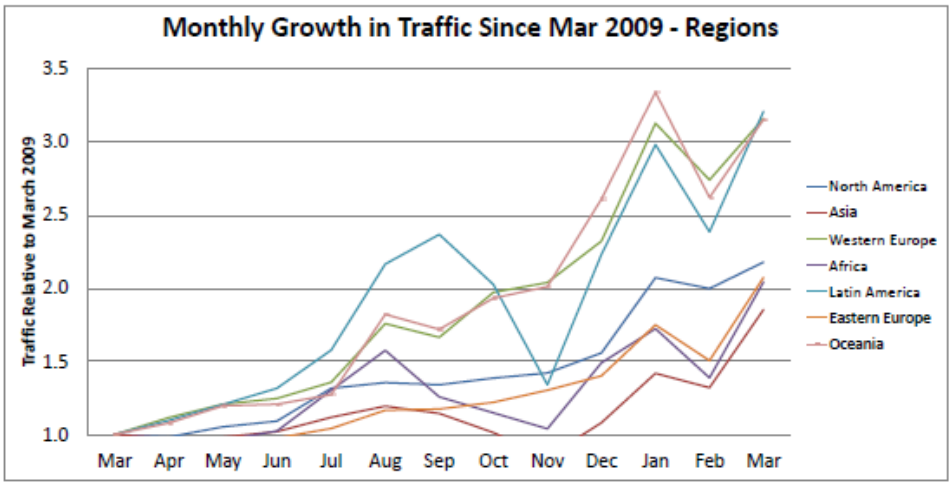


Abbildung 1.2: Weltweites relatives Wachstum des Smartphone-Traffics von März 2009 bis März 2010 (Quelle: Admob [Adm10]).

so schnell und unscheinbar voran, dass ihre Präsenz oftmals gar nicht mehr wahrgenommen wird. Eine Folge hiervon ist, dass der Zeitraum, bis diese Geräte eine kritische Masse am Markt erreicht haben, erheblich verkürzt worden ist und dass sich die absolute Anzahl der verkauften Geräte mit jeder neuen Generation verzehnfacht. Verdeutlicht wird dieser Effekt in Abbildung 1.1, welche die absolute Anzahl an verkauften Geräten pro Generation in Millionen Geräten zeigt (Quelle: Morgan Stanley [Staa]). Gerade bei den mobilen Geräten ist vor allem bei dem Form-Faktor Smartphone ein signifikantes Wachstum zu beobachten. So erwartet Morgan Stanley [Stab] bis 2013 alleine 880 Millionen globale Smartphone Nutzer. Verdeutlicht wird dieser Trend auch in Abbildung 1.2, welcher das weltweite Wachstum des Smartphone-Traffics zwischen März 2009 und 2010 zeigt (Quelle: Admob [Adm10]). Dieses in manchen Teilen der Welt beinahe exponentielle Wachstum ist eine Folge einer sehr schnell voranschreitenden Entwicklung im Smartphone Bereich, welcher innerhalb weniger Jahre völlig neue Möglichkeiten und Anwendungen auf diesen mobilen Endgeräten hervorgebracht hat. So durchdringen mobile Endgeräte inzwischen viele Bereiche unserer Gesellschaft und werden immer mehr zu einem Teil unseres alltäglichen Lebens. Sie werden auf lange Sicht den Alltag von Millionen von Menschen nachhaltig prägen und verändern. Insbesondere für die jüngere Generation sind die mobilen Endgeräte bereits heute Teil ihrer Kultur und für sie nicht mehr wegzudenken.

Spätestens mit dem Erscheinen des iPhones von Apple und der Ankündigung des Open-Source Betriebssystems Android ist eine Revolution in Gang gesetzt worden, welche eine ganz neue Generation an MEs hervorgebracht hat, Diese Entwicklung scheint nicht mehr aufzuhalten zu sein. Diese ME der neuen Generation verfügen über eine Vielzahl an Sensoren wie Beschleunigungsmesser, elektrischer Kompass, Lage-, Licht-, Näherungs- und Temperatursensoren. Weiterhin werden diverse drahtlose Technologien, wie 3G (UMTS), WLAN (802.11), Bluetooth (802.15) und in Zukunft vielleicht sogar RFID unterstützt. Zusätzlich verfügen MEs der neusten Generation in der Regel über ein GPS-Modul zur Positionsbestimmung[2].

Die Veränderungen erstrecken sich aber nicht nur auf die Hardware der MEs, sondern vor allem auch auf Veränderungen im Bereich der Software, die auf diesen neuartigen MEs läuft. Der Grund dafür ist, dass die Anforderungen heutiger MEs nicht mit denen von Desktop-Computern oder Mainframes vergleichbar sind. Gerade bei der Software Entwicklung ist in den letzten Jahren ein Fortschritt zu verzeichnen. Zum einen werden Applikation für MEs heute meistens in Form von Mini-Programmen entwickelt, den so genannten „Apps“, welche über “App-Stores„ vertrieben werden. Zum anderen wurde eine ganze Reihe neuer Betriebssysteme, speziell auslegt auf die Bedürfnisse heutiger MEs, entwickelt. Darunter das in dieser Arbeit verwendete und im Anhang C vorgestellte Android Betriebssystem, das von Google entwickelt wurde, das iPhone Betriebsystem von Apple [Appb], das Palm WebOS Betriebssystem [Pal] oder das von Nokia entwickelte Maemo[3]. Zusätzlich haben die Frameworks und Tools, die von diesen neuartigen Betriebsystemen bereitgestellt werden, dazu geführt, dass die Entwicklung auf ME sehr stark vereinfacht und vor allem die Komplexität und die Entwicklungszeiten signi-

erfolgt über eingebaute WLAN- beziehungsweise UMTS-Hardware. Netbooks haben gemeinhin eine Bilddiagonale von 7-12 Zoll.

[2]Bis zum Jahre 2011 werden nach Gartner über 75% der ausgelieferten ME über ein GPS-Modul verfügen [Gar].

[3]Maemo [Corb] wird, obwohl erst 2009 ins Leben gerufen, in seiner jetzigen Form nicht mehr lange Bestand haben. Nokia und Intel, die das Moblin Betriebsystems [Cora] entwickelt haben, gaben auf dem Mobile World Congress im Februar 2010 in Barcelona bekannt, dass die beiden Betriebssysteme zu einem neuen Betriebssystem Namens MeeGo[NC] zusammengeführt werden.

fikant verkürzt worden sind[4]. Ein Hauptgrund ist, dass jedes Betriebssystem ein so genanntes Software-Development-Kit (SDK) zur Verfügung stellt, welches einfach und intuitiv nutzbare Schnittstellen zur Verwendung von Systemdiensten und Sensoren anbietet und durch einen hohen Abstraktionsgrad die Implementierung wesentlich vereinfacht. Weil die bereitgestellte Software weit über die eines Betriebssystems hinausgeht, ist es deshalb auch angebracht, im Allgemeinen von mobilen Plattformen und nicht von Betriebssystemen zu sprechen. Hierbei gibt es einen Trend in Richtung komponentenbasierter Applikationen, welche permanent über das Internet Informationen abrufen und geräteübergreifend miteinander vernetzt sind. Dieses neuartige Ökosystem bestehend aus ME, mobiler Plattform, Cloudservices und Applikation hat eine komplette Eigendynamik entwickelt und hat eine neue Kategorie von mobilen Applikationen, die der ortsbezogenen Dienste, auch Location Based Service (LBS) genannt, hervorgebracht. Diese stellen vorrangig Dienste anhand der aktuellen Position eines Nutzers bereit und versuchen so den Gegebenheiten des aktuellen Kontextes eines Nutzers oder einer Maschine zu erkennen und auf diesen zu reagieren.

1.1.2 Auf den Aufenthaltsort kommt es an

Viele der neuesten Anwendungen gehören in den Bereich der standortbezogenen Dienste, den so genannten Location Based Services (LBS). Darunter versteht man mobile Dienste, die mit Kenntnis des Aufenthaltsortes eines Nutzers diesem spezielle Informationen oder Dienste bereitstellen. Heutige LBS können jegliche Informationen mit einer aktuellen Position versehen, von Dokumenten über Fotos und Videos bis hin zu Objekten aus der Umgebung in der realen Welt. Diese neue Art von Diensten hat vor allem Implikationen auf Anwendungen zur Filterung von Informationen aus dem Internet, dem vernetzten Arbeiten, der näherungsbasierten Interaktion mit Kollegen und Freunden, aber auch auf neuartige Darstellungen von Kontext bezogenen Informationen in den Benutzerschnittstellen, zum Beispiel durch Augmented Reality[5]. Hinzu kommt die allgemein stark anwachsende Bedeutung der Informationen über den aktuellen Aufenthaltsort eines Nutzers, welche mit zu den wichtigsten Informationen im Bereich des Context-Aware-Computing gehört.

Zur Umsetzung dieser standortbezogenen Dienste muss die Position des Nutzer bekannt sein. Die Position der Endgeräte wird heutzutage vorrangig durch das Global Positioning System (GPS) bereitgestellt. Das GPS System hat aber den einen entscheidenden Nachteil, dass es nur bei direktem Sichtkontakt zu den Satelliten im Orbit funktioniert. Sobald es innerhalb geschlossener Räume verwendet wird, hat man aufgrund der Tatsache, dass die Funkwellen nicht, oder nur sehr schwer, in Gebäude eindringen können, das Problem, dass die Position eines MEs nur unzureichend beziehungsweise überhaupt nicht festgestellt werden kann. Dies hat angesichts der Tatsache, dass sich Menschen, vor allem im städtischen Bereich, vornehmlich innerhalb von Gebäuden aufhalten, zur Folge, dass keine Bestimmung der Position vorgenommen werden kann und die Möglichkeiten neuartiger Dienste nur unzureichend ausgenutzt werden können.

[4] An dieser Stelle sei darauf hingewiesen, dass sich diese Aussage auf die Entwicklung innerhalb einer Plattform bezieht. Gemeinhin zeichnet sich der Markt für mobile Softwareentwicklung durch eine starke Fragmentierung aus. So muss bei der Entwicklung einer Applikation für mehrere Plattformen gleichzeitig entwickelt werden. Lösungsansätze in diesem Bereich können unter anderem Web-basierte Applikationen darstellen. Auf weiterführende Informationen zur Fragmentierung sei auf [oS] verwiesen.

[5] Unter Augmented Reality versteht man die Anreicherung einer Benutzerschnittstelle mit Informationen, die ein Mensch durch seine Sinne nicht wahrnehmen kann. Deshalb spricht man auch oft von Superkräften, weil Informationen in Echtzeit als zusätzliche Schicht über die eigentliche Anzeige der Benutzerschnittstelle gelegt werden.

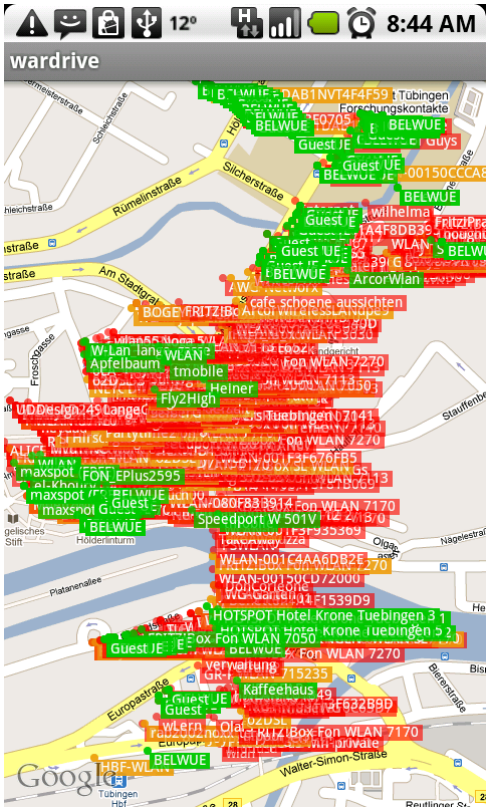


Abbildung 1.3: WLAN Abdeckung im Tübinger Innenstadtbereich. Dargestellt sind mit dem Testgerät Google Nexus One aufgezeichnete WLAN-Access-Points über einen Zeitraum von 3 Monaten. Insgesamt empfangen wurden 4137 WLAN-APs, davon 3402 verschlüsselte Netzwerke: WPA (rot), WEP (orange) und 735 unverschlüsselte Netzwerke (grün).

1.1.3 Ubiquitäre Funkinfrastruktur

Es sind aber nicht nur die Computer und die Gesellschaft, die sich weiterentwickelt und verändert haben. Ein weiterer Teil dieser Entwicklung ist die ubiquitäre Verfügbarkeit beziehungsweise die allgegenwärtige Präsenz von drahtlosen Kommunikationsnetzen, vor allem im innerstädtischen Bereich. So verfügten Ende 2009 laut Statistischem Bundesamt 82% der privaten Haushalte in Deutschland über einen Breitbandanschluss [Buna]. Hierbei ist Digital Subscriber Line mit 75% die weit verbreitetste Zugangsart zum Internet, die Anzahl der alternativen Zugangsarten unter anderem UMTS (Universal Mobile Telecommunication System), WiMAX (Worldwide Interoperability for Microwave Access) liegen bei 9%. Interessant ist auch die Tatsache, dass die Zahl der Haushalte, die stationäre Computer für den Internetzugang nutzen, von 81% im Jahre 2008 auf 76% im Jahre 2009 zurückgegangen ist und gleichzeitig die Internetnutzung mit MEs (Laptop, Notebook, Netbooks, PDAs) von 47% im Jahre 2008 auf 56% im Jahre 2009 angestiegen ist. Weiterhin beträgt die Nutzung von Smartphones zur Internetnutzung in privaten Haushalten 2009 17% und die mobile Internetnutzung durch Mobilfunktechnologien ist ebenfalls, vor allem im Bereich von Unternehmen, stark angestiegen [Bunb].

Auch wenn der direkte Anschluss an das Internet in den meisten Haushalten drahtgebunden ist, ist davon auszugehen, dass eine signifikante Anzahl dieser Haushalte, innerhalb ihrer Räum-

lichkeiten, auf die Wireless Local Area Network (WLAN) Technologie zurückgreift. Dies wird vor allem durch das Wachstum der ME beim Internetzugang deutlich. Belegt wird dies auch durch Abbildung 1.3. Diese zeigt die im Rahmen dieser Arbeit über 3 Monate erstellte Karte zur WLAN Abdeckung in der Tübinger Innenstadt. Insgesamt konnten in diesem Zeitraum 4137 WLAN-APs aufgezeichnet werden, davon 3402 verschlüsselte Netzwerke: WPA (rot), WEP (orange) und 735 unverschlüsselte Netzwerke (grün)[6]. Drahtlose Kommunikationsnetze sind deshalb heute allgegenwärtig, insbesondere ist eine hohe Penetration im innerstädtischen Bereichen als auch auf Firmen- und Universitätsgeländen zu verzeichnen. Genau dieser Umstand der hohen Abdeckung durch drahtlose Funknetze wird im Laufe dieser Arbeit zur Umsetzung einer allgemein einsetzbaren Indoor-Lokalisierungs Lösung verwendet.

1.1.4 Anwendungen

Die Anwendungsfälle zum Einsatz von ILS sind breit gefächert und können auf viele Bereiche des öffentlichen- und privaten Lebens angewendet werden. Deshalb sind im Laufe der letzten 10 Jahre bereits viele ILS-Systeme implementiert worden, manche davon sogar im produktiven Einsatz. Warum immer mehr dieser Systeme, vor allem auf Basis der verfügbaren drahtlosen Technologien wie WLAN- GSM- und Bluetooth auch kommerziell eingesetzt werden, wurde in den vorangegangen Abschnitten bereits erläutert. Dieser Abschnitt soll einen Überblick über Anwendungsfälle geben, für welche ILS eingesetzt werden können und teilweise schon eingesetzt werden. Die Anwendungsszenarien decken dabei ein breites Spektrum an Bereichen ab und reichen von Installationen im zivilen Bereich bis hin zur Installation in Gefängnissen. Anhand der Menge von möglichen Anwendungsszenarien, die nachfolgend vorgestellt werden und bei weitem nicht vollständig aufgeführt sind, wird klar, dass es im Bereich der Indoor-Lokalisierung zukünftig keine alles abdeckende Technologie beziehungsweise kein allgemeines Verfahren geben wird. Deswegen stehen die ILS auch nur bedingt in direkter Konkurrenz, sondern sind ganz im Gegenteil ein komplett neuer Markt mit Platz für viele verschiedene Technologien, Verfahren und Anwendungen [Sal].

Sicherheit und Überwachung

Im Bereich der Überwachung dominieren heutzutage kamerabasierte Systeme, mit deren Hilfe ein Wächter auf seinen Monitoren beobachten kann, wer gerade wo ist, bzw. ob sich unbefugte Menschen auf dem Gelände befinden. Wünschenswert wären jedoch Systeme, welche befugte und unbefugte Bewegungen automatisiert erkennen und eigenständig reagieren, ohne dass man das gesamte Gelände lückenlos mit Kameras erfassen muss. Dies ist insbesondere in Gefängnissen aufwendig und ein Tracking/Alarmsystem für Gefängniswärter [CGL93] würde dort zum Beispiel durch ein einfaches Gefangenentracking [PKB98] Abhilfe schaffen. Ein ähnliches Szenario ist in einer Psychiatrie denkbar, wo es gilt, den Aufenthaltsort der Patienten zu überwachen [PKB98]. Auch denkbar im Bereich der Überwachung ist ein Diebstahlschutz, bei welchem zum Beispiel ein überwachter Laptop und sein Besitzer nur gemeinsam das Büro verlassen dürfen, aber nicht mit anderen Personen [WL98].

[6] Da die Messungen nur im Freien statt fanden und nicht innerhalb von Gebäuden durchgeführt wurde, dürfte die eigentliche Anzahl an WLAN-APs in der Tübinger Innenstadt noch wesentlich höher sein.

Suche, Navigation, Rettungsmaßnahmen und Erste Hilfe

Rettungsteams der Feuerwehr und Notärzte [PKB98] sind darauf angewiesen, in unbekannten Gebäuden schnell zu einem potentiellen Unfallopfer zu gelangen. Dies würden geeignete ILS um einiges vereinfachen. Daneben ist es denkbar, eine ILS einzusetzen, um mobile Objekte aller Art (Smartphones, Notebooks) in einem Gebäude zu lokalisieren[7]. Weiterhin ist ein Einsatz eines ILS in Parkhäusern oder Tiefgaragen denkbar, um festzustellen, wo genau sich ein abgestelltes Auto befindet [WL98]. Ein weiteres klassisches Beispiel für den Einsatz eines ILS ist die Navigation zu bestimmten Kontext abhängigen Zielen, bei der die aktuelle Position auf einer Karte angezeigt wird und die möglichen Ziele in der Nähe mithilfe so genannter Points of Interest (POI) angezeigt werden [DAS01]. Insbesondere in größeren Kaufhäusern ist es denkbar, Kunden eine Produktsuche und eine Navigation zu dem gewünschten Produkt anzubieten. Der Weg durch die Gänge des Kaufhauses könnte so gewählt werden, dass Kunden an Produkten vorbei geführt werden, welche sie aufgrund ihres bisherigen Kaufverhaltens ebenfalls interessieren könnten (Consumer Based Routing, CBR) oder aber an Produkten gezielt vorbeiführen, welche aufgrund ihres Interessenprofils[8] zum jeweiligen Kunden passen könnten (Interest Based Routing, IBR). Eine ähnliche Navigationslösung wäre auch auf Messen oder in Museen sehr hilfreich.

Soziale Organisation und intelligente Vernetzung

Kollegen oder allgemein Leute in großen Bürogebäuden zu finden, kostet Zeit und unterbricht oft andere Aktivitäten. Ein weiterer Anwendungsfall könnte sein, Personen, die in einem Raum zusammentreffen, an etwas zu erinnern, was alle oder einen Teil von ihnen betrifft. Des weiteren könnte eine automatisierte Mitteilung erfolgen, wenn sich ein Bekannter im selben Gebäude aufhält.

Transport und Logistik

Objekte wie zum Beispiel Patientenunterlagen, welche insbesondere in großen Pflegeeinrichtungen oft nicht spontan auffindbar sind, könnten mit ILS jederzeit lokalisiert werden. Allgemein könnten Objekte oder Personen in irgendeiner Weise markiert und stets ohne Zeitverlust aufgefunden werden [WL98]. Ebenso wäre es denkbar, dass in einem Bürogebäude ein Fahrstuhl erkennt, wenn eine Person auf ihn zugeht und daraufhin das entsprechende Stockwerk frühzeitig anfährt [WL98].

Auch bei der automatisierten Inventur und/oder bei Standortbestimmungen von Waren jeglicher Art könnte das ILS Anwendung finden.

[7] So wird unter Zuhilfenahme des in dieser Arbeit implementierten Android basierten Software Smartspace Framework, ein ILS für ein Krankenhaus umgesetzt werden, welches die Patienten mit einfachen, vor allem billigen Android basierten ME, ausstatten wird. Die Patienten können über das ILS im Notfall grob lokalisiert werden. Darüber hinaus wird ihnen eine Navigation zu den Räumen der medizinischen Anwendungen bereitgestellt und es werden jeweils Kontext relevante Informationen eingeblendet.

[8] Hierzu könnten, unter Berücksichtigung des Datenschutzes, Daten aus sozialen Netzwerken, wie zum Beispiel das Facebook-Profil eines Kunden oder dessen Interessenprofile aus anderen Kundendatenbanken, wie Payback oder Deutschland-Card (Edeka) verwendet werden.

Protokollierung und Optimierung von Arbeitsabläufen

Die allgemeine Idee ist es, Objekten und deren Orte festzuhalten, um später diese Informationen bearbeiten oder auswerten zu können: Ein Tierarzt, der in einem Zoo seine Runde macht, notiert einfach alle seine Beobachtungen und Tätigkeiten mit seinem ME, die Position und Zeit wird dabei automatisch abgespeichert [DAS01]. So wird sichergestellt, dass er seinen Arbeitsauftrag vollständig erledigt, ohne dafür umständlich und zeitraubende Notizen machen zu müssen. Zudem könnte man zum Beispiel auch Abläufe in Krankenhäusern, Pflegeeinrichtungen oder Laboren aufzeichnen, um sie später reproduzieren, analysieren und optimieren zu können [HBB02].

Darüber hinaus könnte man einen Konferenz-Assistenten schaffen,[9] welcher dabei hilft, Vorträge und deren Zeitplanung zu verwalten. Betritt man dann den Konferenzraum, bekommt man alle kontextrelevanten Informationen und Materialien zu dem Vortrag angezeigt. Alle Aktivitäten könnten zudem aufgezeichnet und für eine spätere Analyse nach der Tagung verwendet werden [DAS01]. Eine weitere Anwendung wäre, ein intelligentes Whiteboard zu schaffen, das Audio aufzeichnet und welches Zeichnungen ausblendet, wenn Leute den Raum betreten, welche nicht berechtigt sind, diese Zeichnungen zu sehen [DAS01].

Adaption und Intelligente Reaktion

Prinzipiell ist es durch den Einsatz eines ILS möglich auf einen geänderten Aufenthaltsort automatisiert, intelligent und kontextabhängig zu reagieren. Kommt man in die Nähe eines bestimmten Ortes, werden automatisiert bestimmte Aktionen ausgelöst, zum Beispiel Navigationssysteme, die anzeigen, wann abgebogen oder gegebenenfalls gewendet werden muss [DAS01, Incb]. Das (Wohn-) Haus der Zukunft könnte aus Bewegungen der Bewohner lernen und darauf reagieren [HBB02], in dem es selbständig, je nach Aufenthaltsort, zum Beispiel das Licht, die Heizung oder die Klimaanlage regelt. Genauso könnte man Anrufe an einen Apparat in der Nähe des Anzurufenden leiten, wenn dieser ohne ME unterwegs ist [HBB02]. Ganz ähnlich ist die Idee der Context-Aware Mailinglist, welche E-mails nicht an alle Abonnenten einer Liste schickt, sondern zum Beispiel nur an die bei einer Veranstaltung anwesende Abonnenten [DAS01]. Ein weiteres Szenario ist seinen aktuellen Desktop/Browsertabs mit in andere Räume nehmen zu können. So könnte, wenn man sich einem Terminal nähert, automatisch die aktuelle Sitzung angezeigt werden [HHS+99]. Eine solche Applikation wird dann auch als Follow-me Applikation bezeichnet. Sogar einfache Sprechanlagen (Intercom) könnten intelligenter werden. Man könnte statt durch Broadcasts gezielt mit Menschen in bestimmten Räumen sprechen, ohne sie suchen zu müssen. Auch könnte das System davor warnen, wenn zusätzliche Leute im Raum sind und somit die Diskretion nicht mehr gewahrt ist [DAS01].

1.2 Ziele dieser Arbeit

In der Vergangenheit wurden diverse Ansätze zur Indoor-Lokalisierung untersucht und umgesetzt. Jedoch verwenden die meisten dieser Ansätze spezialisierte Hardware, die nicht im

[9] In diesem Bereich gibt es ebenfalls ein interessantes Projekt Namens „The Conference App“ der Samsung Mobilers [Bla], welche zukünftig eine ILS-Lösung auf Basis des in dieser Arbeit entwickelten Smartspace Frameworks bereitstellen soll. Die Applikation stellt alle für einen Event relevanten Informationen, wie Veranstaltungsprogramm und Teilnehmerlisten, direkt auf einem Android basierten ME bereit und soll später die Positionen aller Teilnehmer anzeigen.

Massenmarkt zur Verfügung steht. Ziel der Arbeit ist es, die Identifizierung und Gegenüberstellung von Techniken zur Indoor-Lokalisierung auf mobilen Endgeräten, unter Verwendung neuester MEs auf Basis der von Google entwickelten und später noch genauer vorgestellten Betriebssystem-Technologie Android. Zusätzlich sollen anhand der theoretischen Gegenüberstellung, die besten Konzepte ausgewählt und in einem Framework implementiert werden. Hierbei soll der Fokus stets darauf gelegt werden, wie Anforderungen heutiger MEs am besten abgebildet werden. Das heißt, dass die ausschließlich bereits vorhandene Funk-Infrastruktur aus der unmittelbaren Umgebung genutzt wird, eine angebrachte Technik zur Sensor-Fusion verwendet wird, ohne dabei auf spezielle Hardware auf den ME angewiesen zu sein. So kann die eingangs erörterte Tatsache ausgenutzt werden, dass heutzutage zunehmend mehr Menschen ein ME besitzen und eigentlich überall, wo sie sich gerade aufhalten, Funkinfrastruktur, welche zur Lokalisierung passiv eingesetzt werden kann, verfügbar ist. Dies trägt dazu bei, dass die Indoor-Lokalisierung zukünftig genauso ubiquitär verfügbar ist wie die MEs und das Internet. Dabei sollen neben WLAN, GSM auch die Inertial Sensoren für Bewegung, Lage und Richtung mit einbezogen werden und zusätzlich Positionskorrekturen über den Touchscreen oder über die Kamera entgegen genommen werden können.

Diese Diplomarbeit trägt zur wissenschaftlichen Aufarbeitung der Herausforderung „Indoor-Lokalisierung“ wesentlich bei:

1. Die Aufarbeitung und Gegenüberstellung von Indoor-Lokalisierung: Technologien, Techniken und Verfahren und die Einschätzung und Diskussion zur Verwendung auf Android basierten mobilen Endgeräten.

2. Die Implementierung eines multimodalen Location Fingerprint ILS-Frameworks zur Indoor-Lokalisierung auf Basis der Android Plattform, unter Verwendung der im theoretischen Teil identifizierten Technologien, Techniken und Verfahren.

3. Eine Performanz Analyse (Genauigkeit, Zuverlässigkeit und Geschwindigkeit) der Lokalisierung mittels des implementierten Smartspace Frameworks, unter Verwendung des Location Fingerprinting Verfahrens, in Bezug auf verschiedene Technologien und Umgebungen.

1.3 Gliederung

Das erste Kapitel gibt eine Einführung in die Intention der Arbeit. Des weiteren werden die Ziele und der Aufbau beschriebenen.

Das zweite Kapitel vermittelt die Grundlagen der Indoor-Lokalisierung. Es soll dazu beitragen, die Terminologie von Lokalisierungssystemen besser zu verstehen und einordnen zu können. Es werden die allgemeinen Begrifflichkeiten der Lokalisierung beschrieben und eine Einführung in die allgemeinen Eigenschaften zur Signalausbreitung von Funkwellen gegeben.

Das dritte Kapitel gibt einen umfassende Einführung in ILS. Vorgestellt werden verschiedene Technologien, Techniken und Verfahren, die bei der Indoor-Lokalisierung eingesetzt werden können. Darüber hinaus gibt es einen Überblick über bereits implementierte ILS und zeigt auf, mit welchen Technologien diese realisiert worden sind.

Das vierte Kapitel gibt eine Einführung in die Methoden und Techniken zur Lokalisierung anhand des Location-Fingerprinting (LFPT) Verfahrens. Vorgestellt werden: WLAN-LFPT-Technik, GSM-LFPT-Technik, LFPT-Algorithmen, Architektur und allgemeine Systemparameter von LFPT-ILS.

Das fünfte Kapitel beschreibt die Anforderungen an das implementierte Smartspace Framework, gibt einen Überblick über die dem Smartspace Framework zugrunde liegenden Konzepte und untersucht die Potentiale der Android Plattform zur Indoor-Lokalisierung. Schließlich wird das aus dem Konzept abgeleitete theoretische Modell zur Implementierung des Smartspace Framework vorgestellt.

Das sechste Kapitel stellt die Implementierung des Smartspace Framework anhand des theoretischen Modells vor und gibt einen detaillierten Überblick über die wichtigsten Design-, Architektur- und Implementierungsentscheidungen. Dieser Teil ist auch für Entwickler gedacht, welche die interne Arbeitsweise des Smartspace Framework verstehen und erweitern möchten.

Das siebte Kapitel stellt die Ergebnisse der im Laufe dieser Arbeit durchgeführten Messungen zum Leistungsverhalten des Smartspace Framework vor. Dabei wird die Genauigkeit, Zuverlässigkeit und Geschwindigkeit der Lokalisierung unter Verwendung des implementierten Smartspace Frameworks, bezüglich verschiedener Technologien, Umgebungen und Infrastrukturmetriken untersucht.

Im den letzten beiden Kapiteln werden zunächst die einzelnen Schritte der Diplomarbeit repetiert. Dabei findet ein Rückblick und eine Diskussion der gesamten Arbeit statt. Ebenso werden die Erkenntnisse, die sich im Verlauf der Arbeit ergeben haben, diskutiert und schließlich werden die erreichten Ziele zusammengefasst und erörtert. Abschließend wird anhand der gewonnenen Erkenntnisse ein Ausblick auf zukünftige Erweiterungen der Arbeit und Potentiale rund um das Smartspace Framework gegeben.

2 Grundlagen und Einordnung von Lokalisierungssystemen

Zum besseren Verständnis der Arbeit vermittelt dieses Kapitel die Grundlagen von Lokalisierungssystemen. Es soll dabei helfen, die Terminologie von Lokalisierungssystemen besser zu verstehen und einordnen zu können. Im ersten Teil werden allgemeine Begrifflichkeiten rund um das Thema Lokalisierung beschrieben. Im zweiten Teil wird eine Einführung in allgemeine Eigenschaften zur Signalausbreitung von Funkwellen gegeben.

2.1 Lokalisierung, Ortung, Tracking

Unter Lokalisierung (oder auch Positions-, Ortsbestimmung oder Verortung) versteht man die Ermittlung des aktuellen Standortes eines Objektes im Raum. Man unterscheidet absolute und relative Referenzsysteme. Beide werden im weiteren Verlauf dieses Kapitels genauer beschrieben. Lokalisierung im speziellen bezeichnet den Vorgang der autonomen Ortsbestimmung ohne externe Hilfsmittel wie Lokalisierungsserver. Lokalisierung hilft einem Objekt im Raum, sich die Frage zu beantworten: „Wo bin ich?“ [Mut09], in Bezug auf ein globales Referenzsystem und bezeichnet somit die Selbstverortung eines mobilen Objektes im Raum.

Der Begriff der Ortung ist dem der Lokalisierung sehr ähnlich und unterscheidet sich vorrangig dadurch, dass ein statisches oder mobiles Objekt von einem externen System geortet wird, ohne dass das ME selber aktiv an der Ortung beteiligt ist. Es wird also der Frage nachgegangen: „Wo befindet sich Objekt X?“. Ortungssysteme basieren also meist auf Fremdverortung und ermöglichen ein „Tracking“ der aktuell im Raum gegenwärtigen mobilen Objekte.

Tracking ist ebenfalls ein in der Literatur oft verwendeter Begriff und ist eng mit Ortungssystemen gekoppelt. Vor allem mit solchen, bei denen die Ortung zentral organisiert ist. Tracking Systeme erlauben die Aufzeichnung kontinuierlicher Positionsströme aller im System befindlichen Objekte. Da sich anhand der erfassten Daten eines Tracking Systems detaillierte Bewegungsprofile erstellen lassen, sind solche Systeme anfällig für Datenschutzverletzungen. Ein Beispiel für ein Ortungs-/Tracking-System ist das Active Badge [WHFG92] oder Active Bat [Cam] System. Ein Lokalisierungssystem ist hingegen das allgemein bekannte Global Positioning System (GPS) [Guo07].

In dieser Arbeit wird vorrangig der Begriff der Lokalisierung beziehungsweise der des Indoor-Lokalisierungs-Systems (ILS) benutzt. Der Grund ist, dass es sich bei der später vorgestellten Implementierung des SmartSpace Framework (SSF) um ein passives Lokalisierungssystem handelt. SSF ist in der Lage, selbständig festzustellen, wo sich ein ME relativ zu seinem Referenzsystem befindet.

2.2 Physikalische und symbolische Position

Die von einem Lokalisierungssystem gelieferten Positionen können in zwei Teile unterschieden werden, physikalische und symbolische Position [HB01]. Eine physikalische Position stellt eine absolute Position in Form von Koordinaten dar. Ein typisches Beispiel einer physikalischen Position ist eine Geo-Koordinate, wie sie vom GPS geliefert wird. Die GPS Geo-Koordinate für das Wilhelm-Schickard-Institut Tübingen ist gegeben durch 48° 32' 5.00", 9° 4' 18.76". Gegebenenfalls ist diese physikalische Positionsangabe noch durch eine Höhenangabe in Metern ergänzt.

Eine symbolische Position hingegen beschreibt eine Position auf eine abstrakte Art und Weise. Symbolische Positionen sind deshalb meist auch für Menschen leicht verständlich. Ähnlich dem aus dem Internet bekannten Domain Name System (DNS), bringt eine symbolische Position eine physikalische Position in eine menschenlesbare Darstellung. Symbolische Positionen sind auch nicht an eine physikalische Geo-Koordinate gebunden, sondern können eigenständig verwendet werden. Sie sind normalerweise eher grobgranular, das heißt, dass ein Gebiet bestimmten Ausmaßes, ein Raum oder ein ganzes Stockwerk abgebildet wird. Dies steht im Gegensatz zu physikalischen Positionen, welche eine sehr hohe Genauigkeit und somit eine feinere Granularität aufweisen. Einige Beispiele für Lokalisierung anhand symbolischer Positionen in geschlossenen Räumen sind:

- Raumnummern in Bürogebäuden,
- Identifizierung von Regalen in einer Bibliothek anhand der Regalbezeichner,
- Vergabe von symbolischen Position in einem Haushalt (Küche, Wohnzimmer, Arbeitszimmer etc.).

Eine Art von symbolischen Positionen wird von Menschen fast täglich verwendet. Städtenamen, Straßennamen und Hausnummern werden häufig zur Orientierung benutzt, wenn mit dem Auto zu einem bestimmten Ziel navigiert werden soll. Würde man anstatt der symbolischen eine physikalische Position in Form von Geo-Koordinaten verwenden, hätten Menschen ohne elektronische Unterstützung sehr große Probleme, sich in diesen unbekannten Umgebungen zurechtzufinden.

Ein starker Trend in Richtung symbolischer Positionen wird vor allem durch eine neue Generation von mobilen, cloudbasierten GPS-Navigationsystemen[1] vorangetrieben. Diese neue Generation von Navigationssoftware erlaubt es nicht nur, wie in den bisherigen Navigationssystemen, nach Name, Straße und Ort zu suchen, sondern auch nach allgemeinen Geo-Referenz Begriffen, wie „Metropolitan Museum of Arts". Da diese Navigationssysteme permanent mit dem Internet verbunden sind, kann über eine Technik namens Geocoding (GC) die zu dem Begriff zugehörige physikalische Position ermittelt werden.[2][3] Einen guten Überblick über verfügbare Geocoding APIs gibt [Geo]. Die vorgestellten Geocoder sind auch Teil aller neuerer mobilen

[1] Vorreiter ist hier Google mit der Google Maps Navigation Software [Incb] auf ME mit Android Betriebssystem.

[2] Geocoding funktioniert auch in die entgegengesetzte Richtung und wird dann als Reverse-Geocoding bezeichnet.

[3] Die Übersetzung der symbolischen Koordinate in ihre physikalische Darstellung und umgekehrt wird meistens von einem Webservice übernommen.

Plattformen, unter anderem Google Android Geocoder [Alle] und das iPhoneOS Mapkit Framework [Appa]. Außerdem stellen immer mehr Anbieter solche Dienste auch als Javascript-Bibliothek bereit. Hierzu zählen unter anderem die Google Data Protocol Maps API [Inca] und die Yahoo Geocoding API [Incc]. Beide können über den Browser des ME angefragt werden. Im Falle des „Metropolitian Museum of Arts“ liefert eine Anfrage über den Android OS Geocoder folgende physikalischen Koordinaten zurück: +40° 46’ 43.44“, -73° 57’ 43.60“. Interessant ist die Tatsache, dass die Abfrage weder eine Stadt noch eine genaue Adresse enthält und die Position lediglich über eine symbolische Position abgefragt wurde.

Es ist denkbar, dass sich in Zukunft solche Dienste mit riesigen Datenbanken im Hintergrund auch in geschlossenen Räumen durchsetzen werden. Vor allem das später vorgestellte Verfahren des Mobile-Taggings, welches anhand der Erkennung optischer Marker arbeitet, eignet sich hierfür besonders gut.

2.3 Absolute und relative Lokalisierung

Man unterscheidet Lokalisierungssysteme in absolute und relative Systeme. Bei der absoluten Lokalisierung wird die Lokalisierung unabhängig von der vorherigen Position des zu lokalisierenden Objekts durchgeführt [HB01]. Alle zu lokalisierenden Objekte in einem solchen System nutzen dasselbe absolute Referenzsystem, um sich zu lokalisieren. Das System muss sozusagen immer Kenntnis über die exakte Position besitzen. Das Paradebeispiel für eine absolute Lokalisierung ist die Angabe von Längen- und Breitengraden. Eine darauf aufsetzende Technik, die sich diese Kartographie zu Nutze macht, ist das GPS. So liefern zwei unterschiedliche GPS Empfänger bei der Lokalisierung immer annähernd dieselbe Position zurück.[4]

Anders als bei der absoluten Lokalisierung muss bei der relativen Lokalisierung die vorherige Position des Objektes bekannt sein. Eine relative Position bezieht sich dabei immer auf die zuvor bestimmte Position, um in Abhängigkeit von dieser eine Positionsangabe zu machen. So wäre die Angabe in der Form, $5m$ links oder $5m$ rechts von der zuvor ermittelten Position, eine mögliche Positionsangabe. Diese Technik kann autonom von der Umwelt, in der sie eingesetzt wird, eine Positionsverschiebung messen und eine Lokalisierung durchführen. Relative Lokalisierung kann mit absoluter Lokalisierung kombiniert werden. So kann bei Ausfall einer der beiden Techniken immer noch eine Lokalisierung vorgenommen werden. Ein Beispiel für relative Ortung ist die Suche nach Lawinenopfern, sofern sie einen Signalgeber tragen. Jeder Lawinenhelfer kann die Richtung, aus der das Signal eines Verschütteten kommt, mit seinem Gerät empfangen. Dabei werden keine absoluten Daten bezüglich der Position der Opfer auf den Empfängern angezeigt, sondern nur die Richtung angepeilt, in die ein Lawinenhelfer sich bewegen muss, um die Position des Opfers ausfindig zu machen.

2.4 Genauigkeit und Auflösung

Leistungskriterien in der Nachrichtentechnik sind grundsätzlich von denen eines ILS verschieden. Wird bei Telekommunikationssystemen die Bit Error Rate (BER) als Leistungskriterium herangezogen, haben ILS andere Kriterien zur Leistungsevaluation [PLM02]. So dienen vorrangig die Metriken Genauigkeit und Zuverlässigkeit zu den Leistungskriterien eines ILS. Das

[4] Vorausgesetzt, die Empfänger werden an der gleichen Position platziert.

heißt, mit welcher Auflösung eine Lokalisierung in einer Indoor-Umgebung durchgeführt werden kann und mit welcher Zuverlässigkeit dies geschieht. Die Genauigkeit kann als der Fehler des Abstandes zwischen der berechneten Position und des tatsächlichen Standorts des zu lokalisierenden ME verstanden werden. Die Genauigkeit eines ILS, welches das in Kapitel 4 vorgestellte Location Fingerprint Verfahren verwendet, ist zum Beispiel stark von der Granularität der gewählten Trainingspunkte bei der Datenaufnahme zur Erstellung des Innenraummodells und der daraus resultierenden Auflösung des ILS abhängig.

2.5 Sensoren

Sensoren bilden die Grundlage der Indoor-Lokalisierung. Sensoren sind mit die wichtigsten Komponenten bei der Realisierung eines ILS. Sie liefern die Daten, die benötigt werden, um eine Lokalisierung durchzuführen. Der Sensorbegriff ist sehr generisch und kann je nach Verwendung anders ausgelegt werden. Da dieser Begriff aber so wichtig ist und in dieser Arbeit immer wieder vorkommt, soll hier eine allgmeine Definition des Sensorbegriffs und die Verwendung des Begriffs in dieser Arbeit beschrieben werden:

> „*Aufnehmer (Sensoren) formen zuerst in Messeinrichtungen und ganzen Messsystem, die zu messenden nichtelektrischen physikalischen Messgrößen in elektrische Messsignale um. Diese elektrischen Messsignale werden, [. . .] so aufbereitet und umgeformt, dass genormte analoge Messsignale gebildet werden. [. . .]. Nach Messwertverarbeitung stehen dann die gewünschten Informationen zur Verfügung, die analog oder digital ausgegeben werden können.*“
>
> (Tränkler [Tra89])

> „*A transducer that converts a physical phenomenon such as heat, light, sound or motion into electrical or other signals that may be further manipulated by other apparatus.*“
>
> (Zhoa et al. [ZG04])

> „*Sensor is a device that responds to physical stimulus (as heat, light, sound, pressure, magnetism, or a particular motion) and transmits resulting impulse (as for measurement or operative control)*“
>
> (Kaemarungsi et al. [Kae05])

> „*The underlying services accessed by applications today are not just device components and operating system features, but data subsystems: locations, social networks, indexes of web sites, speech recognition, image recognition, automated translation. It's easy to think that it's the sensors in your device – the touch screen, the microphone, the GPS, the magnetometer, the accelerometer – that are enabling their cool new functionality. But really, these sensors are just inputs to massive data subsystems living in the cloud.*“
>
> (Tim O'reilly [Rad])

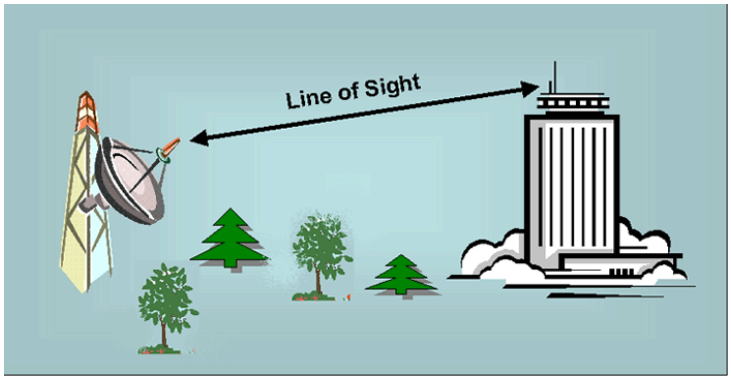


Abbildung 2.1: Direct line of Sight (DLOS). Es besteht eine direkte Sichtverbindung zwischen Sender und Empfänger [Zai].

2.6 Allgemeine Eigenschaften zur Signalausbreitung von Funkwellen

Gemeinhin breiten sich elektromagnetische Wellen im freien Raum mit Lichtgeschwindigkeit aus. Dies gilt aber nur für die Ausbreitung im Vakuum. Die reale Ausbreitungsgeschwindigkeit ist stark vom Medium abhängig, in dem sich die elektromagnetische Welle bewegt. Des weiteren gilt, dass die Signalausbreitung einer elektromagnetischen Welle mit $T_x \approx \frac{T_x}{r^2}$ mit Sendeleistung T_x in Watt und Radius r in Metern abfällt [KK04], wenn keine Hindernisse vorhanden sind. Gerade aber in geschlossenen Räumen wird die Signalausbreitungscharakteristik noch von einer Vielzahl anderer Faktoren bestimmt. Die Signalausbreitung zeichnet sich dort vor allem durch starke Multipath-Effekte, Abschattung, Reflexion, Refraktion und Streuung aus [Sch03].

Elektromagnetische Strahlen $> 30\,MHz$ folgen bei Ihrer Ausbreitung mehr oder weniger einer geraden Linie, der Sichtverbindung, auch Line of Sight (LOS) genannt. Unter LOS versteht man in der Nachrichtentechnik eine Funkverbindung mit direkter Sichtverbindung zwischen Sender und Empfänger. Man unterscheidet in direct Line of sight (DLOS) und No Line of sight (NLOS). Die direkte Sichtverbindung DLOS zwischen Sender und Empfänger ist in Abbildung 2.1 dargestellt. Nachfolgend werden die Begrifflichkeiten DLOS und NLOS genauer definiert.

DLOS bezeichnet die Existenz einer direkten Sichtverbindung zwischen Sender und Empfänger. In geschlossenen Räumen besteht eine direkte Sichtverbindung genau dann, wenn ein WLAN-AP und ein ME direktem Sichtkontakt ausgesetzt sind und die Sichtlinie nicht durch Objekte wie Mauern, Schränke oder Türen verdeckt ist. Pahlavan et al. [PLM02] weisen aber im Hinblick auf Feldstärke basierten ILS zurecht darauf hin, dass vor allem in einem ILS der DLOS nicht unbedingt auch gleich das stärkste Signal sein muss. Grund ist die Komplexität der Umgebung in geschlossenen Räumen.

NLOS bedeutet, dass kein direkter Sichtkontakt zwischen Sender und Empfänger besteht und das Funksignale somit Abschattung, Reflektion und Dämpfung durch Objekte der Umgebung ausgesetzt sind. Bei den meisten Funkübertragungen, vor allem in geschlossenen Räumen, handelt es sich um NLOS-Verbindungen.

Die Tatsache, dass es sich bei den meisten Funkübertragungen um NLOS-Übertragungen handelt, legt nahe, Orte anhand Ihrer Sender-Empfänger-Beziehung zu klassifizieren und das Modell noch weiter zu verfeinern. Pahlavan et al. [PKB98] stellen in einer weiteren Arbeit eine Taxonomy von Orten vor, welche die grobe Einteilung in die vorgestellten Klassen DLOS und NLOS weiter verfeinert. Eine weitere Verfeinerung der Granularität dieser Klassen wird durch die sogenannten Kanal-Profile beschrieben. Kanal-Profile sind in Dominant Direct Path (DDP), Nondominat Direct Path (NDDP) und Undetected Direct Path (UDP) unterteilt und nehmen eine Abstufung der Funkwellencharakteristiken von „Signal vorhanden“ bis „kein Signal vorhanden“ vor.

DDP Ein DLOS Pfad kann vom Empfangssystem direkt entdeckt werden und ist auch der dominante Pfad im Kanal. Ein GPS-Empfänger kann anhand der vorliegenden DLOS direkt die Time of Flight ableiten und so die Entfernung zu einem Satelliten im Orbit abschätzen.

NDDP Der DLOS Pfad kann vom Empfangssystem entdeckt werden, ist aber nicht unbedingt der dominante Pfad im aktuellen Kanal. Um die Signallaufzeit adäquat zu messen, müssen hier bessere Receiver, die intelligente Entscheidung bezüglich der Multipath-Effekte treffen können, verwendet werden. So kann der Fehler in der Genauigkeit so gering wie möglich gehalten werden. Weiterhin impliziert dies, dass diese Orte sich dadurch auszeichnen, dass zwar im Prinzip ein DLOS Pfad besteht, dieser jedoch nicht der stärkste Pfad ist, der von dem System empfangen werden kann.

UDP Das Empfängersystem kann den DLOS Pfad nicht finden und hat keine Möglichkeit, etwas zu empfangen. Dieser Kanal repräsentiert also Orte, an denen nur unzureichend Signale vorliegen. Signale können mit keinem Empfänger empfangen und verwertet werden.

2.6.1 Weitere Effekte bei der Signalausbreitung

Zusätzlich zu der Signalausbreitungscharakteristik kommen bei WLAN und GSM Netzen weitere Effekte hinzu, die dadurch entstehen, dass die LOS durch Hindernisse verdeckt ist. Elektromagnetische Wellen breiten sich mit Lichtgeschwindigkeit aus und weisen die Eigenschaft auf, sich je nach Verhältnis von Wellenlänge zu Größe der Hindernisse, auf welches die Wellen auftreffen, ein geändertes Verhalten aufzuzeigen. Da dieses Verhalten vor allem in geschlossenen Räumen aufgrund der komplexen Umgebungen häufig auftritt, ist es wichtig, sich mit diesen Effekten vertraut zu machen. Nur so kann die Signalausbreitung in geschlossenen Räumen besser verstanden und Lösungen daraus abgeleitet werden.

2.6.1.1 Interferenz

Interferenz tritt immer genau dann auf, wenn zwei Sender gleichzeitig auf derselben Frequenz, am selben Ort senden, und sich gegenseitig stören. Dies ist vor allem in WLAN Netzen ein Problem, weil es den Datendurchsatz vermindert oder zu einem kompletten Zusammenbruch des Netzes führen kann, weil das Rauschen auf dem Kanal zu groß wird. In GSM und UMTS Netzen wirkt man diesem Effekt durch Multiplexing Verfahren wie Frequency Division Multiplexing (FDM), Time Division Multiplexing (TDM) und Code Division Multiple Access (CDMA) entgegen.

2.6.1.2 Abschattung, Reflektion und Refraktion

Abschattung bedeutet, dass der DLOS Pfad eines Signals von einem Objekt blockiert wird. Objekte können Berge, Gebäude, Wände oder Bauwerke aller Art sein. Bei der Reflektion handelt es sich um eine spezielle Form der Abschattung, bei der die Strahlen von einem Objekt reflektiert werden. Reflektion tritt genau dann auf, wenn eine elektromagnetische Welle eine Wellenlänge größer oder gleich dem Objekt besitzt, auf das sie auftrifft. Tritt dieser Fall ein, wird das Signal von dem Objekt zurückgeworfen und verändert somit seine Richtung und erfährt zusätzlich eine Abschwächung des Signals. Zusätzlich zur Reflektion kommt noch die Refraktion. Die Ausbreitungsgeschwindigkeit von elektromagnetischen Wellen hängt stark von dem Medium ab, durch das die Wellen sich bewegen. Nur im Vakuum beträgt diese Lichtgeschwindigkeit. Pflanzen sich die Wellen in einem Medium mit höherer Dichte fort, kommt es unweigerlich zu einer Ablenkung der elektromagnetischen Wellen. Diese Ablenkung wird als Refraktion bezeichnet und ist der Grund, warum Funkwellen von der Atmosphäre auf die Erde zurückgeworfen werden [Sch03]. Abschattung, Reflektion und Refraktion eines Signals treten genau dann auf, wenn ein Objekt größer oder gleich der Wellenlänge des ausgesendeten Signals ist.

2.6.1.3 Streuung und Beugung

Wenn die Wellenlänge des Signals kleiner als das Objekt ist, auf das es auftrifft, kommt es zur Streuung oder Beugung des Signals. Mit Streuung bezeichnet man den Effekt, dass ein Signal in mehrere schwächere Signale aufgespalten wird. Dieses Phänomen tritt vor allem bei GSM und WLAN Netzen sehr häufig auf. Der Grund ist, dass die Wellenlänge von GSM im $900\,MHz$ Band (in Europa) in etwa $35\,cm$ und die Wellenlänge bei WLAN ($2{,}4\,GHz$) in etwa $12\,cm$ beträgt. Deshalb führen viele Objekte in den Umgebungen, in denen GSM und WLAN eingesetzt wird, zu einer Streuung des Signals. So werden GSM Signale zum Beispiel schon von einem Straßenschild nicht mehr reflektiert, sondern gestreut. Ein weiterer Effekt ist die sogenannte Beugung von elektromagnetischen Wellen, welche die Ablenkung der Wellen an Kanten oder Hindernissen beschreibt.

2.6.1.4 Mehrwegeausbreitung

Da in der realen Welt immer eine Kombination aus mehreren dieser Effekten vorkommt, ist es schwierig, die genaue Ausbreitung eines Signals vorherzusagen. Zusätzlich kommt es durch die Kombination der Effekte zu einer Mehrwegeausbreitung des Signals, was dazu führt, dass es beim Empfänger zu einer Laufzeitdispersion des ausgesendeten Signals kommt. Das heißt, dass dasselbe Signal mehrfach und auf unterschiedlichen Wegen (DLOS oder NLOS) beim Empfänger ankommt. Zudem gibt es eine starke zeitliche Variation des ankommenden Signals. Schiller [Sch03] nennt für GSM eine Laufzeitdispersion in Innenstädten von ca. 3μ sec. Die Unterschiede können aber durchaus noch höher sein. Der GSM Standard kann mit einer Verzögerung von bis zu 12μ sec umgehen. Es gibt mehrere Möglichkeiten, diesem Problem zu begegnen. Diese werden aber in dieser Arbeit nicht weiter besprochen.

2.6.1.5 Langsames und schnelles Fading

Weitere Charakteristika des Übertragungskanals von Funkwellen werden durch langsames (short-term) und schnelles (long-term) Fading beschrieben. Diese Effekte treten vor allem dann auf,

wenn sich ein ME während der Übertragung bewegt und es zu einem Signalschwund kommt. Schnelles Fading bezeichnet den kurzzeitigen Schwund der Signalstärke, während langsames Fading den Signalschwund über eine längere Zeitperiode beschreibt. Die Auswirkungen dieser beiden Effekte auf ein ILS werden in Kapitel 4 genauer besprochen. Im Kontext eines ILS könnte langsames Fading zum Beispiel als eine Funktion der Zeit, der Temperatur oder durch den „Blocking-Effekt“des menschlichen Körpers in WLAN Netzen auftreten. Schnelles Fading hingegen kann durch schnelle Bewegungen eines MEs auftreten. Die Mehrwegeausbreitung hat jedoch auch den positiven Effekt, dass eine komplexe Umgebung, wie sie in geschlossenen Räumen gegeben ist, dazu führt, dass jeder Punkt im Raum eine Art eigenen Fingerabdruck als Funktion seiner Signalstärken aufweist. Zusätzlich sind diese Punkte im Normalfall, wenn überhaupt, nur durch langsames Fading charakterisiert [LBR+05].

3 Lokalisierungssysteme

Dieses Kapitel gibt eine umfassende Einführung in die Thematik der Lokalisierungssysteme. Der erste Teil befasst sich mit den verschiedenen Technologien, die zur Lokalisierung in einem ILS eingesetzt werden können, und zeigt auf, welche ILS mit den jeweiligen Technologien bereits realisiert worden sind. Der zweite Teil gibt einen Überblick über die Techniken, die zur Indoor-Lokalisierung eingesetzt werden können. Der letzte Teil diese Kapitels enthält detaillierte Ausführungen zu Lokalisierungsverfahren und deren Umsetzung innerhalb eines ILS.

3.1 Technologien

Während sich für die Lokalisierung außerhalb geschlossener Räume das GPS als der De-Facto-Standard durchgesetzt hat, gibt es bei der Umsetzung eines Lokalisierungssystems innerhalb geschlossener Räume eine Reihe von Technologien, die zur Lokalisierung eingesetzt werden können. Bis zum jetzigen Zeitpunkt konnte sich keine dieser Technologien wirklich durchsetzen. So existieren diverse Systeme, die alle mit unterschiedlichen Technologien implementiert wurden. Die meisten dieser Systeme decken einen klar definierten Anwendungsfall ab und unterscheiden sich teilweise signifikant voneinander. Der folgende Teil soll deshalb einen möglichst vollständigen Überblick der zur Indoor-Lokalisierung bereits verwendeten Technologien geben und immer den Bezug zu in der Literatur erwähnten ILSs herstellen. Die Darstellung erfolgt anhand einer Kategorisierung nach Technologien. Die Reihenfolge der ILSs innerhalb der Kategorien erfolgt chronologisch.

3.1.1 Infrarot

Sender, die Daten über eine Infrarotschnittstelle (IrDA) [Ass] übertragen, oder einfache Infrarotfernbedienungen arbeiten im nahen Infrarot bei einer Wellenlänge von 880 bis 950nm. Ein Sender strahlt Infrarotwellen aus, welche bei einem Empfänger durch einen optischen Sensor aufgefangen werden. Wellen im Infrarotbereich werden dabei von Wänden vollständig reflektiert. Wenn Sender sich in Hosentaschen oder Schubladen befinden oder wenn die Sicht durch andere Gegenstände blockiert wird, ist die Empfangsqualität deutlich vermindert. Die Reichweite eines Infrarotsenders beträgt in etwa 10 bis 20 Metern.

Bei einem Lokalisierungssystem auf Basis von Infrarot muss der Benutzer einen Infrarotsender (Tag) bei sich tragen. Fest installierte Empfänger im Gebäude leiten alle Signale der Sender an einen Server weiter, welcher aus den Signalen und den Positionen der Empfänger eine Position des Senders errechnet. Ein System, welches ein solches Verfahren einsetzt, ist das Active Badge System.

3.1.1.1 Active Badge System

Das Active Badge System [WHFG92] ist das erste Indoor-Lokalisierungssystem überhaupt. Es wurde im Jahre 1992 von dem nordamerikanischen Telekommunikationskonzern AT&T

entwickelt. Das vorrangige Ziel des Systems war die Lokalisierung von Mitarbeitern in den Büroräumen von AT&T. Alle Angestellten mussten sogenannte „Badges“ bei sich tragen. Badges waren kleine Infrarotsender, welche periodisch Signale an die im Gebäude verteilen Infrarotempfänger schickten. Die Empfänger leiteten die empfangenen Signale dann an einen zentralen Lokalisierungserver weiter, welcher die Positionen aller Mitarbeiter verwaltete. Das System lieferte schon eine angemessene Genauigkeit, hatte jedoch Probleme, wenn keine Direct Line Of Sight (DLOS) vorhanden oder der Sender/Empfänger durch direkte Sonneneinstrahlung blockiert wurde.

3.2 Ultraschall

Ultraschall sind Schallwellen im Frequenzbereich von $16\,kHz$ und $1,6\,GHz$. Sie können vom Menschen nicht wahrgenommen werden. Ultraschallwellen werden von Wänden oder anderen Hindernissen (je nach Material) reflektiert oder absorbiert. Wie bei allen Wellen können dabei Brechung, Beugung und Interferenzen auftreten. Ultraschall findet schon lange in Echoloten und Sonaren Anwendung. Sonare benutzen dabei das Echoprinzip wie eine Radaranlage. Sie strahlen aktiv ein Signal aus und empfangen dessen Echo. Dann bestimmen Sie über die Laufzeit des Echos die Entfernung zu dem Objekt, welches das Echo erzeugt beziehungsweise die Schallwellen reflektiert hat. Ein anderes Verfahren lässt Sonare nur die (natürlichen) Signale bzw. Geräusche von Objekten passiv abhorchen. Zudem kann jedes Sonar auch die Richtung des einfallenden Schalls bestimmen.

Diese Verfahren lassen sich auch in Lokalisierungssystemen anwenden. Systeme, welche auf Ultraschall basieren sind zum Beispiel Cricket und das Active Bat System von AT&T. Diese und andere Systeme werden übersichtlich von Zhou et al. [ZYN08] und Hightower et al. [HBB02] beschrieben und verglichen.

3.2.1 Cricket indoor location support system

Das Cricket Location Support System [PCB00] benutzt Ultraschallsender zum Aufbau der Infrastruktur und rüstet Objekte, welche lokalisiert werden sollen, mit Empfängern aus. Diese können dann per Triangulation ihre Position selbst bestimmen. Die Sender müssen dabei nicht an fixen bekannten Positionen angebracht werden. Cricket nutzt die gesendeten Signale nicht nur zur Synchronisation der Zeitmessung, sondern auch um den Zeitpunkt zu beschreiben, an dem der Empfänger die empfangenen Signale berücksichtigen soll. Zudem verbreiten die Sender über Kurzwellenfunk semantische Informationen über den aktuellen Aufenthaltsort.

Die Positionsbestimmung wird entweder über Lateration- oder Proximity-Techniken vorgenommen. Der Empfang von Signalen mehrerer Sender ermöglicht den Empfängern die Triangulation ihrer Position. Wird nur ein einzelnes Signal empfangen, können zusätzlich nützliche Informationen aus der räumlichen Nähe dieses Signals und den semantischen Informationen der Kurzwellensignale gewonnen werden. Die Genauigkeit des Systems liegt bei ca. $1,2\,m$ (4 Fuß). Die Vorteile von Cricket sind der Schutz der Privatsphäre durch die dezentrale Positionsbestimmung, niedere Kosten der Sender und Empfänger sowie eine gute Skalierbarkeit des Systems, da durch eine Erhöhung der zu lokalisierenden Empfänger keine größere Rechenkraft benötigt wird. Nachteile sind das Fehlen einer zentralen Verwaltung und die Tatsache, dass die mobilen Empfangsgeräte viel Rechenkraft zur Positionsbestimmung benötigen.

3.2.2 Active Bat System

Das Active Bat Location System von AT&T aus dem Jahr 1999 [Cam] rüstet Benutzer und Objekte mit sogenannten Active Bat Tags aus. Jedes Bat hat eine eindeutige GUID zur Adressierung und Zuordnung. Bats senden als Antwort auf eine Anfrage, welche über Kurzwellenfunk von einem Controller versendet wird, einen Ultraschallimpuls an ein an der Decke angebrachtes Raster aus Empfängern. Der Controller sendet neben der Anfrage gleichzeitig ein Reset-Signal über ein drahtgebundenes serielles Netzwerk an die Empfänger an der Decke. Diese messen alle das zeitliche Intervall zwischen dem Reset und der Ankunft des Ultraschallimpulses und berechnen daraus jeweils ihre Entfernung zum Bat. Diese Abstandsmessungen werden an einen zentralen Server geschickt, welcher per Lateration die Positionsbestimmung durchführt. Die Genauigkeit liegt dabei bei ca. $9\,cm$. Benötigt wird dafür ein großes und präzise ausgerichtetes Sensornetz an der Decke, was die Skalierbarkeit einschränkt und hohe Kosten verursacht. Eine gute Beschreibung des Active Bat Systems findet man außerdem in [HHS+99], und Ward beschreibt das System auch ausführlich in seiner PhD [War99].

3.3 Funkwellen

3.3.1 WLAN

Unter Wireless Local Area Network (WLAN) [CWKS02] versteht man ein lokales Funknetz nach dem IEEE 802.11 Standard. Diese Netzwerke senden im $2,4\,GHz$ Band und im $5GHz$ Band. Die Reichweite eines Senders beträgt zwischen 50 und 100 Metern und ist stark von Hindernissen, der Art und Form der Gänge und Zimmer sowie der Einrichtung abhängig. Zudem liegt das Frequenzspektrum von $2,4\,GHz$ genau im Bereich der Radiowellen, welche von Wasser absorbiert werden, somit dämpft der menschliche Körper die Empfangsstärke. Da WLAN sich inzwischen als weit verbreiteter Standard für den Aufbau lokaler Netze etabliert hat, liegt es nahe, diese bereits vorhandenen Strukturen auch für Lokalisierungssysteme zu benutzen.

In der Regel werden WLANs im Infrastrukturmodus betrieben. Hierbei übernimmt ein AP die Koordination aller anderen Knoten im Netzwerk. Dazu sendet er kleine Datenpakete, sogenannte Beacons, in kleinen Intervallen von ca. 10 Paketen pro Sekunde ($10\,Hz$), an alle Knoten im Sendebereich. Diese Beacons kann jeder Knoten (passiv) lesen und verarbeiten, auch ohne mit dem AP verbunden zu sein. Die Beacons enthalten unter anderem den Netzwerknamen (SSID) und vor allem die Feldstärke des Signals. Unter der Annahme, dass diese Feldstärken an bestimmten Punkten im Raum stets gleich bleibend sind, kann man durch einen Abgleich der gespeicherten Feldstärken und der aktuell gemessenen bis zu einem gewissen Grad auf die aktuelle Position im Raum schließen.

Eine andere Herangehensweise zur Lokalisierung ist, gezielt Pakete auszutauschen, wenn ein ME (aktiv) zu einem AP verbunden ist, um die Zeit zwischen dem Absenden eines Paketes und dem Eintreffen der Antwort zu messen. Macht man dies mit mehr als 2 APs, kann man über das Laterationsverfahren die Position errechnen.

Die Genauigkeit der Positionsbestimmung mit Hilfe von WLAN liegt je nach System bei ca. 0,5-2 Metern. WLAN wird unter anderem von den Systemen RADAR [BP00], 3D-iD (PinPoint) [WL98], SpotOn [Jef], Nibble [DoCSUtUoC], Ekahau [Ekab], Herecast [Pac04], Placelab [BCLN05], WLocater [PKL07], HORUS [YASN02, YAS03, YA03, YA05] und Xiang

[XSC+04] eingesetzt.

3.3.1.1 RADAR

RADAR, entwickelt von der Microsoft Research Group von Bahl et al. [BP00], ist das erste auf Signalstärke basierte WLAN ILS und Grundlage für viele andere Systeme. Dabei werden sowohl empirische als auch theoretische Modelle zur Lokalisierung verwendet. RADAR misst an den APs die Signalstärke und das Signal-Rausch-Verhältnis der Signale, welche mobile Endgeräte senden. Dafür werden anstatt normaler APs PCs mit einem WLAN Adapter, FreeBSD und der Software HostAP betrieben. Dies ermöglicht, die Broadcast Signale aller Clients in der Umgebung zu empfangen. Die Daten werden zur Berechnung der 2D-Position innerhalb eines Gebäudes benutzt. Bahl et al. haben dafür zwei unterschiedliche Ansätze implementiert: eine empirische Methode, basierend auf Szenenanalyse, und eine andere, basierend auf Lateration anhand eines Radio Propagation Modells.

Für den ersten Ansatz werden initial bei der Installation des Systems in einer Datenerfassungsphase von mehreren Positionen im Gebäude Referenzmessungen in einer Datenbank gespeichert. In der Onlinephase ermittelt ein Pattern-Matching-Algorithmus mit Hilfe der EuklidischenDistanz die Position des MEs.[1] Dies gelingt mit einer Genauigkeit von ca. 3 Metern, bei einer Wahrscheinlichkeit von 50 %.

Der Laterationsansatz bestimmt die Position des MEs anhand eines Modells. Dieses berechnet die Abstände des MEs unter Zuhilfenahme der Vorhersage der Signalausbreitung. Je besser dieses Modell und die Vorhersage der Signalausbreitung ist, umso genauer ist später die Lokalisierung. Dieser Ansatz führt zu einer Genauigkeit von ca. 4,3 Metern bei ebenfalls 50 %.

Der große Vorteil von RADAR ist, dass ohne eine Anwendung auf dem ME zu installieren, dieses lokalisiert werden kann. Die verwendeten APs können dabei als ganz normale APs zum Aufbau eines WLANs mitbenutzt werden. Ein Nachteil ist jedoch, dass RADAR nicht mit gängigen APs realisiert werden kann. Außerdem bietet RADAR keine Lösung für stockwerkübergreifende Lokalisierung. Es existiert eine Erweiterung zu RADAR, nachzulesen in [BBP00].

3.3.1.2 3D-iD (PinPoint) Location System

Pinpoints 3D-iD [WL98] ist ein proprietäres System, welches dem Microsoft RADAR System ähnelt. Personen (oder Dinge) tragen permanent Tags bei sich und ein zentraler Server ermittelt von allen Tags die Position über spezielle Pinpoint-Basistationen. Die Basisstationen senden periodisch ein Broadcastsignal aus. Tags, welche einen Broadcast empfangen, senden ein Signal auf ihrer eigenen eindeutigen Frequenz moduliert mit einer ID zurück. Die Basisstation ermittelt aufgrund der Laufzeit (ToF) des Signals die Entfernung des Tags zu den verschiedenen Antennen. Per Lateration kann 3D-iD daraus die Position mit einer Genauigkeit von ca. 3 Metern errechnen.

[1] Die von Bahl verwendeten Begriffe Onlinephase und der Pattern-Matching-Algorithmus werden im Rahmen dieser Arbeit als Realtimephase und LFPT-Verfahren bezeichnet.

3.3.1.3 SpotOn System

Das SpotOn System [Jef] implementiert eine Ad-hoc-Lateration mit Low-Cost-Tags. SpotOn Tags messen die Signaldämpfung, um den Abstand zwischen 2 Tags zu schätzen. Das Ziel in SpotOn ist es, MEs in Relation zueinander zu lokalisieren, anstatt zu festen Basisstationen. Die Tags werten dabei die Dichte der anderen Tags und die Korrelation von mehreren Messungen aus. Standorte von Objekten ohne feste Infrastruktur zu bestimmen, stellt einen hoch skalierbaren und kostengünstigen Ansatz dar.

3.3.1.4 Ekahau

Das Ekahau Real Time Location System [Ekab] ist ein kommerzielles, WLAN basiertes Lokalisierungssystem, welches mit Nebenräumen, Räumen und Stockwerken arbeiten kann. Das System verwendet patentierte softwarebasierte Algorithmen, um den Aufenthaltsort von Objekten zu berechnen, und kann problemlos bis Zehntausende von Tags pro Server skalieren. Das Ekahau-System arbeitet mit bestehender WLAN-Infrastruktur und bietet die branchenweit genaueste und kostengünstigste Standortbestimmung. Mehrere internationale Unternehmen wie HP, 3M, McKesson, Nortel, Siemens und andere setzen Ekahau als die beste Kombination von Technologie, Leistung, Kosten und Produktreife ein. Neben dem Lokalisierungssystem vertreibt Ekahau ein Tool namens HeatMapper [Ekaa]. Diese kostenlose Software dient zur schnellen und einfachen Erfassung und Kartographierung von 802.11 Netzwerken. Es zeigt auf einer Karte die Netzabdeckung zu Hause oder im Büro an und lokalisiert alle APs. HeatMapper verwendet dabei einen normalen WLAN-Adapter. Alles, was man braucht, ist ein windowsbasierter Laptop.

3.3.1.5 Sonstige Systeme

Im Bereich der Lokalisierung mittels WLAN-Technologien gibt es noch viele weitere Systeme, unter anderem Nibble [CCKM01]. Dies ist das erste System, welches mit einem Wahrscheinlichkeitsmodell zur Positionsbestimmung gearbeitet hat. Nibble nutzt Bayesian networks, um den Standort eines Geräts abzuleiten.

Placelab [BCLN05] und Herecast [Pac04] ermöglichen es WLAN-fähigen MEs, automatisch ihre Position zu bestimmen, indem sie Signale von bekannten 802.11 APs abhören. Placelab speichert die MAC-Adresse, sowie Längen- und Breitengrade der einzelnen APs auf den MEs ab. Diese berechnen dann den Durchschnitt der Längen- und Breitengrade für alle APs, welche sie empfangen. Herecast speichert die APs und einen symbolischen Namen für den Standort des APs. Der Standort des APs mit der höchsten Signalstärke wird als der Standort des ME angegeben. APs mit sehr schwachen Signalstärken haben bei Placelab also keine Auswirkungen auf die Bestimmung der Position, es sei denn, man empfängt ausschließlich sehr schwache Signale. Schwankende und schwache Signale in Herecast verändern die geschätzte Position hingegen signifikant.

Das WLocater System von Philips et al. [PKL07] beschreibt die Theorie und Implementierung eines WLAN und Bluetooth basierten ILS, welches mit Hilfe eines eigenen Algorithmus versucht, Fluktuationen und den Ausfall von Infrastrukturkomponenten zu kompensieren.

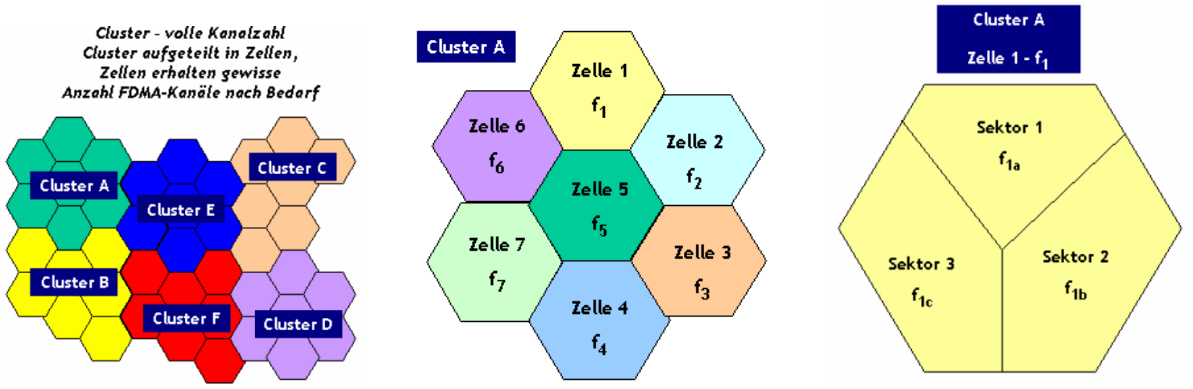
Hemmersdorf et al. [Her06] entwickelte am Nokia Research Center in Helsinki ein grobes Lokalisierungsystem mit der Granularität von Räumen/Fluren, basierend auf Feldstärken be-

stimmter APs, deren Position bekannt ist.

Ein weiterer Ansatz, der nicht auf eine Szenenanalyse (LFPT), der Abstandsbestimmung anhand der Signalausbreitung, sondern auf der Ableitung von Entfernungen zu bekannten APs durch Messung der Laufzeit setzt, ist das am Wilhelm-Schickard-Institut für Informatik der Eberhard-Karls-Universität Tübingen in der Forschungsgruppe für interaktive Kommunikationssysteme entwickelte Goodtry [Ambb, HW08] System, welches eine Genauigkeit im Zentimeterbereich liefert. Außerdem existieren noch zwei auf Wahrscheinlichkeitsmodellen basierende ILS: Das HORUS System [YASN02, YAS03, YA03, YA05] und das Xiang System [XSC+04]. Beide messen die Feldstärken immer auf Seite der MEs und führen zudem die Lokalisierung direkt auf dem ME durch.

Einen allgemeinen Überblick über WLAN basierte Lokalisierungssysteme bieten außerdem [PKL07], [HBB02] und [LBR+05], sowie die grundlegenden Untersuchungen von Hossain et al. [M. 07].

3.3.2 GSM



(a) Mehrere GSM Cluster, jeweils aus 7 Zellen. Die Cluster haben alle denselben Frequenzbereich (Channels)

(b) Ein GSM Cluster bestehend aus 7 GSM Zellen. Jede GSM Zelle hat ihren eigenen Frequenzbereich (Channel)

(c) Eine GSM Zelle mit 3 Sektoren, jeder Sektor sendet auf einem Teil der Channels der Zelle

Abbildung 3.1: Aufbau eines GSM Netzes in einer Wabenstruktur nach [Ste]

Das Global System for Mobile Communications (GSM), der Mobilfunkstandard der zweiten Generation, ist in Zellen organisiert. Diese Zellen werden von Basisstationen (BTS, Base Transceiver Station) aufgebaut und haben eine eindeutige Identifikationsnummer (CID). Auf eine Basistation fallen in der Regel 3 Zellen. Je nach Gelände und Bevölkerungsdichte dehnen sich GSM Zellen von einigen 100 Metern bis zu 35 Kilometern aus. Die verwendeten Frequenzen liegen in Europa entweder im $900\,MHz$ Band oder aber im $1800\,MHz$ Band. GSM ist heute sowohl in Städten, Vororten, Dörfern und auch im Freien flächendeckend verfügbar.

Die Architektur von GSM umfasst neben der BTS noch weitere Infrastrukturkomponenten. Die Architektur ist in Abbildung 3.2 dargestellt. Sie verfügt über drei Subsysteme: dem Radio Station Subsystem (RSS), dem Network Switching Subsystem (NSS) und dem Operation Subsystem (OSS). Das Radio Station Subsystem enthält das eigentliche zellulare Netz und ist die

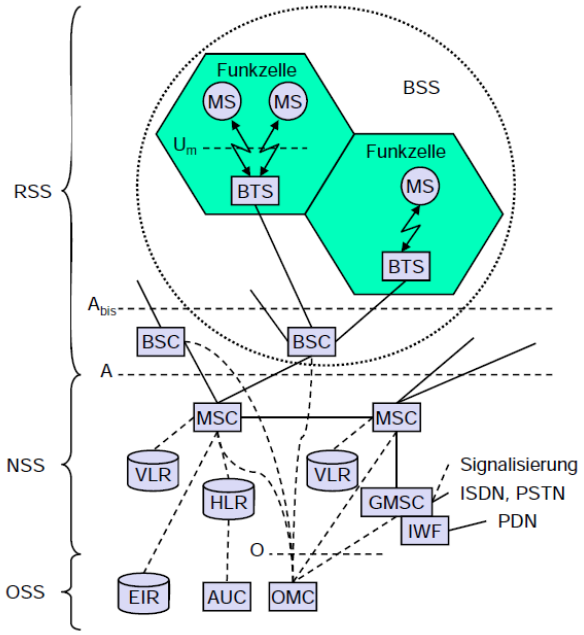


Abbildung 3.2: GSM Architektur mit allen drei Subsystem: RSS, NSS und OSS [Sch03].

Schnittstelle hin zu dem Mobilvermittlungsnetz NSS. Es beinhaltet das Base Station Subsystem (BSS), welches wiederum eine BTS und einen Base Station Controller (BSC) enthält. Der BSC ist eine der BTS übergeordnete Komponente, welche die Steuerung mehrerer BTS übernimmt. Die Funktionsweise einer BTS wird wegen seiner Relevanz zur Lokalisierung genauer vorgestellt. Für Details zum NSS und OSS sei auf Schiller et al. [Sch03] verwiesen.

Eine GSM Basisstation (BTS) besteht aus mehreren Zellen, aufgeteilt in mehrere Sektoren, und ist in Abbildung 3.1 dargestellt. Jeder dieser Zellen wird eine ID und ein Kanal (Channel), ein bestimmtes Frequenzspektrum, zugeordnet. Benachbarte Zellen werden nie demselben Kanal (Channel) zugeordnet, um Interferenzen zu vermeiden. Da die Mobilfunkanbieter aber nur über ein begrenztes Frequenzspektrum verfügen, müssen die Kanäle in weiter entfernt liegenden Zellen wiederverwendet werden. Um Kanäle möglichst häufig wiederverwenden zu können und gleichzeitig Interzell-Interferenzen zu minimieren, muss eine Vielzahl von Messungen vorgenommen und mit komplexen Berechnungen die Zuordnung der Kanäle realisiert werden. Die auf diese Weise errichtete Zell-Infrastruktur wird deshalb einmal eingerichtet und äußerst selten verändert.

Ein ME ist ist immer zu der Zelle mit dem stärksten Signal verbunden und verfügt zusätzlich über einen speziellen Broadcast Control Channel (BCCH). Dieser wird dazu genutzt, die Zell-IDs der benachbarten Zellen, den Neighbouring Cells, zu übertragen. Das ME ist somit durch Abgleich der Feldstärken der benachbarten Zellen in der Lage, periodisch zu überprüfen, ob es noch zu der stärksten Zelle verbunden ist oder ob in eine neue Zelle gewechselt werden muss. Damit dies zuverlässig funktioniert, findet die Übertragung im BCCH ohne Transmission-Power-Control statt. Das heißt, die Daten im BCCH Kanal werden immer mit voller Leistung gesendet. Das ME misst hierzu die Feldstärken des von der BTS empfangenen Signals und übermittelt diesen Wert in einem Measurement Report zurück an die BTS. Der BSC, als übergeordnetes Netzelement, entscheidet dann anhand den vorgegebenen Schwellenwerten, ob

es die Sendeleistung verringern, erhöhen, beibehalten oder einen Zellwechsel vornehmen soll. Nähere Information zum Zellwechsel finden sich in Abschnitt 3.5.4.1 bei der Beschreibung des Cell of Origin Verfahrens.

Da GSM aufgrund seines lizenzierten Frequenzbereiches keine Probleme mit Interferenzen anderer Technologien hat, besitzt GSM eine sehr stabile und konstante Wellenausbreitung und man kann mit Kenntnis der genauen Positionen der BTS, der aktuellen Zell-IDs und der Feldstärke über Trilateration eine Positionsbestimmung durchführen. Die Stabilität des GSM Netzes kommt auch daher, dass Mobilfunkprovider selten Änderungen an ihrer Infrastruktur vornehmen.[2] Daneben sind Ansätze der Szenenanalyse – genau wie schon zu WLAN beschrieben – realisierbar und finden auch in dieser Arbeit Anwendung.

3.3.2.1 GSM Systeme

Otsason et al. [OVLL05] beschreiben das erste etwas genauere GSM basierte ILS. Es werden nicht nur die aktuelle Zelle und die direkt benachbarten Zellen mit einbezogen, sondern auf Wide-Signal-Strength-Fingerprints gesetzt. Dabei werden die Feldstärken der Zellen mit in den Fingerprint aufgenommen, die an der aktuellen Position zwar noch messbar sind, zu denen aber nicht mehr verbunden werden kann. So werden Daten von den 6 stärksten GSM Zellen und von bis zu 29 weiteren Zelle berücksichtigt. Die zusätzlichen Zellen erhöhen die Genauigkeit um bis zu 50%, welche dann zwischen ca. $2,5m$ und $5,5$ Metern liegt. Dabei ist insbesondere die Erkennung von Stockwerken mit einer Zuverlässigkeit von meist über 90 % möglich.

Shyy et al. [SR00] beschreiben ein Lokalisierungssystem auf Basis von 2G- und 3G-Netzwerken. Der Ansatz basiert darauf, eigene GSM BTSs in geschlossenen Räumen zu installieren, um die Position eines Handset zu bestimmen. Dabei wird auf eine einfache Szenenanalyse durch LFPT gesetzt. Es wird dafür zunächst die Feldstärke aufgenommen. Des weiteren wird der Antennensektor der GSM BTS bestimmt, indem sich das ME befindet, um die Richtung zu ermitteln. Zudem wird die Genauigkeit der Positionsbestimmung durch ein Modell zur Einbeziehung der Multipath-Propagation signifikant erhöht.

3.3.2.2 WLAN und GSM kombinieren

Zhou et al. [ZYN08] beschreiben zunächst einen GSM basierte Ansatz. Sie Stellen aber auch fest, dass sich in Kombination mit WLAN ganz neue Möglichkeiten ergeben und zeigen ein System zur Ortung in Gebäuden auf, welches GSM- und WLAN-Signale zur Standortschätzung heranzieht, so dass das daraus resultierende System genauer und stabiler wird.

3.3.3 Bluetooth

Bluetooth ist der drahtlose Netzwerkstandard nach IEEE 802.15 mit einer Reichweite je nach Klasse von bis zu 1, 10 oder 100 Metern. Gesendet wird im $2,4\,GHz$ Band und jedes Bluetooth-fähige Gerät hat eine eindeutige ID (die MAC-Adresse), worüber es identifiziert werden kann. Bluetooth kann also für Lokalisierungssysteme mit denselben Ansätzen, wie schon bei WLAN und GSM erwähnt, benutzt werden. Allerdings gibt es keine weit verbreiteten Bluetoothnetze, wie es bei WLAN und GSM der Fall ist. Allerdings ist Bluetooth heutzutage in fast allen MEs

[2] Die durchgeführten Messungen, zu dieser Arbeit haben aber gezeigt das es durchaus vorkommen kann das ein Mobilfunkanbieter die Zell ID's in periodischen Abständen ändert.

mit verbaut. Darüber hinaus gibt es kleine kostengünstige Bluetooth-Tags, die in jede Hosentasche passen. Es müssten also nur genügend Bluetooth-Tags (Landmarken) als äquivalente zu den BTS bei GSM oder den APs bei WLAN in dem Gebäude installiert werden, in welchem man ein Bluetooth basiertes Lokalisierungssystem errichten möchte.

Bei einem Szenenanalyse basierten Bluetooth-Verfahren muss nicht unbedingt die RSSI gemessen und zur Positionsbestimmung herangezogen werden. Oft reicht in solchen Installationen die Verwendung eines räumlichen Annäherungsverfahrens aus, welches den primitiven Ansatz verfolgt, die MAC-Adressen der Landmarken zu vermerken und so festzustellen, ob ein Bluetooth-Knoten mit einer Reichweite von 1 bis 10 Metern empfangen werden kann oder nicht [M. 07].

Es existieren zwar keine vergleichbaren Bluetooth basierten Systeme, wie es sie bei WLAN und GSM gibt, aber in ihren allgemeinen Untersuchungen zu Szenenanalyse basierten ILS beschreiben Hossain et al. [M. 07] neben WLAN auch einen Ansatz mit Bluetooth. Des weiteren beschreiben Feldmann et al. [FKZL03] ein Bluetooth basiertes ILS, welches auf Signalstärkemessungen durch die näherungsweise erhaltenen RSS auf einem ME setzt. Die Positionierung erfolgt über eine Distanzmessung und anschließender Lateration. Die Abhängigkeit zwischen den erhalten RSSI und der Entfernung wird durch eine passende Polynomapproximation abgebildet. Die Genauigkeit hierbei liegt bei ca. $2,8$ Metern. Ein sehr innovatives Projekt wird am Wilhelm-Schickard-Institut für Informatik an der Eberhard-Karls-Universität Tübingen, am Lehrstuhl der Technischen Informatik, im Rahmen des Ambisense Projektes [Amba] durchgeführt. Anhand eines Modells zur Signalausbreitung von Bluetooth Landmarken wurde hier eine Partikel-Filter-Implementierung umgesetzt, welche ein Bluetooth basiertes Lokalisierungssystem [FBSR08, SSZ+09] mit einer sehr hohen Genauigkeit im Zentimeterbereich hervorgebracht hat.

3.3.4 Optisch

Kamera

ME haben oft eine eingebaute Kamera. Auch diese kann als Grundlage für ein Lokalisierungsverfahren dienen. Dabei gibt es Systeme, die versuchen, anhand eines Szenenanalyseansatzes, von in einer Datenbank hinterlegten Bildern, durch Abgleich selbiger die aktuelle Position eines ME zu bestimmen [SHTH07]. Das Gebäude ist also in Form von Fotos in einer Datenbank abgebildet worden. Nach Aufnahme eines Bildes mit dem ME wird dieses gegen alle Bilder der Datenbank verglichen. Weitere Verfahren basieren auf der Verwendung von Marker und Augmented Reality (AR) Technologie.

3.3.5 Satelliten-Systeme

Navigationssatellitensysteme bestimmen Aufenthaltsorte durch den Empfang der Signale von Navigationssatelliten. Mehrere Satelliten senden ihre genaue Position und ihre aktuelle Uhrzeit an einen Empfänger. Dort werden dann die Signallaufzeiten errechnet und daraus die aktuelle Position bestimmt.

(a) NAVSTAR GPS Satellit der zweiten Generation [kowb]

(b) GPS Satellitenbahnen (Abstände maßstabsgetreu) [kowb]

Abbildung 3.3: 3.3(a) zeigt einen GPS Satelliten, 3.3(b) die Verteilung der Satelliten im Orbit um die Erde.

3.3.5.1 GPS

Das Global Positioning System (GPS) [Guo07] ist das derzeit bekannteste Lokalisierungssystem. Es setzt bis zu 31 Satelliten (Abbildung 3.3), auf einer kreisförmigen Umlaufbahn in 20.200 km Höhe mit 55° Inklination ein. Die Satelliten senden permanent ihre aktuelle Position und die genaue Uhrzeit. GPS-Empfänger können aus den Signallaufzeiten ihre eigene Position und Geschwindigkeit berechnen. Dazu würden eigentlich die Signale von drei Satelliten ausreichen (einfache Trilateration). Meistens haben GPS-Empfänger aber keine genaue Uhr und können die Signallaufzeiten nicht korrekt messen. Mit dem Signal eines vierten Satelliten kann allerdings die genaue Zeit im Empfänger bestimmt werden.

Die Technik hinter GPS und die damit verbundene Mathematik ist sehr komplex, das System selber ist aber für einen Anwender denkbar einfach einzusetzen. Auch sind GPS-Empfänger heute sehr handlich und kostengünstig. GPS erzielt zu 95 % eine durchschnittliche Genauigkeit von 7,8 Metern. Um sich über den aktuellen Fehler des GPS zu informieren, gibt es einen Webservice, der die Genauigkeit der GPS Position über 24 Stunden [kowb] in Echtzeit darstellt. Abbildung 3.4 zeigt einen Ausschnitt dieser Darstellung. Durch Differenzielles GPS (DGPS) kann die Genauigkeit auf weniger als ein Meter erhöht werden [kowa]. Dazu werden stationäre GPS-Empfänger zur Korrektur der Messung installiert. Diese senden periodisch Signale aus, die von den GPS-Empfänger dazu verwendet werden, den aktuellen Fehler zu korrigieren.

GPS eignet sich für Navigationssysteme in Autos und führt durch die Verfügbarkeit auf immer mehr MEs zu einer wachsenden Anzahl an Diensten, die Nutzern Informationen anhand ihrer aktuellen Position anbieten. So wird bei Fotoaufnahmen der Aufenthaltsort mit erfasst und dem Nutzer, wenn er unterwegs ist, zusätzliche Kontextinformationen bereitgestellt. Weiterhin werden GPS fähige Endgeräte zunehmend im Profi- und Freizeitsport zur Trainingsplanung und -überwachung eingesetzt. Innerhalb von Gebäuden ist GPS jedoch so gut wie nicht verwendbar oder liefert eine unzureichende Genauigkeit.

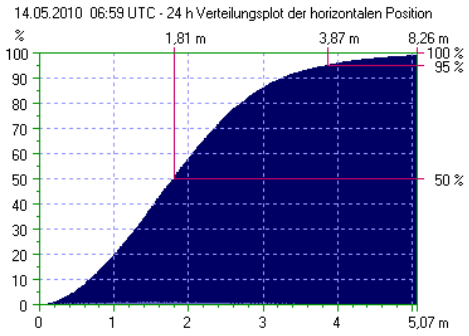


Abbildung 3.4: Genauigkeit der GPS Position über 24 Stunden [kowb]. Die Y-Achse gibt die Genauigkeit in % an, die X-Achse die Abweichung in Metern. Somit sind 50 % aller Positionsbestimmungen der letzten 24 Stunden innerhalb eines Kreises mit dem Radius von 1,81 Meter um den Mittelwert. Der letzte Wert der X-Achse (5,07 Meter) gibt den Umkreis für 99 % aller Messpunkte an.

3.3.5.2 Das WGS 84 Koordinatensystem

In der Geodäsie und Kartografie bezeichnet man die Position, Orientierung und den Maßstab eines zwei- oder dreidimensionalen Koordinatensystems als geodätisches Referenzsystem oder auch geodätisches Datum. Das World Geodetic System 1984 (WGS 84) [wgs04] ist ein solches Referenzsystem und dient als einheitliche Grundlage für Positionsangaben auf der Erde und im erdnahen Weltraum. WGS 84 legt dafür ein dreidimensionales Koordinatensystem in Form eines kartesischen Rechtssystems fest, wobei die x-Achse in Richtung 0° Länge und Breite zeigt, die y-Achse nach 90° Ost und die z-Achse zum Nordpol. GPS verwendet dieses System zum Errechnen der Koordinaten.

3.3.6 Inertial

Inertial Navigation, oder auch Trägheitsnavigation genannt, dient der Ermittlung der Geschwindigkeit und Position eines ME ohne Zuhilfename von Infrastruktur in der Umgebung stets relativ zu ihrer Ausgangsposition. Inertial Navigation kann deshalb denen in Kapitel 2.3 vorgestellten relativen Lokalisierungssystemen zugeordnet werden. Solche Systeme sind in der Regel mit einem elektrischen Kompass zur Richtungsbestimmung, einem Accelerometer (Beschleunigungssensor) und einem Gyroskop (Lagesensor)[3] ausgestattet. Diese Sensoren werden auch als Inertial Measurement Units (IMU) bezeichnet.

Zur Positionsbestimmung müssen die Beschleunigung und die Winkelgeschwindigkeiten der drei Axen im Raum bestimmt werden. Mit der Beschleunigung erhält man durch Integration über die Zeit die Geschwindigkeit und nach nochmaliger Integration die dadurch verursachte Positionsänderung. Dank Integration der gemessenen Winkelgeschwindigkeiten über die Zeit in den drei Raumrichtungen kann man die resultierende Richtung der Bewegung ermitteln. Interessanterweise basiert der Gleichgewichtssinn bei Säugetieren auf demselben Prinzip. IMUs liefern nur für kurze Messperioden verlässliche Werte. Kleinste Messfehler wirken sich durch die 2-fache Integration mit dritter Potenz aus. In der Praxis kombiniert man deswegen

[3] 1817 vom Professor für Astronomie, Mathematik und Physik Johann Gottlieb Friedrich von Bohnenberger an der Universität Tübingen erfunden.

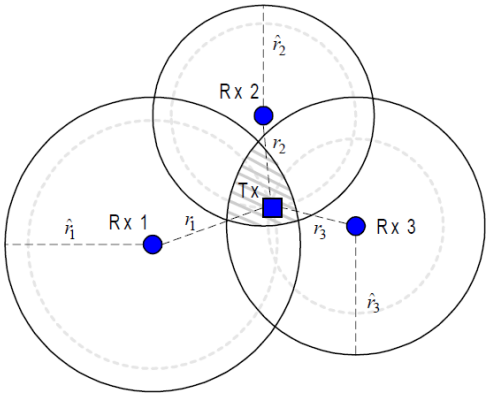


Abbildung 3.5: Time of Flight (TOF) [PLM02]

ein Inertial-Navigation-System oft mit anderen ILS oder Navigationssystemen wie GPS.

3.4 Techniken

Zu der Durchführung einer Indoor-Lokalisierung können verschiedene Techniken verwendet werden. Die eingesetzte Technik ist hierbei oft durch die eingesetzte Technologie oder die verwendete Hardware vorgegeben. Es gibt aber auch Technologien, bei denen mehrere der vorgestellten Verfahren verwendet werden können. Die hier vorgestellten Techniken können in Ihrer Form in jedem Outdoor- oder Indoor-Lokalisierungssystem eingesetzt werden. Der Schwerpunkt bei den folgenden Ausführen liegt aber auf dem Einsatz in einem ILS.

3.4.1 Time of Flight (TOF)

Das TOF Verfahren basiert auf der Messung der Zeit, die ein Signal vom Sender zum Empfänger benötigt. So wird sowohl beim Sender als auch beim Empfänger des Signales eine Zeitmessung vorgenommen und anhand der bekannten Signalausbreitungsgeschwindigkeit auf die Entfernung geschlossen. Die Ausbreitungsgeschwindigkeit von elektromagnetischen Strahlen beträgt in etwa

$\frac{2}{3} * c$ mit Lichtgeschwindigkeit $c = 299792458 \frac{\text{Meter}}{\text{Sekunde}}$ (in Kupferkabeln).

TOF Techniken führen zu einer Herausforderung bei der Verwendung zur Lokalisierung, da diese bei der gegebenen Ausbreitungsgeschwindigkeit elektromagnetischer Strahlen von etwa $300\,m$ pro μs einen Messfehler von etwa $300\,m$ bei nur $1\mu s$ in der Synchronisation der Uhren bedeuten. Um eine möglichst genaue Lokalisierung durchzuführen, ist eine ständige Synchronisation der Uhren bei Sender und Empfänger unabdingbar. Der so berechnete Abstand zwischen Sender und Empfänger lässt aber noch nicht auf die endgültige Position des zu lokalisierenden Objektes schließen. Hierzu muss das Lokalisierungsverfahren der Trilateration verwendet werden. Jeder gemessene Abstand zu einem Sender definiert einen geometrischen Kreis um den Empfänger. Durch Trilateration kann durch das Messen der TOF an drei Empfängern eine Position fixiert werden. Probleme bereiten bei der Verwendung der TOF vor allem die Mehrwegeausbreitung und NLOS Signalwege. Die TOF Technik zählt aber wohl zu einer der genauesten Methoden zur Lokalisierung.

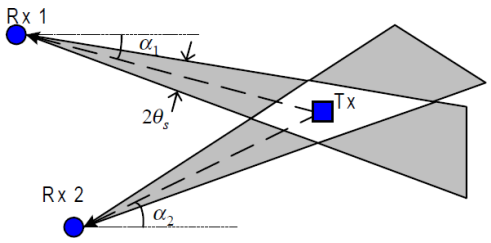


Abbildung 3.6: Angle of Arrival (AOA) [PLM02]

3.4.1.1 DTOA

Differential Time of Arrival (DTOA) basiert auf Messung der Entfernungsdifferenz aller Positionen. Von zwei Sendern wird jeweils der Startzeitpunkt eines Signals gemessen (t_0). Diese Signale werden vom Empfänger nacheinander empfangen (t_1, t_2). Hierbei müssen nur die Uhren der Sender, nicht die des Empfängers synchronisiert werden. Für eine elektromagnetische Welle ist die Entfernungsdifferenz gegeben durch:

$$\Delta = c|t_2 - t_1|$$

3.4.1.2 Received Signal Strength (RSS)

Ein weiteres Verfahren zur Lokalisierung besteht darin, die Feldstärke des empfangenen Signals zu messen und daraus Rückschlüsse auf die Position der ME zu ziehen. Die Feldstärke (Signal Strength) wird in *dBm* gemessen, wobei eine Feldstärke von p Watt in etwa $10 \cdot \log_{10}(\frac{s}{0.001})\ dBm$ entspricht [BP00]. Die Signal to Noise Ratio (SNR) wird in dB angegeben [BP00]. Eine Signalstärke s in Watt und Noise Power in n Watt führen zu einer SNR von $10 \cdot \log_{10} \frac{s}{n} dB$

Zur Abstandsbestimmung anhand von Feldstärken gibt es zwei verschiedene Ansätze. Eine allgemeine Feldstärkefunktion zur Abstandsbestimmung ist gegeben durch:

$$db = 92{,}4 + 20 \cdot \log_{10}(r) + 20 \cdot \log_{10}(f)$$

Mit Abstand r und Frequenz f. Mit Hilfe dieser Funktion kann beim Empfänger über ein mathematisches Modell zur Signalausbreitung (anhand der Dämpfung des Signals mit der Entfernung) die Distanz zwischen Sender und Empfänger berechnet werden. Die Lokalisierung kann mittels Lateration oder tabellenbasiertem Verfahren (Location Fingerprinting) erfolgen.

3.4.2 Angle of Arrival (AOA)

AOA ist ein Verfahren zur Lokalisierung eines Senders, welches die Signaleinfallswinkel an mehreren Basisstationen dazu nutzt, die Position eines Senders zu bestimmen. Damit das Verfahren angewendet werden kann, müssen mindestens zwei unabhängige Basisstationen den Signaleinfallswinkel bestimmen, zusätzlich muss der Standort der beiden Basisstationen bekannt sein. Um die Signaleinfallswinkel zu bestimmen, kommen sogenannte Antennen Arrays zum Einsatz. Diese bestehen aus mehreren gerichteten Antennen, die jeweils einen eigenen Öffnungswinkel haben und es ermöglichen, die Richtung des ankommenden Signals zu bestimmen. Dabei müssen Sender und Empfänger LOS sein. Abbildung 3.6 gibt einen Überblick über die Funktionsweise der Technik. Die Genauigkeit des Verfahrens hängt stark von den

Multipath-Effekten ab [SR00], welche dazu führen, dass die Signale in einem anderen Winkel bei der Basisstation ankommen.

3.5 Lokalisierungsverfahren

3.5.1 Triangulation

Die Technik der Triangulation ist ein rein mathematischer Lösungsansatz, um eine Position zu bestimmen. Hierbei nutzt man die geometrischen Eigenschaften eines Dreiecks, um Abstände und damit Positionen von Objekten zu berechnen [DZ02]. Dabei unterscheidet man zwei grundsätzliche Herangehensweisen, das Dreieck zur Positionsbestimmung aufzuspannen: Angulation, basierend auf Winkelmessung, und Lateration, beruhend auf Entfernungs- bzw. Abstandsmessungen.

3.5.1.1 Lateration

Wie in Kapitel 3.4.1 beschrieben, kann der Abstand einer ME zu einem AP anhand der Signallaufzeit bestimmt werden. Ist der Abstand eines Objektes zu einem Punkt in der Ebene bekannt, dann befindet sich das Objekt auf einem Kreis um den Punkt herum, wie es Abbildung 3.5 zeigt. Im dreidimensionalen Raum wäre dies auf einer Kugel. Kennt man den Abstand eines Objektes zu zwei Punkten im Raum, dann befindet sich das Objekt auf den Schnittpunkten der beiden Kugeln, was einen Kreis ergibt. Mit der Kenntnis einer dritten Entfernung zu einem Punkt wird eine eindeutige Bestimmung der Position des Objektes möglich, da drei sich schneidende Kugeln maximal zwei gegenüberliegende Schnittpunkte haben. In der Regel lässt sich einer davon dann durch Plausibilitätsüberlegungen ausschließen.

3.5.1.2 Angulation

Angulation [HB01] ist ein winkelbasiertes Lokalisierungssverfahren, ähnlich der Lateration, allerdings mit dem Unterschied, dass – anstatt Abstände und Signallaufzeiten – in diesem Verfahren Winkel, Signalstärken und spezielle Antennen benutzt werden. Wie zum Beispiel mit Hilfe von AOA. Es werden also die Winkel zwischen dem zu lokalisierenden Objekt und zweier bekannter Referenzpunkte bestimmt. Als Bezugspunkt für die Winkelmessung dient zum Beispiel der magnetische Nordpol. Zur Berechnung der Position muss die Entfernung zwischen den beiden Referenzpunkten bekannt sein. Vereinfacht gesagt: Von zwei Punkten mit bekanntem Abstand werden Winkelmessungen zu beliebig anderen Punkten im Raum gemacht, um deren Position zu berechnen, wie es auch Abbildung 3.6 verdeutlicht. Im dreidimensionalen Fall ist dabei zu beachten, dass sich die zwei Geraden der beiden Basisstationen im Normalfall mathematisch nicht exakt schneiden.

3.5.2 Dead Reckoning

Dead Reckoning [RDM03], oder auch Koppelnavigation, ist ein Verfahren, das relative Positionierung verwendet. Die aktuelle Position wird auf Basis der vorherigen Position berechnet. Dies geschieht über Sensoren, welche in der Regel elektrisch arbeitende IMUs wie elektrischer Kompass, Beschleunigungssenser, Lagesensor oder ein mechanisch arbeitendes Hodometer[4]

[4] Kilometerzähler in Fahrzeugen.

sind. Dead Reckoning bezeichnet also die permanente Lokalisierung eines ME durch Bestimmung der Richtung, der Geschwindigkeit des MEs sowie der Zeit seit der letzten Positionsbestimmung. Zwischen den Messungen wird die Geschwindigkeit als konstant angenommen. Ausgehend von einer bekannten Ausgangsposition kann so also bestimmt werden, wo sich das ME nach einer Bewegung befindet.

3.5.3 Szenenanalyse

3.5.3.1 Location Fingerprinting (LFPT)

Location Fingerprinting ist ein tabellenbasiertes Verfahren zur Lokalisierung von MEs. LFPT wurde erstmals von Bahl et al. [BP00] im Jahre 2000 in WLAN Netzen verwendet, kann aber auch für GSM [ZYN08], [OVLL05], [SR00] oder Bluetooth eingesetzt werden. Das LFPT Verfahren basiert auf der Messung von Feldstärken, genauer gesagt der Feldstärke (RSS). Die Menge aller an einer Position gemessenen Feldstärken, aller empfangbaren WLAN-APs, GSM-BSs oder Bluetooth Landmarken werden als Eintrag, dem sogenannten Fingerprint, in einer Datenbank gespeichert. Um eine Lokalisierung durchführen zu können, muss deshalb zuerst in einer Trainingsphase (TRP) eine sogenannte Radio-Map (RM) erstellt werden. Dies geschieht durch Aufnahme von LFPTs an festgelegten Trainingspunkten (TPs). Die eigentliche Bestimmung der aktuellen Position wird anschließend in einer Realtimephase (RTP) durch Abgleich der aktuell gemessenen Feldstärken (RPs) mit allen TPs in der Datenbank durchgeführt. Das LFPT Verfahren wird hier nur kurz erläutert, weil dem Verfahren aufgrund der Tatsache, dass es bei der in dieser Arbeit implementierten Software Anwendung findet, das ganze Kapitel 4 gewidmet ist.

3.5.3.2 Visuelle Szenenanalyse

Ein weiteres Verfahren zur Szenenanalyse wird von Kourogi et al. [KK03] beschrieben. Die Autoren beschreiben ein System, welches ähnlich dem LFPT Verfahren arbeitet, aber auf einen visuellen Abgleich von Szenen anhand von Bildern als Daten setzt. Das vorgestellte System erstellt ein Innenraummodell in Form von Aufnahmen aller Orte. Die Bilder werden vorab erstellt und in einer Datenbank abgelegt. Diese Phase entspricht der Trainingsphase beim LFPT Verfahren. Die eigentliche Lokaliserung wird in einer zweiten Phase durchgeführt und versucht, durch einen Abgleich der in der Datenbank gespeicherten Bilder mit aktuell vom ME gelieferten Bilddaten auf die aktuelle Position der ME zu schließen. Das Verfahren hat jedoch den Nachteil, dass es sehr anfällig für Veränderungen in der Umgebung ist, da durch den Abgleich von Bilddaten jede Veränderung in der Umgebung zu einem größeren Abstand der zu vergleichenden Bildern führt.

3.5.4 Annäherungsverfahren

3.5.4.1 Cell of Origin (COO)

COO ist ein Verfahren, welches darauf basiert, die Position eines ME anhand seiner aktuell assoziierten Zelle zu bestimmen. Anhand des bekannten Standpunktes einer Basisstation, wie zum Beispiel einer GSM-Basisstation, einem WLAN-AP oder auch einer Bluetooth Landmarke, kann auf die aktuelle Position der ME geschlossen werden. Dieses Verfahren ist daher grobgranular und liefert eine maximale Genauigkeit auf Basis der Zellgröße. Das COO Verfahren wird vor allem in Mobilfunknetzen mit GSM Technologie eingesetzt, um eine Ortungen

von MEs innerhalb eines Providernetzes durchzuführen. Das COO Verfahren wird aber auch in Verbindung mit WLAN Lokalisierung eingesetzt und wird hier als Strongest Basestation (SBS) bezeichnet [BP00]. Der SBS Ansatz eignet sich vor allem für einen Einsatz in einem ILS, welches nur eine grobe Auflösung der Örtlichkeiten benötigt. Die Funktionsweise eines COO oder SBS Verfahrens ist in beiden Anwendungsfällen dieselbe.

Für das Funktionieren des COO Verfahrens ist es wesentlich, dass das ME immer zu der Zelle mit der besten Signalqualität, der aktuell stärksten Zelle verbunden ist. Um dies zu gewährleisten, muss ein Monitoring der aktuellen Zelle und aller Nachbarzellen durchgeführt werden, um bei Bedarf einen Wechsel der assoziierten Zelle vorzunehmen. Dieses Vorgehen wird gemeinhin als Handover bezeichnet. Es gibt verschiedene Strategien, wie Handover realisiert werden können [ESKF05]. Die einfachste Strategie ist das Durchführen eines *Wechsels aufgrund relativer Signalstärke*. Die Zelle wird hierbei genau dann gewechselt, wenn die Signalstärke der aktiven Basisstation unter die einer oder mehrerer benachbarter Zellen fällt. Eine Optimierung dieser Strategie ist der *Wechsel aufgrund relativer Signalstärke mit Schranke*. Das Einführen einer Schranke bedeutet, dass ein Wechsel zwischen zwei Zellen nur dann stattfindet, wenn die Feldstärke einer benachbarten Zelle stärker als die der aktiven Zelle ist und zugleich die Feldstärke der aktiven Zelle unter einen bestimmten Schwellenwert (Schranke) sinkt. Eine weitere Optimierung wird als *Wechsel aufgrund relativer Signalstärke mit Hysterese*[5] bezeichnet und soll das Auftreten von Ping-Pong Effekten verhindern. Weisen zwei oder mehrere Basisstation in etwa die gleichen Feldstärken auf, kann es passieren, dass permanent ein Handover durchgeführt wird, obwohl sich das ME nicht bewegt. Um diesem Problem zu begegnen, führt die Strategie mit Hysterese einen Absolutbetrag in dBm ein. Bevor ein Handover durchgeführt wird, muss sich das Signal zweier Zellen mindestens um diesen Betrag unterscheiden. Um die Handover Strategie noch zu verfeinern, können die beiden letzten Strategien kombiniert werden zu einem *Wechsel aufgrund relativer Signalstärke mit Schranke und Hysterese*.

Bei der GSM Technologie wird ein Handover zwischen verschiedenen Zelle autonom von einer der ME übergeordneten Netzkomponente, dem Base Station Controller (BSC) durchgeführt. Die Feldstärkemessungen zu allen empfangbaren Zellen werden direkt von dem ME durchgeführt[6] und an den BSC übermittelt. Die Entscheidung, wann ein Handover durchgeführt wird, trifft der BSC im Hintergrund und weist vor einem Wechsel in eine neue Zelle dem ME neue Frequenzen und Zeitschlitze zu.

Bei WLAN Netzen gibt es nach wie vor keinen einheitlichen Standard, der das Handover-Management durch Netzwerkkomponenten der Infrastruktur beschreibt, wie dies bei GSM Netzen der Fall ist. Der Grund hierfür ist, dass ein WLAN-AP für sich alleine nicht in der Lage ist, eine solche Entscheidung zu treffen. Es sei denn, er ist in einem Verbund mit anderen WLAN-APs organisiert. In der Regel liegt die Entscheidung, ob ein Handover durchgeführt wird, deshalb direkt bei dem ME[7], welches sowohl die Feldstärkemessungen als auch das Einleiten eines Handover übernimmt.

[5] Hysterese bedeutet das Zurückbleiben einer Wirkung hinter dem jeweiligen Stand der sie bedingenden veränderlichen Kraft.

[6] Feldstärkemessungen zur Realisierung von Handover nutzen einen speziellen Broadcast Control Channel (BCCH). Das ist ein Kanal, auf dem BSs mit voller Leistung senden. Die Funktionsweise des BCCH wird in Kapitel 3.3.2 noch genauer erläutert.

[7] Leider konnte nicht in Erfahrung gebracht werden, welche Strategie von ME auf Basis von Android dazu verwendet wird, einen Zell Wechsel einzuleiten. Man kann aber davon ausgehen, dass – wie bei anderen WLAN-Chipsätzen – die Strategie mit Schranke und Hysterese zum Einsatz kommt.

Der Vorteil des COO Verfahrens liegt in seiner Einfachheit. Das Verfahren ist einfach zu implementieren und die meisten Betriebssysteme für ME bieten hierfür einfache Schnittstellen zum Auslesen der aktuellen GSM-Zelle beziehungsweise des assoziierten WLAN-APs. Bei WLAN Netzen gilt diese Einfachheit der Implementierung jedoch nur in aktiven Netzen. Unter aktiven Netzen versteht man in diesem Zusammenhang Netze, in denen alle AP einer WLAN Infrastruktur bekannt und unabhängig von Verschlüsselung oder anderen Einschränkungen sind und zu denen jederzeit eine Verbindung aufgebaut werden kann. Der Handover wird hier direkt von der Hardware beziehungsweise dem Treiber des Betriebssystems vorgenommen. Bei der Verwendung von passiven WLAN Netzen findet jedoch im Gegensatz zu den aktiven Netzen keinerlei Aufbau einer aktiven Verbindung zu einem AP statt. Es wird stattdessen nur ein passiver Beacon-Scan nach WLAN-APs durchgeführt. Dieser beinhaltet unter anderem SSID, BSSID, Frequenz und Feldstärke. Da bei dieser Vorgehensweise zu keinem Zeitpunkt eine aktive Verbindung besteht und somit auch kein automatisierter Handover durchgeführt werden kann, muss man das Verhalten selbst in die Software implementieren. Dabei muss festgestellt werden, welcher AP gerade die beste Signalqualität aufweist und welcher AP somit die aktuelle SBS ist. Der Nachteil des Verfahrens wurde im Wesentlichen schon genannt und besteht darin, dass die Genauigkeit dieses Verfahrens immer auf die Zellgröße beschränkt ist.

3.5.5 Label und Tagging Verfahren

Gibt man dem Nutzer eines Systems die Möglichkeit, dem System seine aktuelle Position innerhalb eines Raumes mitzuteilen, spricht man allgemein von „labeln“ bzw. „taggen“. Neben dem Bereitstellen von Markern, die dem Nutzer präsentiert werden und zu denen er sich lokalisieren kann, gibt es eine weitere Möglichkeit, anhand einer Eingabe über das Display dem System seine aktuelle Position mitzuteilen. Bei diesem Vorgehen muss der Nutzer in einem ersten Schritt ein Tagging oder Labeling vornehmen, um einem bestimmten Ort einen Namen zu geben. Bhasker et al. [BBG04], bezeichnen dies auch als Virtual Access Point (VAPs). Bolliger et al. [BPCL09] benutzten das Verfahren des Labelns zur Optimierung des LFPT Verfahrens. Trainingspunkte, wie Positionen oder Orte, werden während der Trainingsphase mit Labeln versehen. Da der Nutzer dies in dem vorgestellten ILS auch zu einem späteren Zeitpunkt nachholen kann, wird die Optimierung als Asynchronous Interval Labeling (AIL) bezeichnet. In den folgenden Abschnitten werden Lokalisierungsverfahren, die dieses Label- und Tag-Prinzip mit Hilfe von visuellen Markern und Points of Interest (POI) verwenden, vorgestellt.

3.5.6 Markerbasierte Verfahren

Lokalisierungsverfahren, die auf Markern aufbauen, lassen sich grob in zwei Bereiche unterteilen: sichtbare und unsichtbare Marker. Sichtbare Marker können sowohl von Menschen als auch von Computern wahrgenommen werden, wohingegen unsichtbare Marker nur computergestützt erkannt werden können. Marker zeichnen sich dadurch aus, dass mit ihnen relativ einfach Information kodiert und in ein maschinenlesbares Format übertragen werden können. Ein Marker kann je nach verwendetem Typ eine bestimmte Menge an Information kodieren.[8]

[8] Werden Marker dazu benutzt, ganze Gebäudekomplexe oder allgemein größere Installationen abzudecken, kann es Probleme mit der Informationsdichte geben, welche in einer großflächigen Installation sehr hoch sein muss. Von der Informationsdichte, die ein Marker aufnehmen kann, hängt letztendlich die Granularität ab, mit der die Installation aufgelöst werden kann.

Abbildung 3.7: EAN-13-Barcode, wie er heute auf vielen Verpackungen in der Materialwirtschaft eingesetzt wird.

Im folgenden werden die in der Arbeit verwendeten QR-Codes genauer vorgestellt und das Prinzips des Mobile-Taggings beschrieben.

3.5.6.1 QR-Codes – Visual Tags

Die Abkürzung QR-Code steht für Quick-Response-Code oder 2-D-Barcode und bezeichnet eine Erweiterung, der seit Jahren in der Materialwirtschaft zur Automatisierung der Unternehmenslogistik verwendeten Barcodes, wie in Abbildung 3.7 dargestellt. Entwickelt wurden die QR-Codes im Jahre 1994 von der japanischen Firma Denso Wave und im Jahre 2000 wurden sie durch die ISO im Standard ISO/IEC 18004 standardisiert.

QR-Codes haben gegenüber 1-D-Barcodes den Vorteil einer wesentlich höheren Informationsdichte und sind omnidirektional abscannbar. Ein QR-Code speichert Informationen – im Gegensatz zu einem 1-D-EAN-13-Code – in vertikaler und horizontaler Richtung in einer 2-D-Matrix und kann bis zu 4,296 alphanumerische Zeichen oder 1,817 Kanji[9] kodieren [DW]. Zusätzlich besitzen QR-Codes viele Eigenschaften, die vor allem der Robustheit beim Auslesen mit Kamerascannern dienen. So können QR-Codes omnidirektional abgescannt werden und durch eine redundante Speicherung der Daten auch bei starker Verschmutzung noch ausgelesen werden. Letztendlich ist ein QR-Code eine effiziente Möglichkeit zum maschinellen Auslesen von Informationen. So ist es zum Beispiel einfach, eine Produktnummer in ein von Maschinen lesbares Format in Form eines QR-Codes zu bringen und anschließend Produkte automatisiert zu verarbeiten. Hingegen ist die Verwendung der später vorgestellten OCR-Algorithmen zur Texterkennung wesentlich komplexer und vor allem fehleranfälliger.

3.5.6.2 Mobile Tagging

Mobile Tagging (MT) beschreibt den Vorgang der Kodierung von Informationen mit Hilfe von QR-Codes als MTs, welche anschließend mit Hilfe eines Kamerascanners, wie er in allen gängigen ME zur Verfügung steht, ausgelesen werden können. Abbildung 3.9 gibt einen Überblick über die übliche Funktionsweise von MT. MTs sind RFID-Tags sehr ähnlich. Mit MTs können Objekte mit einer eindeutigen ID versehen und anschließend automatisiert verarbeitet werden. Man spricht deshalb in diesem Zusammenhang auch von einem „Physical-World-Link“. Dies ist insofern interessant, weil es ein Schritt in Richtung eines neuen Internet, des Internets der Dinge bedeutet. Jedem Objekt der physikalischen Welt kann mit einer virtuellen Instanz, zum Beispiel in Form einer URI [IETb], maschinenlesbare Information hinzugefügt

[9] Bezeichnung für chinesische Schriftzeichen, wie sie in der japanischen Schrift verwendet werden [Gra].

Abbildung 3.8: QR-Code der Berliner Verkersbetriebe (BVG), über den Kunden alle Information zur Linie U 22 abrufen können.

Abbildung 3.9: Typischer Ablauf beim Einsatz von Mobile-Tagging.

werden. Viele Information können heute schon über MT-URIs kodiert werden. Interessant sind hierbei die sogenannten „actionable URIs“. Actionable URIs kodieren nicht nur den Ursprung einer bestimmten URI anhand eines Pfades, sondern ermöglichen durch Angabe eines Schemas [ZXIb] zusätzlich, eine bestimmte Aktion in der virtuellen Welt mit einem Objekt zu assoziieren. Es gibt in diesem Bereich noch keine Standards, jedoch verfügen die meisten Kamerascanner über bestimmte Filter, welche anhand des URI-Schemas versuchen, eine Applikation zu bestimmen, die zur Interpretation der Daten in der Lage ist. So wird zum Beispiel bei einer URL, die das http-Protokoll verwendet, der Internetbrowser geöffnet, während beim Anliegen von Daten im VCard [IETc] die Kontaktverwaltung angesprochen wird. [ZXIb] gibt einen Überblick über Filter, die auf gängigen mobilen Plattformen[10] heute schon unterstützt werden. Darüber hinaus gibt es auch ein viel versprechendes Geo-URI-Format, welches in Zukunft in einem ILS Anwendung finden könnte.

Verwendung finden QR-Code basierte MTs in Marketing- und Informationskampagnen. Hierbei werden Objekte der physikalischen Welt, wie zum Beispiel Flyer, Verpackungen und Zeitschriften, mit zusätzlichen Informationen versehen. Diese Informationen können zum einen Produktinformationen ergänzen oder auch eine URI kodieren, die weiterführende Informationen zu den Produkten beinhaltet. Ein weiterer Bereich ist das Taggen von Objekten des alltäglichen öffentlichen Lebens, wie z.B. Sehenswürdigkeiten, Bushaltestellen, Stromkästen etc., die es ermöglichen, zusätzliche Informationen zu einem bestimmten Objekt zu erhalten. So setzten die Berliner Verkehrsbetriebe (BVG) QR-Codes ein, um Fahrplaninformationen zu bestimmten U-Bahn-Linien effizient an MEs auszuliefern – wie in Abbildung 3.8 gezeigt. Vermehrt werden QR-Codes auch zum Austausch von Kontaktinformationen und Profilen von sozialen Netzwerken benutzt und somit als elektronische Visitenkarten benutzt. Interessant ist auch der vermehrte Einsatz von QR-Codes zum Austausch allgemeiner textueller Informationen und Hyperlinks zwischen Computer und Mobiltelefon. So stellt zum Beispiel die Google Chrome Extension QR-ome [bke] bereit, Textinhalte von Internet Browsern mittels QR-Codes auf MEs zu transferieren.

Abschließend kann man sagen, dass QR-Codes und das MT ein großes Potential bieten und gute Chancen haben, in Zukunft als eine Möglichkeit zur Indoor-Lokalisierung verwendet zu werden. Die QR-Code-Scanner sind in fast allen gängigen ME vorhanden und werden durch den breiten Einsatz in immer mehr Bereichen des täglichen Lebens von den Nutzern der MEs akzeptiert. In Verbindung mit der Möglichkeit, physikalische und symbolische Positionen in MTs zu kodieren und diese Informationen ausführbar zu machen und mit Objekten der realen Welt verknüpfen zu können, könnte MT eines der Verfahren werden, die dazu verwendet werden, damit sich Nutzer selbstständig in geschlossenen Räumen zu einem Ort lokalisieren können.

3.5.6.3 Optical Character Recognition

Im Laufe dieser Arbeit wurde ebenfalls die Verwendung von Optical Character Recognition (OCR) Texterkennungs-APIs zum transkribieren einer symbolischen Position aus visuellen Markern untersucht. Verwendet wurde eine experimentelle Schnittstelle der Google Documents List Data API v3.0 [Goob]. Diese Lösung wurde gewählt, weil es aufgrund seiner cloudbasier-

[10] Unter anderem unterstützt durch Android, iPhoneOS, BlackBerry OS, Symbian.

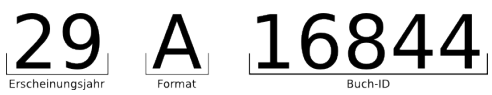


Abbildung 3.10: Buchsignaturformat der Universitätsbibliothek Tübingen. Der erste Teil der Signatur kodiert das Erscheinungsjahr, der zweite Teil das Format und der dritte Teil die ID des Buches.

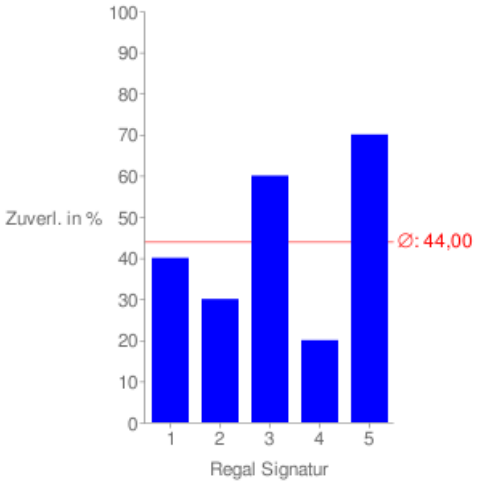


Abbildung 3.11: Zuverlässigkeit der Bilderkennung anhand visueller Marker in Form von Regalsignaturen der Universitätsbibliothek Tübingen, unter Nutzung der Google-Charts-OCR-API in Prozent. Auf der x-Achse dargestellt sind die einzelnen Regalsignaturen. Die durchschnittliche Genauigkeit ist als horizontale Linie eingetragen. Jede Signatur wurde zehn Mal fotografiert und anschließend der Bilderkennung zugeführt.

ten[11] Architektur einen leicht gewichtigen Ansatz zur optischen Texterkennung in Bildern auf ME bereitstellt.

Um die Robustheit der optischen Texterkennung auf MEs in der Praxis zu testen, wurde in der Universitätsbibliothek Tübingen versucht, anhand der an den Regalen angebrachten Buchsignaturen die Position der Bücherregale durch einen Kamerascanner mit Hilfe der OCR-API automatisiert auszulesen. Eine Buchsignatur besteht aus drei Teilen, die in Abbildung 3.10 dargestellt sind. Der erste Teil der Signatur kodiert das Erscheinungsjahr, der zweite Teil das Format und der dritte Teil die ID des Buches. An jedem Bücherregal sind Schilder angebracht, die einen Intervall definieren, welche Bücher sich im Regal befinden (siehe hierzu auch Abbildung 3.12). Abbildung 3.10 gibt einen Überblick über den Aufbau einer Signatur. Bei der Durchführung des Versuches wurden mit einem Android Testgerät (Nexus One) mit einer 5 Megapixel Kamera jeweils zehn Fotos von den Regalbeschreibungen gemacht und an die von Google bereitgestellte OCR-REST-API geschickt.

Die Ergebnisse des Versuches sind in Abbildung 3.11 dargestellt. Sie lassen das Potential der OCR Technologie erkennen. Die Ergebnisse zeigen aber auch, dass diese Technologie noch

[11] Cloud ist eine Abkürzung von Cloud Computing, was wörtlich übersetzt etwa „Rechnen in der Wolke“ bedeutet. Cloud Computing ist ein Architekturansatz, der komplexe Berechnungen und Datenoperationen an externe Systeme im Internet delegiert und die verarbeiteten Daten als Ergebnis zurückliefert.

Abbildung 3.12: Kennzeichnung der Bücherregale in der Universitätsbibliothek Tübingen.

nicht ausgereift ist und es zu hohen Fehlerraten kommt. So wurden im Durchschnitt nur 44 % der Bilder richtig ausgewertet. Ein weiteres Problem, welches nicht direkt mit der optischen Erkennung zusammenhängt, ist die hohe Latenzzeit zum Auswerten der Bilder. Diese Betrug bei einer Größe von einem Megabyte[12] in etwa 30 Sekunden. Dies ist insofern problematisch, weil die Versuche innerhalb des WLAN Netzes der UB Tübingen durchgeführt wurden und die durchschnittliche Latenz in einem Mobilfunknetz der dritten Generation aufgrund der geringeren Bandbreite noch wesentlich höher sein dürfte. Abschließend lässt sich feststellen, dass sowohl die hohe Fehlerrate als auch die hohe Latenzzeit den Einsatz einer cloudbasierten OCR zum jetzigen Zeitpunkt nahezu unmöglich machen und keine Alternative zu QR-Codes auf ME darstellen.

Saumenlosen Muster

Saito et al. [SHTH07] beschreiben noch einen Marker basierten Lokalisierungsansatz anhand von saumenlosen Mustern, welche die Position im Raum kodieren. Die Kodierung der Position wird hierbei durch die Anordnung der Winkel zwischen mehreren Mustern der gleichen Form kodiert. Der Ansatz liefert eine sehr hohe Genauigkeit im Zentimeterbereich. Zur eigentlichen Lokalisierung muss außer den Gegenständen, die als Träger des Muster verwendet werden, keine zusätzliche Infrastruktur errichtet werden. Träger des Musters können zum Beispiel Teppiche oder Tapeten sein. Einmal mit dem Muster bedruckt, können diese Träger für eine Lokalisierung mittels der Kamera eines MEs verwendet werden, weil die Positionen im Raum alleine durch die Anordnung der Muster kodiert wird.

[12] Dies entspricht einem 5 Megapixel Bild mit einer Kamera, die in dieser Konfiguration in vielen MEs zu finden ist.

3.5.7 Displaybasierte Verfahren

Durch die Vorgaben der Plattformhersteller haben alle gängigen Smartphones heute berührungssensitive LED- oder OLED-Displays mit Auflösungen von mehreren hundert Pixeln.[13] Dies erlaubt es, auf den Displays der MEs detaillierte Karten einer Indoor-Umgebung darzustellen und sie als Schnittstelle zur Nutzerlokalisierung zu benutzen. Bei dieser Art der Lokalisierung wird der Nutzer explizit in den Prozess der Lokalisierung miteinbezogen, indem er seinen Standpunkt dem System manuell mitteilt. In vielen Fällen, in denen einfach nur ein Dienst aufgrund eines dem Nutzer bekannten Aufenthaltsortes bereitgestellt werden soll, ist dies völlig ausreichend. Um diesen Punkt zu verdeutlichen, kann man zwei einfache Anwendungsfälle betrachten. Ein Anwendungsfall soll in Form einer Android Applikation die Energieeffizienz in Haushalten steigern. Dies soll dadurch gewährleistet werden, dass, wenn sich ein Bewohner innerhalb eines bestimmten Raumes befindet, alle anderen Lichtquellen im Haus abgeschaltet werden. Um dies zu realisieren, ist die denkbar einfachste Lösung, dass der Nutzer, der Kenntnis von seinem aktuellen Standort hat, diesen der Applikation mitteilt. Dies kann dadurch geschehen, dass er im Vorfeld symbolische Identifier in Form von Points of Interest (POI) wie Wohnzimmer, Schlafzimmer, Küche oder Toilette definiert, und eine Aktion mit ihnen verknüpft. Die Applikation könnte dann anhand des POI einen Server anfragen, mit der Bitte zu überprüfen, welche Lichtquellen im Haus angeschalten sind, und bei Bedarf gezielt Lichtquellen auszuschalten. Ein weiterer Anwendungsfall wäre ein ILS, welches in einem Supermarkt dazu eingesetzt wird, Gutscheine und weiterführende Informationen zu Angeboten und Bereichen innerhalb eines Supermarktes mithilfe von POI zu liefern. Hierzu könnten einzelne Bereiche der Supermärkte mit bestimmten visuellen Markern in Form von Icons oder Piktogrammen versehen werden, zum Beispiel Obst/Gemüse, Getränke, Süßigkeiten. Denkbar ist auch die Verwendung verschiedener Farben zur Kennzeichnung der einzelnen Bereiche. Diese Markierungen der realen Welt könnten auf eine Karte der Indoor-Umgebung abgebildet werden und dem Nutzer in Form der erwähnten POI präsentiert werden.

Die displaybasierten Ansätze sind aufgrund ihrer visuellen Ästhetik und der Möglichkeit zur Interaktion mit dem Touchscreen für den Einsatz einer leicht gewichtigen ILS Lösung prädestiniert. POI sind dabei die Elemente, die einen Ort in symbolischer und physikalischer Form repräsentieren. POI werden nachfolgend genauer betrachtetet.

3.5.7.1 Point of Interest

Points of Interest (POI) ist ein Begriff der vor allem im Bereich der Location Based Services (LBS) und der Navigationssysteme sehr populär ist. Ein POI beschreibt hierbei einen „Ort von Interesse“, welcher im allgemeinen durch eine Geo-Koordinate beschrieben wird und für einen Nutzer eines Lokalisierungs-Systems ortssensitive Informationen in das System einblendet. Diese Informationen können sehr allgemein sein. Restaurants, Tankstellen, Bankautomaten, Autowerkstätten und Sehenswürdigkeiten können sich aber auch auf den aktuellen Kontext des Nutzer beziehen. So kann eine Adaption des System durch eine Filterung der POI anhand der aktuellen Position oder gegebenenfalls durch Verfügbarkeit weiterer semantischer Informationen und Nutzerattribute, wie zum Beispiel, dass der Nutzer sich gerade im Urlaub befindet oder mit dem Auto unterwegs ist, durchgeführt werden.

[13] Das verwendete Testgerät G1 verfügt über eine Bildschirmauflösung von 320 x 480 Pixel (LED HVGA). Das zweite Testgerät Nexus One, welches zu den neusten am Markt verfügbaren ME gehört, weist aber schon Auflösungen von bis zu 800 x 480 Pixel (AMOLED WVGA) auf.

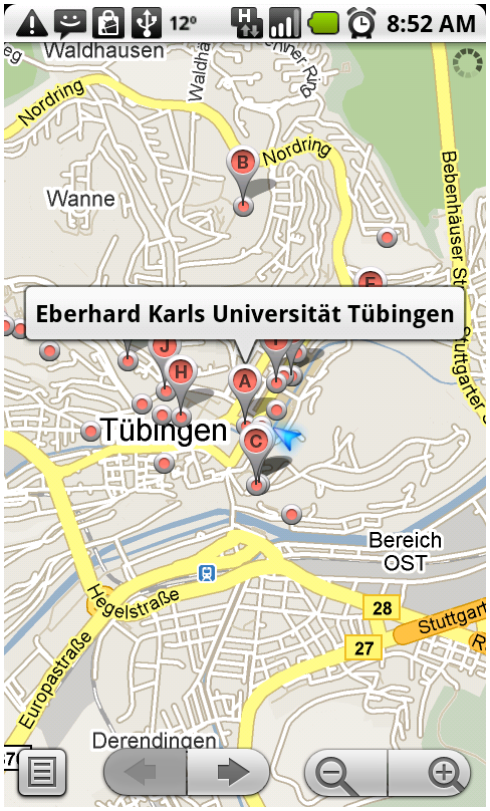


Abbildung 3.13: POI-Anzeige der von Google entwickelten Mobile-Maps Applikation. Gezeigt werden alle Ergebnisse einer Suche nach dem Begriff: Universität Tübingen, in Form von POI.

POI werden in Form kleiner Piktogramme oder Icons auf einer Karte dargestellt. Die Icons sind hierbei nichts anderes als getypte Hyperlinks, die eine Verlinkung zu weiterführender Information vornehmen. „Getypte“ bedeutet hier, dass durch eine visuelle Notation auf die grobe Klasse eines POI geschlossen werden kann.[14] Beim Anklicken eines POI werden dem Nutzer zusätzliche Informationen eingeblendet. Diese sind meist textueller Form, können aber auch Bilder, Audio, Video, Routing Information, Telefonnummern oder auch einen Remix dieser Informationen enthalten. Abbildung 3.13 zeigt die von Google entwickelte Applikation Mobile-Maps, welche eine ortssensitive Suche anhand der aktuellen Position ermöglicht. Die Position wird bestimmt durch GPS, WPS oder unter Verwendung der beschriebenen manuellen Ortsauswahl via Display. Die Nutzer können auch anhand einer festgelegten Klassifizierung (Attraktionen, Banken, Bars, Tankstellen, Hotels etc.) POI in der näheren Umgebung als Icons auf einer Google Map darstellen lassen und durch Klicken der Marker zusätzliche Infos zu den POI erhalten.

Das Prinzip der POI lässt sich genauso auf Umgebungen in geschlossenen Räumen übertragen und somit für die Lokalisierung in einem ILS verwenden. Der Unterschied ist, dass bei einem Klick auf einen POI nicht nur Informationen zum POI eingeblendet werden, sondern dass der Nutzer sich damit auch innerhalb des Systems lokalisieren und auf Dienste zurückgreifen kann, die kontextsensitiv auf seine Position reagieren. Der Vorteil bei der Verwendung

[14] Leider werden diese eigentlich sinnvollen, getypten POI noch selten eingesetzt. So scheint Google semantische Informationen zu den POI zu besitzen, diese aber nur nicht als getypte Marker darzustellen. (Stand März 2010)

von POI ist, dass sie sehr einfach zu implementieren und für den Nutzer leicht verständlich sind. Außerdem hat es sich gezeigt, dass diese Art von „Check-In“ basierten Systemen von Nutzern sehr gut angenommen werden.[15][16] Die Verwendung eines solchen Check-In Prinzips führt vor allem dazu, dass der Nutzer das Gefühl hat, seine Ortsbestimmung selbst durchzuführen, und so die Hemmungen vor dem Einsatz eines solchen System verliert beziehungsweise das Prinzip und die Nutzung des Dienstes zur Indoor-Lokaliserung verstanden hat. Darüber hinaus ergibt sich in POI basierten Systemen noch ein ganz anderer Vorteil: Nutzer fügen dem System eigene POI hinzu und arbeiten sozusagen als Co-Developer bei der Erstellung der Datenbasis des Systems mit.

POI werden in der Regel durch eine spezielle Auszeichnungssprache für Geo-Informationssysteme (wie GPX oder KML) beschrieben und enthalten neben den reinen 2- und 3-D-Koordinaten (wie Längen-, Breiten- und Höhengrad) auch noch symbolische Information (wie Name oder Straße). Darüber hinaus enthält vor allem KML viele weitere Sprachelemente zur Auszeichnung von Meta Informationen für die Darstellung und Annotation von POI.

3.5.7.1.1 GML, GPX, KML Zum Auszeichnen allgemeiner Geo-Daten in Geo-Informationssystemen haben sich mehrere Dateiformate durchgesetzt. Die drei wichtigsten Standards in diesem Bereich sind GML, GPX und KML und können alle dazu eingesetzt werden, POI zu beschreiben. Alle genannten Standards basieren auf XML und sind deshalb auch von Menschen gut lesbar. Vor allem ermöglichen sie aber einen vergleichsweise einfachen Austausch der Daten zwischen unterschiedlichen Applikationen, zum Beispiel durch Einsatz von XSLT Transformationen.

Die Geography Markup Language (GML) [Cona] ist ein vom Open Geospatial Consortium[17] entwickeltes Datei Format, XML-Schema, welches den Standard im Bereich der geographischen Auszeichnung von Objekten darstellt. GML erlaubt die Übermittlung von Objekten mit Attributen, Relationen und Geometrien im Bereich der Geo-Daten unter Einbeziehung von nicht-konventionellen Daten, wie Sensordaten. GML ist nicht gerade ein leichtgewichtiger Standard und eignet sich dazu, ein breites Spektrum von Objekten und deren Eigenschaften wie Gebäude oder Straßen zu beschreiben.

Im Vergleich hierzu ist die von Google entwickelte Keyhole Markup Language (KML), die insbesondere von Google Earth genutzt wird, aber auch von Google Maps, ArcGIS Explorer oder NASA World Wind gelesen werden kann, sehr einfach zu benutzen. KML ist eine Präsentationssprache zur Annotation von Geokarten in Geobrowsern und bietet Unterstützung zur Darstellung von 2- und 3-dimensionalen[18] Geo-Daten. KML ist ebenfalls eine XML basierte Auszeichnungssprache und die Bezeichnung Keyhole ist dem Namen der Firma Keyhole Inc. entlehnt, die das Format ursprünglich im Rahmen ihres „Keyhole Earth Viewer“[19] entwickelt hatte und von Google im Jahre 2004 aufgekauft wurde. KML ist seit April 2008[20] der offizielle, offene OGC Standard zur Auszeichnung von Geo-Daten in Geobrowsern. KML ist

[15] Der sogenannte Check-In erfreut sich immer größerer Beliebtheit bei LBSs und wird von mehreren Applikationen in diesem Bereich zum Prinzip erhoben, unter anderem von Foursquare und Gowalla.
[16] Das Check-In Prinzip lässt sich natürlich genauso auf die markerbasierte Lokalisierung mittels MT übertragen.
[17] Internationales Standardisierungsgremium für Geo-Daten.
[18] Breite, Länge und Höhe, genau in dieser Reihenfolge.
[19] Heute gemeinhin bekannt als der Geobrowser Google Earth.
[20] Seit KML Version 2.2.

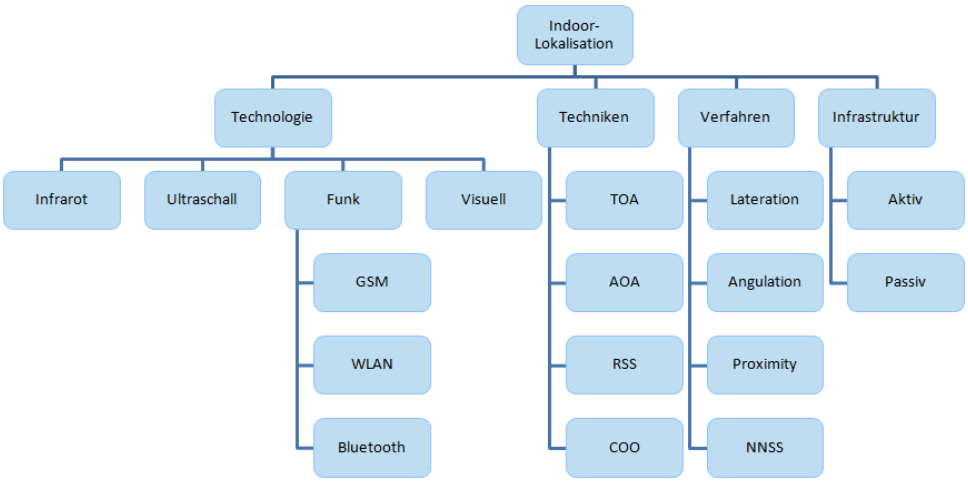


Abbildung 3.14: Klassifizierung und Einordnung der vorgestellten Indoor-Lokalisierungs: Technologien, Techniken, Verfahren.

vor allem durch den Einsatz in Google Earth und Maps sehr weit verbreitet. Einziger Nachteil ist, dass KML als Koordinaten-Referenzsystem nur das WGS 84 [wgs04] definiert und es so nicht möglich ist, wie bei GML eigene Referenzsysteme zu deklarieren. Listing D.2 in Anhang D zeigt einen Ausschnitt der .kml Datei, die der Abbildung 1.3 zugrunde liegt, welche die WLAN Abdeckung der Tübinger Innenstadt zeigt und in der Einführung zu dieser Arbeit abgebildet wurde.

Ein weiteres Dateiformat zur Repräsentation von Geo-Daten ist das von der Firma TopoGrafix entwickelte offene Dateiformat GPS Exchange Format (GPX). GPX basiert ebenfalls auf XML und dient dem saumenlosen Austausch von GPS-Daten zwischen verschiedenen Geräten. GPX ist wenig komplex und ist KML sehr ähnlich und eignet sich insbesondere zur Beschreibung von Punkten und Strecken. Als Koordinaten-Referenzsysteme wird wie bei KML nur WGS 84 unterstützt. Das Format wird bereits von einer Vielzahl von Geräten eingesetzt, um Geo-Daten zurückzuliefern. Es lässt sich leicht zu KML, zur Verwendung in Google Earth, transformieren.

Abschließend kann man festhalten, dass es wichtig ist, diese Formate in einem ILS direkt oder indirekt zu unterstützen. So kann man sein Format aufgrund der Tatsache, dass viele der gerade vorgestellten Formate in XML realisiert sind, welche sich über XSLT sehr leicht in andere Formate transformieren lassen, direkt als Basisformat verwenden. Andererseits sind XML Dateien schnell sehr groß und eignen sich zum Beispiel nicht dazu, in einem QR-Code kodiert zu werden. Des weiteren müssen in der Implementierung eines ILS intern andere Datenstrukturen als XML zur Repräsentation verwendet werden. Es muss deshalb in jeder Implementierung individuell über den Einsatz eines bestimmten Formats nachgedacht werden. Aus Sicht des Autors dieser Arbeit macht es aber am meisten Sinn, ein eigenes Format in Form einer URI zu definieren, dieses in der Implementierung intern durch eine Datenstruktur abzubilden und verschiedene Formate wie KML, GPX oder auch JSON als indirekte Export Formate anzubieten.

3.6 Fazit

In diesem Kapitel wurde eine umfassende Einführung in die Thematik der Indoor-Lokalisierung gegeben. Zum Abschluss soll noch einmal ein Überblick über die vorgestellten Themen erbracht werden. In Abbildung 3.14 dargestellt, ist deshalb eine Klassifizierung und Einordnung der vorgestellten Indoor-Lokalisierungs: Technologien, Techniken und Verfahren. Ergänzend wird in Tabelle 3.6 noch eine Gegenüberstellung bereits implementierter ILS, nach Technologien, Techniken, Verfahren, Genauigkeit und Zuverlässigkeit präsentiert.

System	**Technologie**	**Techniken**	**Verfahren**	**Genaugikeit**	**Zuverl.**
GPS	RF	ToF	Lateration	$1-5m$	$95-99\%$
Active Badge	IR	Proximity	-	Raumgröße	-
Active Bat	Ultraschall	ToF	Lateration	$9cm$	95%
Cricket	Ultraschall	Proximity	Lateration	$4x4ft$	100%
RADAR	WLAN	RSS	Szenenanalyse Triangulation	$3-4,3m$	50%
PinPoint 3D-iD	RF	ToF	Lateration	$1-3m$	-
SpotON	WLAN	RSS	Lateration	je nach Clustergröße	-
Ekahau	WLAN	RSS	proprietär	-	-
GoodTry	WLAN	ToF	Lateration	$4m$	90%
SSF	WLAN, GSM	RSS	Szenenanalyse	$2-3m$	80%

Tabelle 3.1: Gegenüberstellung bereits implementierter ILS, nach Technologien, Techniken, Verfahren, Genauigkeit und Zuverlässigkeit.

4 Location Fingerprinting

Location Fingerprinting (LFPT) ist ein vielversprechender Ansatz, um Indoor-Lokalisierung auf heutigen MEs zu realisieren. MEs der neusten Generation verfügen über Sensoren für diverse Funktechnologien, unter anderem für WLAN, GSM und Bluetooth. Dieses Kapitel gibt eine Einführung in die Methoden und Techniken zur Lokalisierung anhand des Location Fingerprinting Verfahrens. Hierbei beschränkt sich diese Arbeit auf WLAN und GSM basiertes Location-Fingerprinting.
Die Basis für das Location Fingerprinting Verfahren bildet die Abnahme der Feldstärke (RSS) mit der Entfernung zum Sender. Um dies Verhalten zu verdeutlichen, zeigt Abbildung 4.2 einen sogenannten Distance-Walk mit einem ME über eine Entfernung von $30m$. Hierbei sieht man sehr schön die Veränderungen der WLAN-Feldstärken zu den vier empfangenen WLAN-APs. Hinzu kommt, dass Funksignale durch Wände, Menschen und andere Objekte abgelenkt, reflektiert und absorbiert werden. Die Folge ist eine charakteristische und messbare Ausprägung der RSS an jeder diskreten Position im Raum. Es gilt hierbei, je komplexer die Umgebung, desto unterschiedlicher ist die Signalcharakteristik an einer bestimmten Position.

Im Folgenden wird zunächst ausgeführt, wie das Location Fingerprinting funktioniert. Dabei wird beschrieben, dass, um in der Praxis die Signalcharakteristik an einer Position zu bestimmen, alle messbaren RSS-Werte der Umgebung in einer Trainingsphase gesammelt werden und als sogenannte Fingerprints (FPT) in einer Datenbank abgespeichert werden. Zur Lokalisierung eines ME wird in der Realtimephase ein FPT aufgezeichnet und mit allen FPT in der Datenbank vergleichen. Es wird dabei auch auf den genauen Aufbau eines FPT eingegangen und auf die Eigenschaften und Unterschiede der Signalcharakteristik von WLAN und GSM. Danach wird die Datenverarbeitung beschrieben, die nötig ist, um die im Anschluss beschrieben Lokalisierungsalgorithmen einzusetzen. Im weiteren Verlauf des Kapitels wird dann auf grundlegende Architekturen und mögliche Infrastrukturen für ein LFPT-ILS eingegangen. Im letzten Teil des Kapitels werden die allgemeinen Systemparameter zum Aufbau eines LFPT-ILS im Detail beschrieben.

4.1 Radio-Map

Eine Radio-Map (RM) wird in einer Datenbank gehalten und bietet die Grundlage für die Lokalisierung. Die RM speichert die Zustände der Signalcharakteristika aller in der Trainingsphase aufgezeichneten FPT. FPT werden in der Regel als Zeilen innerhalb der RM abgespeichert. Dabei gibt es pro Funktechnologie, wie WLAN oder GSM, eine eigene Tabelle. Abbildung 4.1 stellt eine exemplarische WLAN-RM dar. In der Abbildung sieht man, dass pro WLAN-AP oder pro Zelle bei GSM, eine Zeile in der Tabelle benötigt wird und ein FPT sich somit über mehrere Zeilen und Tabellen erstreckt. Ein FPT besteht also aus allen Zeilen und aus eventuell mehreren Tabellen mit derselben Trainingspunkt ID.[1] Eine RM ist dabei nicht unbedingt starr,

[1] Sicherlich kann man dies auch anders realisieren, allerdings hat sich im Lauf der Arbeit herausgestellt, dass diese Art der Speicherung am geeignetsten ist, um ein dynamisches Wachstum der RM zu ermöglichen.

	ID	igp_name	SSID	MAC	MeanFrequency	RSSMean
1	0	TP00	Morpheus	00:16:38:42:b6:51	2462	-76.000
2	0	TP00	WLAN-001A4F147C25	00:1a:4f:14:7c:25	2462	-93.9999
3	0	TP00	yogibaer	00:d0:de:84:bc:5f	2462	-84.1269
4	0	TP00	WLANJBK	00:30:f1:e7:2f:4e	2462	-93.9999
5	0	TP00	ThoughtWorks	00:22:6b:70:1b:80	2412	-51.0058
6	0	TP00	o2DSL	00:19:cb:9f:e9:00	2412	-92.0000
7	0	TP00	CocaCola_S	00:23:08:09:c0:15	2447	-59.0000
8	0	TP00	o2DSL	00:19:cb:ca:10:16	2412	-94.0000
9	0	TP00	IJTM	00:1a:4f:02:4d:55	2412	-94.0001
10	0	TP00	WLAN-001F3F6A668B	00:1f:3f:6a:66:8b	2412	-94.0000
11	0	TP00	o2DSL	00:19:cb:9f:c9:94	2412	-93.9999
12	100	TP01	ThoughtWorks	00:22:6b:70:1b:80	2412	-34.000
13	100	TP01	Morpheus	00:16:38:42:b6:51	2462	-95.9955
14	100	TP01	WLAN-001A4F147C25	00:1a:4f:14:7c:25	2462	-96.9998
15	100	TP01	CocaCola_S	00:23:08:09:c0:15	2447	-84.9999
16	100	TP01	yogibaer	00:d0:de:84:bc:5f	2462	-95.9970

Abbildung 4.1: Datenbanktabelle einer WLAN-RM mit Einträgen für zwei aufgenommene Trainingspunkte TP00 und TP01.

sondern kann flexibel sein und nicht nur initial mit Werten befüllt, sondern über die Zeit immer wieder dynamisch angepasst werden. Dies ist insbesondere bei Funktechnologien wie WLAN, die starken Interferenzen ausgesetzt sind, unabdingbar.

4.2 Trainingsphase und Realtimephase

Die Trainingsphase (TRP), in der Literatur auch als Offlinephase bezeichnet, dient dazu, die RM initial anzulegen. Dazu nimmt ein ME an zahlreichen definierten Punkten, sogenannten Trainingspunkten (TP), die Signalcharakteristik in Form eines LFPTs auf und speichert diesen zusammen mit den Koordinaten seiner Position und/oder einem symbolischen Identifier in die RM.

In der Realtimephase (RTP), auch Onlinephase genannt, nimmt ein ME erneut einen FPT auf, diesen bezeichnet man auch als Realtime-Fingerprint (RTFPT); die Position der Aufnahme als Realtimepunkt (RP). Ein RTFPT dient dazu, ihn mit den LFPTs in der RM zu vergleichen. Die Koordinaten der Position des LFPT aus der RM, welcher dem aktuellen RTFPT am Ähnlichsten ist, zum Beispiel der LFPT mit der geringsten euklidischen Distanz, wird während der RTP als die aktuelle Position des ME ausgegeben.

4.3 Aufbau eines Fingerprint

Ein Fingerprint kann als eine Datenstruktur mit symbolischem Identifier und den dazugehörigen Daten angesehen werden [PKL07]. Der grundlegende Aufbau eines Fingerprints ist bei LFPTs und RTFPTs gleich. LFPTs speichern zusätzlich zu den unten aufgeführten Messdaten Angaben zur Position mit ab. Dies können relative Koordinaten, GPS Koordinaten nach WGS 84, oder einfach nur symbolische Identifier sein. Ansonsten ist der Aufbau eines FPT gegeben durch:

- RSS Werte,
- Zell-IDs und BSSIDs,
- Orientierung,

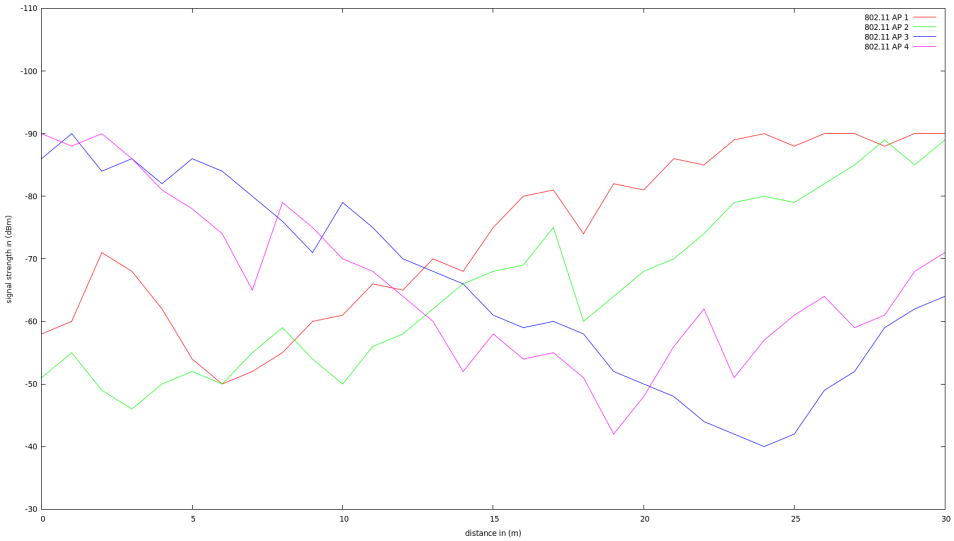


Abbildung 4.2: Ein sogenannter distance Walk über 30 m. Dargestellt wird die RSS von 4 WLAN APs

- Zeitpunkt der Aufnahme (timestamp),
- Genauigkeit, Varianz und Standardabweichung,
- Bewegung (dynamic state) oder Ruhelage (static state).

4.4 Techniken

Bislang wurde beschrieben, wie das LFPT-Verfahren theoretisch funktioniert. Beim praktischen Einsatz des Verfahrens müssen die Eigenschaften und auch die Unterschiede der Signalcharakteristik von WLAN und GSM berücksichtigt werden und möglichst so eingesetzt werden, dass sie das Verfahren unterstützen. Dabei wird zunächst ausführlich auf Techniken für WLAN eingegangen. Vieles davon gilt genauso oder zumindest in ähnlicher Form für jede Funktechnologie. Deswegen wird auf GSM auch nur kurz eingegangen und nur die für GSM besonders charakteristischen Merkmale hervorgehoben.

4.4.1 WLAN-LFPT

Viele der in Kapitel 2 vorgestellten Forschungsarbeiten nutzen die WLAN-Technologie für das LFPT-Verfahren, um die Lokalisierung eines ME durchzuführen. Allen Arbeiten gemeinsam ist die Erkenntnis, dass die Signalausbreitung von WLAN starken Interferenzen unterworfen ist. Um das LFPT-Verfahren besser zu verstehen und um selber intelligente Algorithmen entwerfen zu können, ist es deshalb wesentlich, sich mit den Eigenschaften der RSS zu befassen, da mehrere Faktoren die Genauigkeit und Zuverlässigkeit eines LFPT-ILS beeinflussen. Nur so kann ein optimiertes LFPT-Verfahren entworfen werden. Daher wird nun ausführlich auf die Signalcharakteristik von WLAN eingegangen.

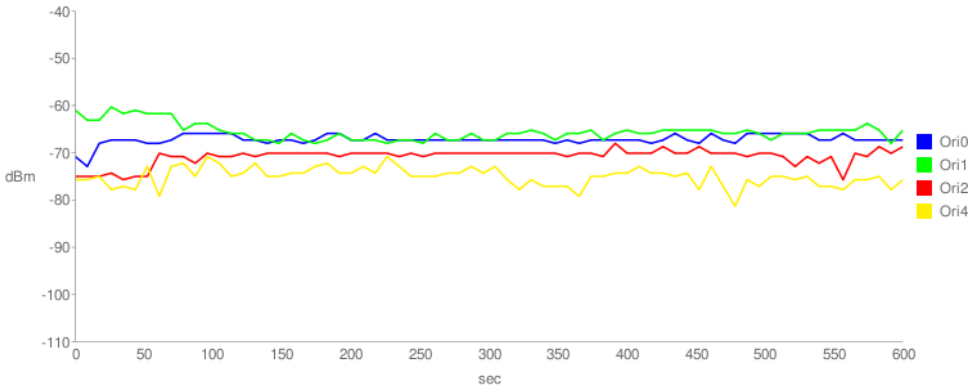


Abbildung 4.3: Auswirkung der Nutzer-Orientierung an einem festen Standort auf die RSS. Die Orientierung ist in 4 Intervalle eingeteilt mit Ori0 = 0° – 89°, Ori1 = 90° – 179°, Ori2 = 180° – 269°, Ori3 = 270° – 359°.

4.4.1.1 Eigenschaften der RSS Signalwerte in Bezug auf optimiertes LFPT

4.4.1.1.1 Signalausbreitung Die Feldstärke einer elektromagnetischen Welle mit der Sendeleistung T_x in Watt und dem Radius r in Metern, fällt mit $\frac{T_x}{r^2}$ ab. Weiterhin wird angenommen, dass sich die Signalausbreitung wie eine Normal- bzw. Gaussverteilung verhält [KK04]. Wie jedoch die Versuche in [LBR+02] zeigen, ist das aber nicht immer so. In den Messungen wurde eine leichte Linkslastigkeit der Verteilung festgestellt. Allgemein ist es sehr schwierig, verlässliche Aussagen über die Verteilung der Signalausbreitung zu machen.

4.4.1.1.2 Anwesenheit von Personen im Raum Die Anwesenheit von Personen im Raum hat großen Einfluss auf die gemessene RSS. So wurde in einer der ersten Arbeiten über WLAN basiertes LFPT von Bahl et al. [BP00] festgestellt, dass Personen im Raum starke Fluktuationen der RSS-Werte um bis zu $5\,dBm$ verursachen können. Des weiteren stellten sie eine Korrelation zwischen der Orientierung des Nutzer und den gemessenen RSS fest. Kaemarungsi et al. [KK04] kommen bei ihren Untersuchungen zu ähnlichen Ergebnissen. Bei anwesenden Personen erhöhte sich die Standardabweichung im Testaufbau von $0.68\,dBm$ auf $3\,dBm$, die Mittelwerte der RSS änderten sich von $-70.4\,dBm$ auf $-71.6\,dBm$. Im Verlauf dieser Arbeit wurde ebenfalls die Auswirkung der Nutzerorientierung auf die absoluten RSS untersucht. Abbildung 4.3 zeigt die mit einem HTC G1 gemessenen RSS zu einem WLAN-AP in $8\,m$ Entfernung mit DLOS. Die Messungen wurden am selben Trainingspunkt über einen Zeitraum von 300 Sekunden durchgeführt, erfasst wurden die absoluten RSS-Werte in jeweils 4 Orientierungen mit Abstand von jeweils 90° aus den Intervallen $Ori0 = 0° - -89°, Ori1 = 90° - -179°, Ori2 = 180° - -269°, Ori3 = 270° - -359°$. Die Werte zeigen anschaulich die Auswirkung der Orientierung des Nutzers auf die absoluten RSS-Werte. Am Deutlichsten zeigt sich der Effekt beim Vergleich $ori1$ (DLOS) und $ori3$ (NLOS). Die Mittelwerte der beiden Orientierungen unterscheiden sich um $5,31\,dBm$ und die Standardabweichung erhöhte sich von $1.1\,dBm$ auf $2\,dBm$. Ladd et al. [LBR+02] liefern in ihrer Arbeit eine Antwort: den „Blocking-Effekt“ des menschlichen Körpers. WLAN sendet auf dem $2.4\,GHz$ Band. Unglücklicherweise ist $2.4\,GHz$ auch die Resonanzfrequenz von Wasser. Da der menschliche Körper zu 70 % aus Wasser besteht, absorbiert er Teile der von einem WLAN-AP ausgesendeten Signale und führt zu einer signifikanten Abschwächung der empfangenen RSS von bis zu $9.32\,dBm$ bei NLOS. Um dieser Problematik entgegen zu wirken, wird deshalb vorgeschlagen, beim Aufnehmen

der LFPTs zusätzlich noch die Orientierung des MEs zu erfassen, um den Blocking-Effekt zu minimieren. Zur Verwendung der Orientierung bei der Aufnahme eines LFPT empfiehlt sich zum effizienteren Abgleich der Orientierung, in der RTP eine Quantisierung der Orientierung vorzunehmen. Zur Granularität der Quantisierung der Orientierung gibt es in der Literatur keine konkreten Angaben. So schlagen King et al. [KKH+06] vor, eine Anzahl von 5 Orientierungsintervallen vorzunehmen, so dass der Abstand der Winkel folgendermaßen verteilt ist: $\frac{360^\circ}{69^\circ} \simeq 5$. Young et al. [YZN07] schlägt hingegen vor, nur 4 verschiedene Winkel im Abstand von 90° jeweils nach Himmelsrichtungen $ori = \{Norden, Westen, Süden, Osten\}$ zu verwenden. Genaueres zur Verwendung der Orientierung im später vorgestellten SSF findet sich in Kapitel 5.3.3.

Zusammenfassend lässt sich feststellen, dass es sich empfiehlt, aufgrund des Blocking-Effekts des menschlichen Körpers, bei der Aufnahme von LFPT die Orientierung – wenn verfügbar – mit aufzunehmen, um robustere LFPTs zu erhalten. Weiterhin sollten die Messungen möglichst unter realen Bedingungen stattfinden, das heißt, ein möglichst realitätsnahes Setup beim Aufzeichnen der LFPTs zu wählen: mit anwesenden Personen im Raum und die Messungen mit einem Probanden, der das ME in der Hand hält, durchführen.

4.4.1.1.3 Standardabweichung Die Standardabweichungen der RSS derselben WLAN-APs sind in der Regel, wenn die DLOS nicht gerade durch die Orientierung des Nutzer verdeckt ist, gleichbleibend [KK04]. Eine wichtige Feststellung ist die Beziehung der durchschnittlichen RSS Werte zur Standardabweichung: Je größer die Entfernung von mobilem Endgerät und AP, desto geringer ist die Standardabweichung. In [SK08] wurde herausgefunden, dass die Standardabweichung der RSS $6-7\,dBm$ beträgt, wenn das ME nah ($-60\,dBm$ bis $-40\,dBm$) an einem AP lokalisiert ist und DLOS besteht. Im Gegensatz dazu beträgt die Standardabweichung nur $1-2\,dBm$, wenn sich das ME weiter entfernt von einem AP befindet ($-95\,dBm$ bis $-85\,dBm$). Dies ist meistens bei NLOS der Fall. Eine hohe Standardabweichung führt auch zu einer höheren Fehlerwahrscheinlichkeit bei der Auswahl der Position und führt dazu, dass LFPTs nur sehr schwer zuzuordnen sind. Eine Lösung besteht zum Beispiel darin, für die Bereiche in Antennennähe eine höhere Auflösung des Rasters zu wählen und somit mehr TPs in diesen Bereichen aufzunehmen.

4.4.1.1.4 Auswirkungen der Uhrzeit auf die RSS Die RSS kann ebenfalls eine Varianz bezüglich der Zeit aufweisen. Abbildung 4.4 zeigt eine Messung der absoluten RSS-Werte über 4 Stunden. Genauere Untersuchungen zur Auswirkung der Uhrzeit auf die RSS befinden sich in [KK04]. Die Ergebnisse zeigen, dass es zu unterschiedlichen Tageszeiten Unterschiede in den gemessenen RSS Werten geben kann. Diese sind vor allem darauf zurückzuführen, dass zu bestimmten Zeiten eine höhere Frequenz in der Benutzung der Räume vorliegt, die Umgebung sich fortlaufend ändert oder Wetterbedingungen, wie starker Regen, die Signalausbreitung beeinflusst. Weiterhin haben Versuche im Rahmen dieser Arbeit gezeigt, dass Lastspitzen der gemessenen WLAN AP ebenfalls signifikante Auswirkungen auf die RSS haben und im Vergleich zu weniger belasteten APs eine höhere Varianz der RSS aufweisen.

4.4.1.1.5 Weitere Eigenschaften

Unabhängigkeit und Interferenzen zwischen verschiedenen APs Die Versuche von Kaemarungsi et al. [KK04] zeigen, dass die einzelnen RSS-Werte der APs unabhängig voneinander sind und dass es auch keine Interferenzen zwischen den einzelnen APs gibt, selbst

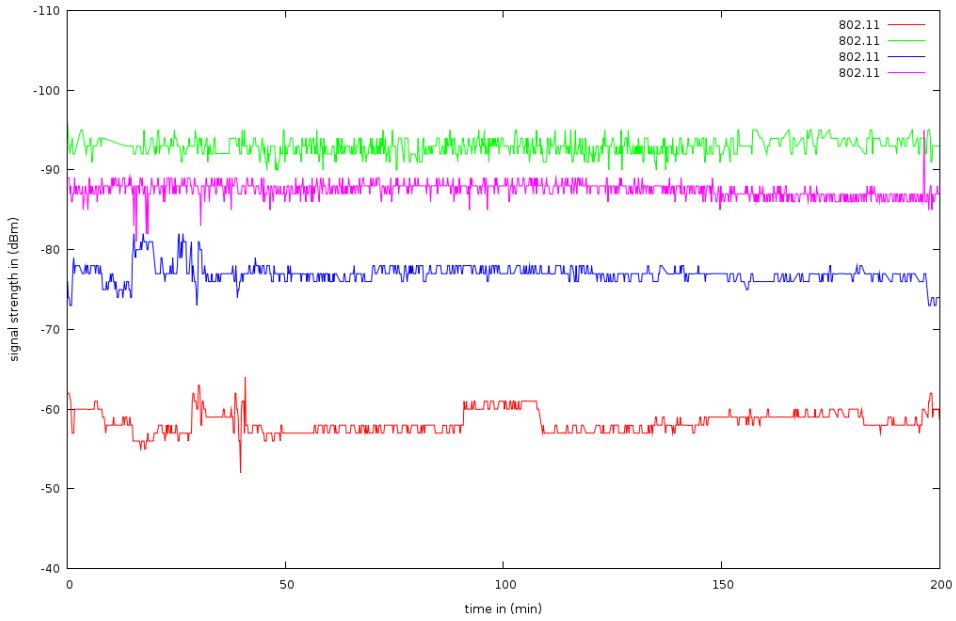


Abbildung 4.4: Messung der RSS von 4 unterschiedlichen 802.11 APs, über einen Zeitraum von 3 Stunden und 20 Minuten.

wenn Sie auf demselben Kanal senden. Dies ist laut Kaemarungsi et al. auf die MAC Schicht von WLAN zurückzuführen, die nur dann sendet, wenn das Medium auch frei ist.

Atmosphärische Störungen Ladd et al. [LBR+05] weisen noch darauf hin, dass atmosphärische Änderungen, zum Beispiel Luft, Temperatur oder Regen, sich ebenfalls auf die Signalausbreitung auswirken können.

4.4.2 GSM-LFPT

Zum Aufbau eines ILS für ME ist es naheliegend, neben WLAN Netzen auch die heute allgegenwärtigen Mobilfunknetze nach dem GSM Standard zu nutzen, um die Lokalisierung eines ME durchzuführen. GSM kann ebenso wie WLAN zur Erstellung von RSS basierten LFPTs verwendet werden. GSM ist heutzutage, vor allem im innerstädtischen Bereich, flächendeckend verfügbar und bietet eine wesentlich höhere Abdeckung als WLAN Netze. GSM Netze sind zum einen sehr stabil und weisen dennoch eine Abweichung bezüglich der gemessenen RSS an bestimmten Punkten im Raum auf. GSM-LFPT eignet sich vor allem dort, wo keinerlei andere WLAN-Infrastruktur vorhanden ist oder nicht installiert werden kann.

Einen Vorteil bei der Verwendung von GSM-LFPT ist, dass GSM Netze aufgrund ihrer fixen Infrastruktur sehr stabile Werte bei den Messungen der RSS auch über einen längeren Zeitraum aufweisen. Dies lässt sich auch darauf zurückführen, dass GSM ein lizenziertes Frequenzband benutzt und es nicht zu Interferenzen mit Geräten auf derselben Frequenz kommen kann. Ein weiterer Vorteil bei der Verwendung GSM ist die Tatsache, dass diese Netze praktisch immer verfügbar sind, zum Beispiel auch bei einem lokalen Stromausfall [ZYN08].

Die einfachste Methode, um einen GSM-LFPT zu erstellen, beruht auf Messungen der RSS der aktiven Zelle, das heißt der Zelle, zu der das ME aktuell verbunden ist. Bei Verwendung dieser Methode lassen sich, wie in den Arbeiten von Shyy et al. [SR00], Zhou et al. [ZYN08] und Otsason et al. [OVLL05] beschrieben, schon beachtliche Genauigkeiten erzielen. Eine noch höhere Genauigkeit kann erzielt werden, wenn man zusätzlich den beschriebenen BCCH dazu einsetzt, um auch die RSS der empfangbaren Nachbarzellen zu messen. Otsason et al. [OVLL05] haben durch die Verwendung der RSS von den 6 stärksten Funkzellen zusammen mit bis zu 29 schwächeren GSM-Kanälen[2] eine Genauigkeit mit einer durchschnittlichen Abweichung von ca. $2,5\,m$ erreicht. Des weiteren wurde festgestellt, dass mit GSM zuverlässig das aktuelle Stockwerk, in der sich eine ME befindet, unterschieden werden kann.

4.5 Datenverarbeitung

Mit dem Wissen, wie das LFPT Verfahren theoretisch funktioniert, und Kenntnis von eben beschriebenen Signalcharakteristika ist es nun als nächstes wichtig zu verstehen, wie die in der TRP aufgezeichneten LFPTs in der RM mit den RTFPTs aus der RTP verglichen werden können. Dazu müssen die Daten in den LFPTs zunächst aufbereitet werden. Dieser Abschnitt geht daher zunächst auf die Datenverarbeitung bei der Verwendung des LFPT-Verfahrens ein. Allgemein gilt: die von den WLAN- und GSM-Sensoren aufgezeichneten RSS-Rohdaten sollten nach Möglichkeit nicht direkt in der RM gespeichert werden. Es ist sinnvoll, die Daten aufgrund der beschriebenen Eigenschaften der Signalausbreitung vor dem Speichern in die RM in einem ersten Schritt aufzubereiten. Hierzu gibt es mehrere Strategien, wie die Messdaten verarbeitet werden können. Diese einfachen statistischen Berechnungen werden nachfolgend vorgestellt und dann mit der Signal Strength Difference eine Methode vorgestellt, um geräteunabhängige LFPTs zu erstellen.

4.5.1 Mittelwert und Standardabweichung

In einem ersten Schritt bei der Datenverarbeitung sollten Mittelwert und Standardabweichung der gemessenen RSS-Werte bestimmt werden. Das verwenden der Rohdaten ist nicht zu empfehlen, da – wie in diesem Kapitel beschrieben – die Charakteristik eines Raumes bezüglich der Signalausbreitung starken Fluktuation ausgesetzt ist und eine einzelne Messung in diesem Zusammenhang nicht ausreichend ist. In dieser Arbeit wurde deshalb eine konfigurierbare Anzahl von Samples pro Sensor aufgezeichnet und anschließend Mittelwert und Standardabweichung aus den anliegenden RSS-Werten berechnet und abgespeichert. In den in Kapitel 7 dokumentierten Messreihen wurde die Anzahl der Samples deshalb immer separat angegeben und liegt meistens im Bereich um die 10 Samples. Bisherige LFPT basierte ILS haben eine ähnliche Anzahl an Samples verwendet. Die Standardabweichung kann neben dem Mittelwert als Maß für die Genauigkeit herangezogen werden und spielt vor allem bei der Verwendung probabilistischer Algorithmen eine große Rolle.

4.5.2 Signal Strength Difference (SSD)

Bei der Aufnahme von LFPTs empfiehlt es sich je nach Einsatz des ILS, nicht nur die absoluten RSS-Werte an den Trainingspunkten aufzunehmen, sondern zusätzlich noch die Signal

[2] Kanäle, die zwar empfangen werden können, aber nicht für eine Kommunikation geeignet sind.

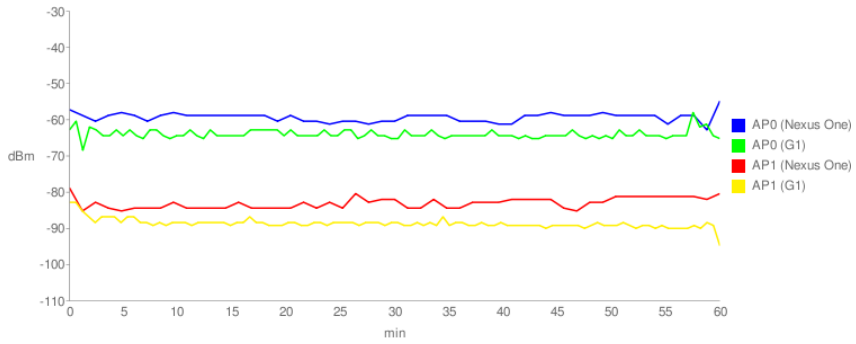


Abbildung 4.5: Differenz der gemessenen RSS zu zwei APs mit mehreren Endgeräten am selben Trainingspunkt.

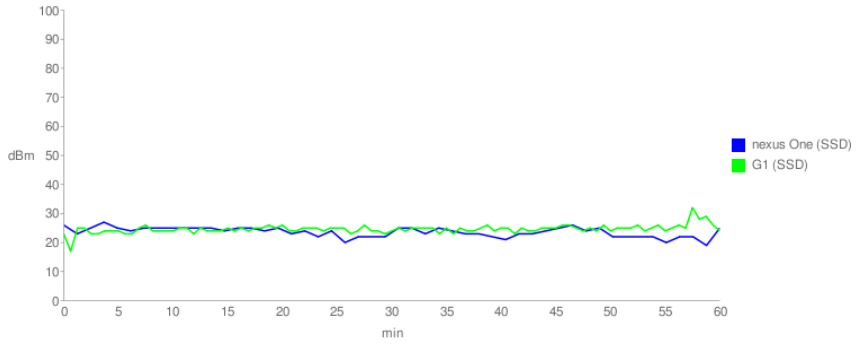


Abbildung 4.6: Signal Strength Difference zu zwei APs mit mehreren Endgeräten am selben Trainingspunkt.

	E(X)[dBm]	Var(X)[dBm]	σ[dBm]	Δ[dBm]
AP0 (Nexus One)	-59.3	1.31	1.15	4.5
AP0 (G1)	-63.8	1.21	1.10	4.5
AP1 (Nexus One)	-82.9	1.96	1.40	5.6
AP1 (G1)	-88.5	1.69	1.30	5.6

Tabelle 4.1: Mittelwert, Varianz, Standardabweichung und Differenz der Mittelwerte der mit dem HTC G1 und Nexus One aufgenommenen absoluten RSS Werte.

	E(X)[dBm]	Var(X)[dBm]	σ[dBm]	Δ[dBm]
AP0/AP1 (Nexus One)	-23.58	2.84	1.69	1.06
AP0/AP1 (G1)	-24.64	2.19	1.48	1.06

Tabelle 4.2: Mittelwert, Varianz und Standardabweichung der mit dem HTC G1 und Nexus One aufgenommenen SSD-Werten von AP0 und AP1.

Strength Difference (SSD). Die SSD gibt anstatt der absoluten RSS-Werte pro AP die Differenz der RSS zwischen zwei APs am selben TP wieder. So wird, wenn zwei APs an einem TP empfangen werden können, nur die Differenz der Mittelwerte der gemessenen absoluten RSS-Werte als LFPT verwendet oder dieser um die SSD ergänzt. Die Verwendung der SSD ist vor allem dann sinnvoll, wenn in einer Installation mehrere ME verschiedensten Typs verwendet werden.[3] So zeigt Abbildung 4.5 die absoluten RSS-Werte zu zwei verschiedenen WLAN-APs mit zwei unterschiedlichen MEs. Zum einen das HTC G1 und zum anderen das HTC Nexus One. Beide ME nutzen das Android Betriebssystem und verfügen zur Aufnahme der Werte über dieselbe Version des SSF. Die Werte wurden jeweils an denselben TPs über einen Zeitraum von 60 Minuten aufgenommen. Die Werte zeigen eine deutliche Differenz zwischen den beiden Endgeräten. So beträgt die Abweichung der Mittelwerte zwischen den beiden ME an AP0 $4.5\,dBm$ und AP1 $5.6\,dBm$. Mittelwert, Varianz, Standardabweichung und die Differenz sind in Tabelle 4.5.2 dargestellt. Im Vergleich hierzu zeigt Abbildung 4.6 die SSD zwischen den beiden ME. Die Werte wurden aus den Messwerten abgeleitet, welche für die absoluten RSS-Werte verwendet wurden. Schon aus der Abbildung wird deutlich, dass die gemessenen SSD-Werte eine viel geringere Abweichung der Mittelwerte aufweisen. So beträgt diese im Falle von AP0/AP1 gerade einmal $1.06\,dBm$. Alle Werte sind in Tabelle 4.5.2 dargestellt.

Dieses Verhalten ist insofern problematisch, weil es bedeutet, dass ein und dieselbe RM nicht von den gleichen Endgeräten benutzt werden kann. Was wiederum bedeutet, dass jedes ME eine eigene RM der Räumlichkeiten erstellen müsste. SSD kann deshalb eine Strategie zur Bestimmung robusterer, vom Endgerät unabhängigerer LFPTs darstellen und wurde von Hossain et al. [M. 07] vorgeschlagen und genauer untersucht. Die Autoren empfehlen, nach zahlreichen Versuchen beim Verarbeiten der LFPTs zusätzlich zu den absoluten RSS-Werten die Differenz der RSS-Werte zwischen einzelnen an einem TRP empfangenen WLAN-APs aufzuzeichnen. Es wird gezeigt, dass hierdurch ein stabilerer Signalverlauf erreicht werden kann und das bei der Verwendung der SSD als LFPT eine wesentlich bessere Genauigkeit und

[3] Es hat sich jedoch bei Versuchen zu dieser Arbeit gezeigt, dass auch die Messwerte eines Geräts vom selben Typ Abweichungen aufweisen, welche aber nicht so stark sind wie diejenigen, die mit ME verschiedenen Typs auftreten.

Zuverlässigkeit des Gesamtsystem gegenüber den LFPTs mit absoluten RSS-Werten erreicht werden kann.

4.6 Algorithmen

Nachdem eingangs das grundlegende Prinzip beschrieben, danach auf die charakteristischen Merkmale von WLAN und GSM eingegangen und auch die Aufbereitung der Daten erklärt wurde, muss nun noch die Beschreibung der Algorithmen für das Vergleichen der in der TRP aufgezeichneten LFPTs mit den RTFPTs aus der RTP erfolgen, um das LFPT-Verfahren zu verstehen. Dieser Abschnitt stellt Algorithmen vor, die bei der Realisierung eines LFPT-ILS Anwendung finden können. Algorithmen zur Lokalisierung in LFPT-ILS können dabei in propabilistische und deterministische Algorithmen aufgeteilt werden. Dieser Abschnitt stellt die wichtigsten deterministischen Algorithmen für das LFPT-Verfahren vor und zeigt darüber hinaus noch einige Optimierungen auf, die dazu verwendet werden können, die Genauigkeit eines LFPT-ILS zu erhöhen. Propabilistische Algorithmen werden nur der Vollständigkeit halber referenziert und sind nicht im Detail beschrieben, weil sie in dieser Arbeit nicht verwendet wurden.

4.6.1 Determinitische Algorithmen

4.6.1.1 Nearest Neighbour in Signal Space (NNSS)

Der Nearest Neighbor in Signal Space (NNSS) Algorithmus basiert auf einem einfachen Nearest Neighbor (NN) Algorithmus. Das Ziel ist die Bestimmung der aktuellen Position eines MEs anhand des nächsten Nachbarn aus der Menge von Referenzdaten. Es wird bei diesem Algorithmus immer genau der Nachbar ausgewählt, welcher die größte Ähnlichkeit zur aktuell durchgeführten Messung (RTFPT) aufweist. Zur Bestimmung der Ähnlichkeit können verschiedene Metriken verwendet werden. Eine Metrik kann zum Beispiel ein einfaches Distanzmaß für den Abstand der aktuellen Messung von den Referenzdaten in Form des euklidischen Abstands sein. Ermittelt wird vom Algorithmus letztendlich genau der LFPT aus der Menge der Referenzdaten, letztlich also aus der RM, der den kleinsten euklidischen Abstand aufweist und somit der unmittelbare Nachbar der aktuellen Messung ist.

4.6.1.2 K Nearest Neighbour in Signal Space (KNNSS)

Ein auf NNSS aufbauender Algorithmus ist der k-Nearest Neighbor in Signal Space (k-NNSS) Algorithmus. Dabei wird nicht der nächste Nachbar als mögliche Position ausgewählt, sondern die k nächsten Nachbarn der aktuellen Messung (RTFPT). Es werden also genau die k LFPTs aus den Referenzdaten in der RM ermittelt, welche die größte Ähnlichkeit mit der aktuellen Messung aufweisen. Zur eigentlichen Lokalisierung werden dann die Positionen der nächsten k Nachbarn gemittelt. Hierzu kann zum Beispiel das in Kapitel 3.5.1 beschriebene Laterationsverfahren zum Einsatz kommen. Die endgültige Berechnung der Position erhöht die Genauigkeit zur Positionsbestimmung deutlich und wird sehr anschaulich gezeigt in [M. 07, SK08].

4.6.1.3 Euklidean-Distance

Die Euklidean-Distance ist verallgemeinert der Abstand zweier Punkte und ist definiert als:

$$d_E(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Ein Spezialfall der Euklidean-Distance mit $n = 2$ ist zum Beispiel die Berechnung eines Abstands im kartesischen Raum und entspricht damit dem Satz des Pythagoras. Da beim LFPT Verfahren aber oft mehr als nur zwei APs pro TP empfangen werden können, muss die Distanz zwischen einem LFPT und einem RTFPT deshalb im n-dimensionalen Raum bestimmt werden. Hierzu werden in der RTP die Abstände des aufgenommenen RTFPT zu allen in der RM gespeicherten LFPTs bestimmt. In einer n-dimensionalen Messreihe eines LFPTs mit den fünf APs $x = (x_{AP1}, x_{AP2}, x_{AP3}, x_{AP4}, x_{AP5})$ würde der Abstand zu einem in der RTP aufgenommenen RTFPT, der ebenfalls diese fünf APs $y = (y_{AP1}, y_{AP2}, y_{AP3}, y_{AP4}, y_{AP5})$ empfangen hat, bestimmt durch:

$$d_E(x,y) = \sqrt{\sum_{i=1}^{5}(x_{APi} - y_{APi})^2}$$

$$= \sqrt{(x_{AP1} - y_{AP1})^2 + (x_{AP2} - y_{AP2})^2 + (x_{AP3} - y_{AP3})^2 + (x_{AP4} - y_{AP4})^2 + (x_{AP5} - y_{AP5})^2}$$

Wie das Beispiel zeigt, muss der Abstand nicht in Metern gegeben sein, sondern wie im Falle des LFPT-Verfahrens ist auch ein relatives Distanzmaß in Form der absoluten RSS-Werte möglich. Euklidean-Distance ist die Basis von [BP00] und findet außerdem in [YZN07], [ZYN08] und vielen anderen Systemen Verwendung; so auch im SSF (siehe Kapitel 5).

4.6.1.4 Mahalanobis Distance

Der Mahalanobis Abstand nach [Mah36] ist ähnlich der Euklidean-Distance und hilfreich, den Abstand – und damit die Ähnlichkeit – einer unbekannten Probe zu einer bekannten Probe zu bestimmen. Der Unterschied dabei ist, dass der Mahalanobis Abstand im Vergleich zur Euklidian-Distance gewichtet ist. Bildlich kann man sich das so vorstellen, als seien alle Punkte gleicher Mahalanobis-Distanz von einem Zentrum im zweidimensionalen Raum eine gedrehte und verzerrte Ellipse, während es bei der Euklidean-Distance ein Kreis ist. Die Mahalanobis-Distanz ist definiert als:

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}$$

Mehr Details zur genauen Berechnung der Mahalanobis-Distance findet sich in Mahalanobis et al. [Mah36] und wird im Rahmen einiger LFPT Verfahren ebenfalls zum Auffinden des nächsten bekannten LFPTs verwendet. Mahalanobis-Distance findet in verschiedenen Systemen wie zum Beispiel [YZN07] und [ZYN08] Anwendung.

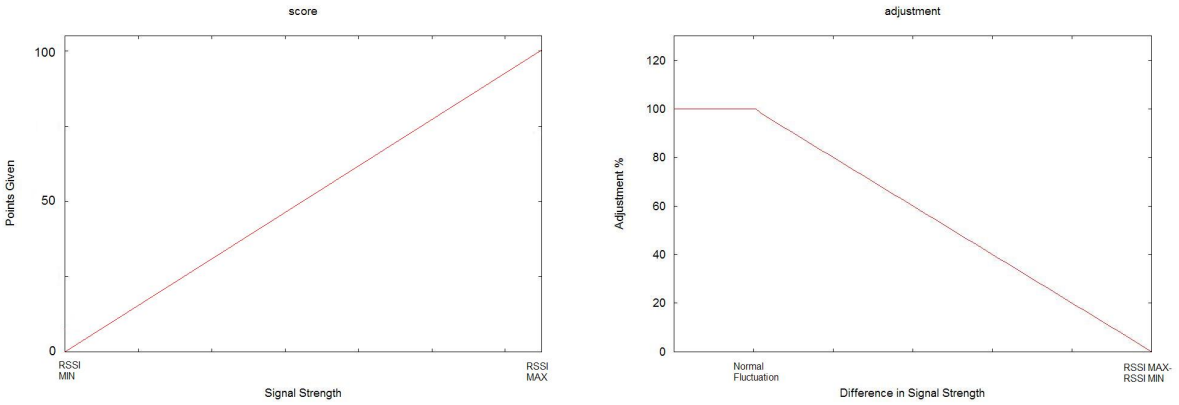
4.6.1.5 Composed Distance

Hat man zwei unabhängige Datensätze, wie zum Beispiel WLAN und GSM LFPTs, kann man sowohl die Mahalanobis-Distance d_M als auch die Euklidean-Distance d_E eines RTFPT zu allen LFPTs in der RM bestimmen. Sind die berechneten LFPTs mit den kleinsten Distanzen d_M und d_E nicht identisch, nimmt die Composed Distance den Abstand von d_M und d_E und mittelt die Position [ZYN08].

4.6.1.6 Similarity matching algorithm (SMA)

Philips et al. [PKL07] stellen zur genaueren Positionsbestimmung den Similarity-Algorithmus vor. Der SMA führt die zwei neuen Größen *Score* und *Adjust* ein, welche dazu dienen sollen, eine Ähnlichkeitsmetrik zu beschreiben. Hierzu summiert der Algorithmus als erstes mit Hilfe der Scoring Funktion alle Messwerte auf, um eine Metrik für den gemessenen RTFPT zu bekommen $sum_x = sum_x + score(r_x d)$, mit RTFPT x und RSS Werten $r_x d$. Danach wird über alle in der RM vorhandenen LFPT iteriert und ebenfalls die Summe aller Messwerte sum_l gebildet. Hierbei wird nicht nur ein Scoring sondern auch ein *Adjusting* der Werte vorgenommen, bevor sie aufsummiert werden.

Um nun die Ähnlichkeit zwischen dem gemessenen RTFPT und den LFPTs der RM festzustellen, wird eine $ratio_l = \frac{sum_l}{sum_x}$ gebildet, wobei ein höherer *ratio* Wert für ein besseres Match steht. Schließlich wird mit einer *similiarity* Funktion die eigentliche Ähnlichkeit bestimmt. Dies ist auch die Metrik, welche vom Algorithmus zurückgegeben wird und letztendlich dazu verwendet wird, die Position zu bestimmen.



(a) Die Scoring-Funktion

(b) Die Adusting-Funktion

Abbildung 4.7: Die zwei grundlegenden Funktionen des Similarity matching algorithm (SMA) nach [PKL07].

4.6.1.6.1 Scoring Die verwendete *Scoring*-Funktion ist Teil des verwendeten Algorithmus und dient dazu, RSS Werte mit einer Punktzahl zu versehen. Die Scoring-Funktion ist wie in Abbildung 4.7(a) gezeigt, eine lineare Funktion über dem RSS-Wertebereich, welche jedem RSS Wert eine Punktzahl zuweist.

Nimmt man an einem RP im Raum RTFPTs zu zwei verschiedenen Zeitpunkten x und x' auf, so kann es vorkommen, wenn ein RP sehr weit von einem AP entfernt ist, dass dieser AP nicht

bei jeder Messung sichtbar ist. Dies führt zu Problemen bei der Lokalisierung. Deshalb werden bei dem verwendeten Algorithmus nicht vorhandene APs mit der Konstante *LOW* belegt, eine der MIN_{rss} entsprechende negative Zahl, die von der Scoring-Funktion immer durch eine Score von 0 abgebildet wird. Dadurch, dass aber ein solcher AP im Normalfall sowieso eine niedrige RSS im LFPT aufweist und somit sowieso eine niedrige Score erhalten hätte, fällt das Fehlen des APs nicht weiter ins Gewicht. Genau dasselbe gilt, wenn der AP im RTFPT überhaupt nicht existiert, weil man davon ausgehen kann, dass durch das Scoring der AP durch sein schwaches Signal eine so niedrige Score hat, das es nicht signifikant für die Lokalisierung ist.

4.6.1.6.2 Adjusting Löst das Scoring nur das Problem der fehlenden APs bei einer Messung, kümmert sich das Adjusting um die Fluktuationen von Messwerten. Wie beim Scoring wird auch hier eine Funktion verwendet, welche das Adjusting, die Anpassung in Prozent zurückgibt. Jeder AP hat eine eigene Varianz bezüglich seiner RSS Werte. Die Adjust-Funktion ist so gewählt, dass sie bis zu einem Schwellenwert (*fluctation threshold*) immer 100 % zurückgibt. Wird der Schwellenwert überschritten, nimmt die Ungenauigkeit zu. Die Wahrscheinlichkeit nimmt in der Adjusting-Funktion linear mit der Größe der Differenz der beiden zu vergleichenden RSS-Werte ab. Das heißt: je größer die Distanz, desto weniger relevant das Ergebnis, desto stärker müssen die im ersten Teil gemessenen Score-Werte angepasst werden. So würde im Extremfall die Adjusting-Funktion 0 liefern, wenn der Abstand der beiden Messergebnisse minimal ist und die Score somit nicht ins Gewicht fallen. Abbildung 4.7(b) zeigt diese Zusammenhänge sehr schön auf.

4.6.1.7 Orientation Reduction

Ein weiterer interessanter Ansatz, die Orientation Reduction wird von King et al. [KKH+06] beschrieben. Ist die Orientierung eines Nutzers bekannt, können anhand des gemessenen Winkels von vornherein alle LFPTs der RM ausgeschlossen werden, die nicht dieselbe Orientierung haben wie die der aktuell gemessenen RTFPT. Dies wird von King et al. über Zustände realisiert. Für einen Winkel von $\alpha = 69^\circ$ hat das System minimale Fehlerdistanz. Dies kann dadurch begründet werden, dass für kleine α die Anzahl der auszuwählenden Histogramme des probalistischen Modells ≤ 1 sind. Auf der anderen Seite muss man bei zu großem Threshold α aufpassen, nicht wieder auf dieselben Genauigkeiten wie ohne die zusätzlichen Aufnahmen zurückzufallen, was im Hinblick auf den Messaufwand, der zum Aufbau des Wahrscheinlichkeitsmodells von Nöten ist, letztendlich nur Zeitverschwendung wäre.

4.6.2 Probabilistische Algorithmen

Neben den vorgestellten deterministischen Algorithmen gibt es noch probabilistische Ansätze. Diese Algorithmen werden hier aber nur mit ihren Referenzen aufgeführt und nicht näher beschrieben, da es den Rahmen dieser Arbeit gesprengt hätte, diese zu implementieren und mit den deterministischen zu vergleichen. Es handelt sich dabei um:

- Bayesian interference [LBR+05], [M. 07]
- Hidden Markov Model (HHM) [LBR+05]
- Markov Lokalization [LBR+05]
- Monte Carlo Lokalisation [LBR+05]

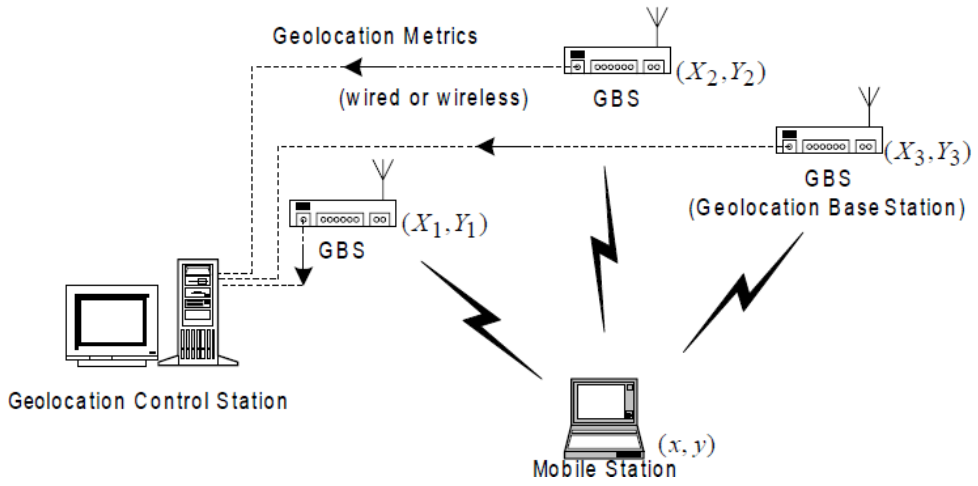


Abbildung 4.8: Architektur eines netzwerkbasierten ILS [PLY+00]

- Voronoi-Diagramme/Proximity-Graphen [SK08].

4.7 Architektur

Die Architektur eines ILS ist von großer Bedeutung, sie bildet das Rückgrat eines jeden ILS. Eine Architektur besteht immer aus mehreren Komponenten. So wird in jeder Architektur eines oder mehrere ME verwendet, die lokalisiert werden sollen. Zusätzlich werden in jeder Architektur Infrastrukturkomponenten verwendet, von denen das ME umgeben ist. Bei diesen externen Infrastrukturkomponenten kann es sich je nach verwendeter Technologie zum Beispiel um GSM-BSs, WLAN APs oder auch visuelle Tags wie QR-Codes handeln. Die Wahl einer geeigneten Architektur zum Aufbau eines ILS hat große Bedeutung. So haben die grundlegenden Architekturentscheidungen starke Auswirkungen auf die allgemeinen Systemparameter wie Genauigkeit, Zuverlässigkeit und Skalierbarkeit eines ILS. Eine Architektur sollte deshalb immer auf die Anforderungen, denen ein ILS beim Einsatz in der Praxis ausgesetzt ist, ausgerichtet werden. Es existieren zwei grundlegende Architekturen, die Network-Based-Architecture (NBA) und die Mobile-Based-Architecture (MBA). Eine NBA setzt auf den Aufbau einer externen Infrastruktur zum Aufbau eines ILS und lagert sowohl das Messen der Sensordaten als auch die Lokaliserung einer ME auf einem externen Netzelement aus. Das ME wird nur zum Anzeigen der aktuellen Position verwendet. Die zweite Architektur, die sogenannte MBA, ist ein fast schon diametraler Ansatz zur NBA und ist um das ME Endgerät zentriert. Das Messen der Sensordaten und die Lokaliserung wird direkt auf dem ME durchgeführt.

Zur Durchführung einer Lokalisierung verwenden beide Architekturen jeweils externe Infrastrukturkomponenten, wie GSM-BSs oder WLAN-APs. Diese Komponenten bestehen nur aus der Hardware, die die Funksignale senden und empfangen kann. Da aber in vielen Fällen in der Praxis schon Infrastrukturkomponenten vorhanden sind, ist nicht nur eine aktive, sondern auch eine passive Nutzung dieser Komponenten möglich. Deshalb unterscheidet diese Arbeit in aktive und passive Infrastruktur.

4.7.1 Network-Based-Architecture (NBA)

Eine NBA ist der aus der Informatik bekannten Thin-Client-Architektur sehr ähnlich. Bei einer Thin-Client-Architektur wird ein ME nur für die Darstellung vorberechneter Daten verwendet. Die Berechnung und Verwaltung der Daten wird von einem in der Hierarchie übergeordneten Netzelement der Infrastruktur durchgeführt. Je nach Anwendungsszenario fällt die Komplexität der ME höher oder niedriger aus. In der Regel dient das ME in einer NBA der Anzeige der aktuellen Position und zur reinen Messung von Sensordaten.

Eine netzwerkbasierte ILS Architektur besteht aus drei Komponenten: Eine Komponente zur Messung der Feldstärken beziehungsweise der Signallaufzeiten, eine zweite Komponente, welche anhand der gemessenen Daten mit Hilfe der bereits vorgestellten Lokalisierungsalgorithmen die aktuelle Position berechnet, und eine Präsentationskomponente zum Anzeigen der aktuellen Position. Pahlavan et al. [PLY+00] haben hierzu in ihrer Arbeit zum Aufbau des in Abbildung 4.8 dargestellten WLAN-ILS folgende NBA gewählt: Eine Geolocation Basisstationen (GBS) führt Feldstärkemessungen (RSS) zu allen ihr bekannten MEs durch und übertragt diese an eine Geolocation Control Station (GCS)[4], wo sie mit den Messwerten der anderen GBS die Position des ME ermittelt und diese an das ME überträgt. Bei diesem Ansatz befinden sich bis auf die Präsentationskomponente alle Komponenten in der Infrastruktur des ILS. Das ME Endgerät dient lediglich dem Anzeigen der aktuellen Position. Die Komponenten zur Datenerhebung und Berechnung der aktuellen Position sind Teil der umgebenden Infrastruktur. Eine Variation dieser NBA besteht darin, nur die Komponente zur Positionsbestimmung in die Infrastruktur auszulagern, die Komponente zur Datenerhebung aber auf dem ME ausführen zu lassen und die Daten anschließend zur Lokalisierung an ein Netzelement, wie zum Beispiel einen Server, übertragen zu lassen.

Denkbar ist auch ein kombinierter Ansatz wie er in [YZN07] beschrieben wird. Hier findet die Datenerhebung sowohl auf dem ME als auch in einer Infrastrukturkomponente, zum Beispiel einem WLAN-AP, statt. Der Grund ist, dass in der Arbeit festgestellt wurde, dass die Feldstärke einer Übertragung vom WLAN-AP zum ME (downlink) und das Signal von einem ME zum AP (uplink) voneinander abweichen können. Dies gilt auch für gleiche Sendeleistung bei AP und ME. Die Arbeit kommt weiterhin zum Ergebnis, dass die Verwendung von Daten beider Endpunkte die Genauigkeit um 20 % bis 30 % erhöht. Die gemessenen Feldstärken werden dann an einen zentralen Data Storage Server gesendet, welcher eine Korrelation der Daten und die eigentliche Lokalisierung des ME durchführt.

Eine NBA lagert also große Teile der Komplexität in Komponenten der Infrastruktur aus. Die Anforderungen an die MEs werden möglichst gering gehalten. Die Vorteile der geringen Komplexität der ME sind geringe Hardwarekosten bei der Anschaffung der ME und eine Steigerung der Einsatzzeiten der MEs, weil aufgrund der Tatsache, dass keinerlei komplexe Algorithmen auf der ME ausgeführt werden, es zu einem signifikant geringeren Stromverbrauch auf Seiten des MEs kommt. Auch die eigentliche Lokalisierung kann von einem netzwerkbasierten ILS wesentlich genauer und schneller durchgeführt werden. Durch die hohe Rechenleistung der Server können auch komplexere Algorithmen als die im Verlauf des Kapitels vorgestellten angewendet werden. Um die Genauigkeit des ILS zusätzlich noch zu verbessern, kann in einem netzwerkbasierten ILS eine Folge von Algorithmen auf die gemessenen Daten angewendet werden. Bei diesem Ansatz können mehrere unabhängige Algorithmen zur Lokalisierung

[4] Die Verbindung zwischen GBS und GCS kann entweder kabelgebunden oder drahtlos sein.

nacheinander angewendet werden. Zusätzlich kann dann zur Berechnung der endgültigen Position der Durchschnitt aller berechneten Positionen gebildet werden. Dieser Ansatz ist ähnlich dem in [BP00] vorgestellten *k*-Nearest Neighbors Algorithmus und kann zu einer erheblichen Steigerung bei der Genauigkeit führen.

Nachteile der netzwerkbasierten Architektur sind die Kosten beim Errichten einer eigenen ILS-Infrastruktur, da diese leistungsstarke und vor allem hoch redundante Hardware benötigt. Die Redundanz ist von großer Bedeutung, da der netzwerkbasierte Ansatz eine starke Zentralisierung des gesamten ILS mit sich bringt. Fällt der zentrale Server zur Lokalisierung aus, kann keine Lokalisierung mehr durchgeführt werden. Dies gilt auch für den Ausfall einer GBS, wobei hier nur in einem bestimmten Bereich keine Lokalisierung mehr durchgeführt werden kann. Ein weiterer Nachteil ist die Datensicherheit in einem zentralisierten ILS. Da die Endgeräte zum Beispiel in einem WLAN basierten ILS über ihre MAC-Adresse eindeutig identifizierbar sind, können Nutzer des ILS permanent überwacht werden.

4.7.2 Mobile-based-Architecture (MBA)

Die MBA ist eine um das ME zentrierte Architektur. Die Datenerhebung, der Algorithmus zur Lokalisierung und die Anzeige der aktuellen Position werden direkt auf dem ME ausgeführt. Um eine Lokalisierung durchzuführen, werden aber auch bei der MBA Komponenten wie BSs beziehungsweise APs benötigt, zu denen die Feldstärken oder Laufzeiten gemessen werden können. Um den Unterschied zur NBA nochmals zu verdeutlichen: eine MBA kommt ebenfalls nicht ohne Komponenten wie APs oder BSs in der Infrastruktur aus, allerdings werden das Messen der Daten sowie die Lokaliserung komplett auf dem ME durchgeführt. Dabei muss bei der in der MBA verwendeten Infrastruktur zwischen aktiver und passiver Infrastruktur unterschieden werden. Die Begriffe aktive und passive Infrastruktur werden im Folgenden erläutert und werden im weiteren Verlauf der Arbeit nach diesen Definitionen verwendet.

4.8 Aktive und passive Infrastruktur

Die Begriffe aktive und passive Infrastruktur beschreiben die Art der Nutzung der externen Infrastrukturkomponenten. Bei der Verwendung einer aktiven Infrastruktur werden beim Aufbau eines ILS gezielt Komponenten aufgestellt, die später zur Lokalisierung verwendet werden können. Bei der Verwendung einer passiven Infrastruktur wird hingegen keinerlei eigene externe Infrastruktur errichtet, sondern lediglich bereits vorhandene Infrastruktur verwendet.

Die aktive Infrastruktur setzt auf den Aufbau einer eigenen externen Infrastruktur zur Lokaliserung von MEs. Je nach verwendeter Lokalisierungstechnologie werden gezielt WLAN-APs, GSM-BS, oder visuelle Tags (QR-Codes) in der Infrastruktur verteilt. Die Standorte der Komponenten werden in einem solchen System so gewählt, dass die Anforderungen an das ILS möglichst genau erfüllt werden.

Einen komplementären Ansatz verfolgt die Verwendung einer passiven Infrastruktur zum Aufbau eines ILS. Im Gegensatz zu der beschriebenen aktiven Infrastruktur werden keinerlei eigene Infrastruktur zur Lokalisierung benötigt. Es wird ganz im Gegenteil versucht, ein ILS mit bereits vorhandenen Infrastrukturkomponenten aufzubauen und anhand passiver Messungen diese zur Lokalisierung zu verwenden. Das Interessante an diesem Ansatz ist, dass der

Umstand ausgenutzt wird, dass wir in der heutigen Zeit fast immer von Infrastruktur, die zu einer Lokalisierung verwendet werden kann, umgeben sind. Als Beispiele seien hier unter anderen WLAN, GSM, Bluetooth und visuelle Tags genannt. Dies gilt vor allem im städtischen Bereich, wo neben der ubiquitären Verfügbarkeit von GSM Netzen auch eine flächendeckende Abdeckung mit WLAN vorhanden ist, die zu einer Lokalisierung verwendet werden kann. Passiv bedeutet in diesem Zusammenhang auch, dass nicht zwingend eine aktive Verbindung zu einem Netz bestehen muss. Dies kann sehr schön am Beispiel der WLAN-Technologie gezeigt werden, welche sich sowohl im geschäftlichen als auch im privaten Umfeld zu einem Technologiestandard entwickelt hat, der nahezu überall dazu eingesetzt wird, sich mit dem Internet zu verbinden. Eine Folge hiervon ist, dass sowohl Firmen wie auch private Haushalte heutzutage in den meisten Fällen über ein eigenes WLAN Netz verfügen. Anders ausgedrückt, man verwendet bei dem passiven Ansatz nicht nur das eigene WLAN, sondern auch das WLAN Netz seines Nachbarn beziehungsweise alle WLAN Netze, die durch einen WLAN-Scan auffindbar sind. Die Infrastruktur wird also sozusagen von vielen kleinen Helfern errichtet und das ILS benutzt diese Infrastruktur passiv. Ein anderes Beispiel sind die erwähnten GSM Netze. Die hohen Wachstumsraten im Mobilfunkbereich haben nicht nur dazu geführt, dass nahezu jeder erwachsene Mensch ein eigenes Mobiltelefon besitzt, sondern auch dazu, dass bis auf wenige Gebiete von einer ubiquitären Verfügbarkeit von GSM/3G Netzen ausgegangen werden kann. Dieser Umstand kann zunutze gemacht werden, indem die Feldstärken zu den BSs passiv gemessen werden. Die genaue Vorgehensweise wurde in Kapitel 4.4.2 beschrieben. Eine andere Art der Verwendung passiver Infrastruktur zur Lokalisierung kann aber auch kamerabasiert arbeiten. Es ist zum Beispiel denkbar, gezielt nach visuellen Markern in der angestrebten ILS Umgebung zu suchen. So werden in heutigen Großraumbüros oder auch in den immer beliebteren CoWorking-Spaces[5] die einzelnen Arbeitsplätze nicht mehr fest vergeben sondern zufällig besetzt. Um festzustellen, wer wann an welchem Arbeitsplatz gearbeitet hat, sind diese mit einer eindeutigen ID Nummer versehen. Es wäre auch denkbar, dass hierbei QR-Codes zum Einsatz kommen. Relevant in Bezug auf ein markerbasiertes ILS ist, dass diese ID Nummer als passive Marker verwendet und mithilfe der vorgestellten OCR oder anderen Image Recognition Technologien ausgelesen werden können. Eine weitere Möglichkeit wäre zum Beispiel die Verwendung von Piktogrammen oder Raumnummern als passive Marker. Die Verwendung passiver Infrastrukturen kann also sehr kreativ sein. Das Schöne an dieser Technik ist, dass sie in den meisten Fällen völlig kostenlos genutzt werden kann.

Es lässt sich feststellen, dass die Ansätze zur Verwendung aktiver beziehungsweise passiver Infrastruktur ihre Vor- und Nachteile haben. Wichtig ist die Erkenntnis, dass die Entscheidung letztendlich von den Anforderungen und deren Auswirkungen auf die allgemeinen Systemparameter des ILS abhängt. So kann festgestellt werden, dass bei einer Genauigkeit im Meterbereich in Verbindung mit einer hohen Zuverlässigkeit die Installation einer aktiven Infrastruktur sinnvoll ist. Jedoch geht dies mit wesentlich höheren Kosten bei der Installation des ILS einher und erhöht die Komplexität des Systems deutlich. Im Gegensatz hierzu ist die Verwendung einer passiven Infrastruktur weitaus weniger komplex und ist, vor allem was die Kosten der Installation angeht, der aktiven Infrastruktur vorzuziehen. Je nach Anwendungsfall muss die Verwendung von passiver Infrastruktur nicht zwingend zu einer verringerten Genauigkeit und Zuverlässigkeit führen.

[5] Das Prinzip des CoWorkings kommt aus den USA und bietet Freiberuflern, die an mehreren Projekten oder Startups gleichzeitig arbeiten die Möglichkeit, flexibel und je nach Auftragslage Arbeitsplätze in den CoWorking Räumlichkeiten anzumieten.

4.9 Allgemeine Systemparameter von LFPT-ILS

Bei dem Entwurf eines FPT basierten ILS müssen mehrere Aspekte betrachtet werden. Deshalb müssen beim Systemdesign zuerst die grundlegenden Anforderungen des Systems in Systemparameter abgebildet werden. Dies ist essentiell, da die Systemparameter Abhängigkeiten aufweisen und Auswirkungen auf die nicht funktionellen Eigenschaften wie Komplexität, Kosten, Erweiterbarkeit und Skalierbarkeit haben.

4.9.1 Größe der Installation

Die Fläche der angestrebten Installation muss festgelegt werden, da sie Auswirkungen auf die Genauigkeit des Systems hat. Insbesondere hängt die Granularität des Rasters von den Ausmaßen der Fläche ab, da bei einem engen Raster und einer großen Installation viel mehr TRP initial zu erfassen sind, wie bei einem gröberen Raster in derselben Installation.

4.9.2 Raster (Grid Spacing)

Um die Positionen der TP, welche in der TRP erfasst werden festzulegen, empfiehlt es sich, ein gleichmäßiges Raster über dem Grundriss der zu vermessenden Fläche aufzuspannen. Dabei ist zunächst darauf zu achten, dass die TPs später an Stellen in zugänglichen Bereichen liegen. Von Nachteil wäre es zum Beispiel, wenn durch das Raster in manchen Räumen kein TP, in anderen dafür mehrere TPs liegen. Es sei denn, dies ist gewollt, um zum Beispiel in einem Raum später eine bessere Genauigkeit erzielen zu können. Je nach Grundriss kann es auch vorkommen, dass manche TPs im Raster trotzdem nicht vermessen werden können, da sie unzugänglich sind. Das ist aber nicht weiter schlimm, da in diesen Bereichen später in der Regel auch keine Lokalisierung durchgeführt werden muss.

Ein Raster wird folgendermaßen erstellt: Ein quadratischer Raum der Länge L hat eine Fläche von $L \cdot L = L^2$. Bei einem Raster mit Abstand 1 Meter ergeben sich also L^2 TPs. Bei einem Raster von 5 Metern ergeben sich nur $\frac{1}{5} \cdot L^2$ verschiedene TPs, die bestimmt werden müssen. Die Position eines TP auf dem Raster hat eine eindeutige 2-D-Koordinate der Form (x,y). Zusätzlich kann dies zum Beispiel noch um die 3. Dimension des Stockwerks durch z erweitert werden zu (x,y,z) [SK08]. Dabei ist der Abstand der TPs auf der z-Achse fest durch die Stockwerkhöhe vorgegeben, während das Raster in x- und y-Richtung frei gewählt werden kann. Die Datenformate werden im Detail in Kapitel 4.9.5 beschrieben.

4.9.3 Anzahl der TP

Die Anzahl der TPs, an denen gemessen werden soll, hängt stark von den Anforderungen eines ILS ab und hat direkten Einfluss auf die spätere Genauigkeit und Zuverlässigkeit des Systems. Bei LFPT basierten Systemen gilt grundsätzlich, je mehr Orte in der Trainingsphase vermessen werden, desto höher ist die Genauigkeit des Gesamtsystems. In den Messungen zu dieser Arbeit wurde jedoch festgestellt, dass ab einem bestimmten Mindestabstand die Genauigkeit aufgrund der Zuverlässigkeit eines LFPT-ILS nicht mehr zu steigern ist. Grund hierfür ist, dass sich oft ganze Cluster von Werten ergeben, die sich gut unterscheiden lassen. Auf dieser Basis sollte auch letztendlich die Granularität der LFPTs und somit die Anzahl der Positionen, an denen man Messungen durchführt, ausgerichtet werden. Dies ist vor allem dann wichtig,

wenn eine hohe Standardabweichung vorherrscht. Eine feine Granularität bei hoher Standardabweichung führt zu hohen Fehlerraten bei der Positionsbestimmung und bringt keine Vorteile. Die Granularität sollte letztendlich an den erkennbaren Clustern festgemacht werden, um eine hohe Genauigkeit zu garantieren. Selbstverständlich ist es oft nicht einfach, die Menge der gewünschten TP mit den möglichen TP, welche das Raster vorgibt, in Einklang zu bringen. Es gibt Gebäude, für die die theoretisch optimale Verteilung der TP nicht möglich ist. Die Verteilung muss dann auf ein machbares Konzept angeglichen werden.

4.9.4 Anzahl der APs

Ein weiterer Parameter mit Auswirkung auf die Genauigkeit und auch die Skalierbarkeit ist die Anzahl der APs. Grundsätzlich kann ein LFPT umso genauer und aussagekräftiger sein, je mehr APs er erfasst. Insbesondere dann, wenn an manchen Stellen der Installation ein AP gerade noch empfangbar, an anderen aber gar nicht mehr zu sehen ist. Bei der Menge der APs kommt es aber auch auf ihre Verteilung an. Fünf APs, welche alle direkt beieinander installiert sind, erhöhen die Genauigkeit nicht so gut wie fünf APs, welche über die gesamte Installation verteilt sind. Prinzipiell kann man also sagen, je mehr APs mit jeweils möglichst größtem Abstand zueinander vorhanden sind, desto unterschiedlichere und wertvollere LFPTs lassen sich erstellen. Andererseits muss man berücksichtigen, dass der Rechenaufwand, die einzelnen LFPTs in der RM mit dem jeweiligen RTFPT zu vergleichen, sich erhöht, je mehr APs an einem RP empfangen werden. Sollte man also Einfluss auf die Menge der APs haben, zum Beispiel, wenn man auf eine aktive Infrastruktur setzt, sollte man nur genau so viele APs aufstellen, dass alle LFPTs eindeutig unterschieden werden können. Hierduch kann der Rechenaufwand in der RTP verringert und die Geschwindigkeit des ILS erhöht werden. Hat man, wie bei der Verwendung eines Systems, das eine rein passive Infrastruktur nutzt, keinen Einfluss auf die Menge der APs, könnte man dazu übergehen, bestimmte APs zu ignorieren. Zum Beispiel solche, deren absolute RSS-Werte zu Nahe beieinander liegen oder eine zu hohe Standardabweichung aufweisen und dies wenig zur Genauigkeit beitragen würde.

4.9.5 Datenformat

Das verwendete Datenformat zur Beschreibung von Positionen innerhalb eines ILS kann auf verschiedene Arten realisiert werden. Eine Position innerhalb eines ILS wird in einem LFPT basierten System immer als eine Funktion der gemessenen Feldstärken an einem bestimmten Punkt im Raum angesehen. Wie bereits erwähnt, wird davon ausgegangen, dass jeder Punkt im Raum seine eigene charakteristische Signatur an gemessenen Feldstärken aufweist [PLM02]. Ein LFPT ist somit immer an eine bestimmte Position im Raum gebunden und kodiert diese anhand der aufgenommenen Feldstärke-Tupel. Battiti et al. [BBV02] weisen zurecht darauf hin, dass es keine eindeutige Abhängigkeit der physikalischen Positionen in Hinblick auf die gemessene Feldstärke gibt. Der Zusammenhang ist nur statistischen Ursprungs. Die Folge ist, dass es keine eindeutig anerkannte Notation für ein Format gibt, welches Indoor-Geokoordinaten beschreibt.

Die meisten in der Literatur beschriebenen Systeme verwenden deshalb ein eigenes Format zur Repräsentation einer Indoor-Koordinate. Die Arbeiten von Bahl et al., Battiti et al. und Swangmuang et al. [BP00, BBV02, SK08] verwenden ein d-Tupel von kartesischen Koordinaten[6] zur Kodierung von Indoor-Geokoordinaten, Battiti et al. [BBV02] verwenden zusätzlich

[6]Kartesische Koordinaten dienen in der Mathematik der Positionansangabe von Punkten im Raum. Sie eignen

noch einen Ansatz, der als Indicator Variable bezeichnet wird und grob beschreibt, ob sich ein Objekt innerhalb oder außerhalb eines bestimmten Bereichs, bei grober Granularität zum Beispiel auf Raumbasis, befindet $L = \{-1, 1\}$. Philips et al. und Bolinger et al. [PKL07, BPCL09] nutzen als Format zur Beschreibung von Positionen dagegen nur symbolische Identifier. Es können auch eine oder mehrere der Ansätze kombiniert verwendet werden. Dies ist im Falle des in dieser Arbeit beschrieben SSF gegeben, welches sowohl ein auf kartesischen Koordinaten basierendes Format besitzt, als auch symbolische Identifier zur Beschreibung einer physikalischen Position nutzt.

4.9.5.1 d-Tupel Format

Die Verwendung von d-Tupel zur Beschreibung von Koordinaten eignet sich sehr gut zur Beschreibung von physikalischen Positionen innerhalb eines ILS. Sie sind vor allem dann gut geeignet, wenn die Trainingspunkte eines ILS sich an einem Raster ausrichten, welches durch das verwendete Koordinatensystem mathematisch beschrieben werden kann. Die Positionen der Trainingspunkte werden innerhalb des ILS meistens anhand eines kartesischen Koordinatensystems mit d-Dimensionen beschrieben. Wie viele Dimensionen verwendet werden, hängt von den Anforderung des ILS ab. Somit sollte je nach Anforderung ein an die Situation angepasstes d-Tupel Format verwendet werden. Für die meisten ILS, die sich nur auf ein Stockwerk beschränken, reicht eine einfache, planare Darstellung der Koordinaten im zweidimensionalen Raum völlig aus. Die Positionen können durch ein Tupel $L = \{x, y\} \mid x, y \in \mathbb{R}^2$ vollständig beschrieben werden. Soll das Format zusätzlich noch das Stockwerk und die Orientierung abdecken kann die Position als 4-Tupel (Quadrupel) $L = \{x, y, z, o\} \mid x, y, z \in \mathbb{R}^3$, mit $\{o \geq 0 \wedge o < 360\}$ beschrieben werden. Die Verwendung von Koordinaten zur Beschreibung einer physikalischen Position in Form von Tupeln wird von Battiti et al. [BBV02] übrigens als Rückführungsproblem beschrieben, weil jede Position, durch die ein LFPT kodiert wird, als Menge der Feldstärkemessungen zu verstehen ist und anhand dieser Rückschlüsse auf die Koordinaten gezogen werden können.

4.9.5.2 Symbolische Identifier (SID)

Zur Beschreibung einer physikalischen Position können neben Koordinaten mit d-Tupel Format auch symbolische Identifier (SID) zur Beschreibung einer Position innerhalb eines ILS genutzt werden. So können einzelne TPs in einem LFPT System einfach Identifier in Form von Strings zugewiesen werden. Diese können fiktiv, systematisch oder auch auf reale Objekte/Orte bezogen sein. Eine fiktive Zuordnung von SIDs könnte zum Beispiel allen Positionen innerhalb eines ILS Namen von Charakteren aus einem Spielfilm zuweisen[7]:

$$TP = \{'V', 'Evey', 'Finch', 'Game'\}$$

Bei einer größeren Installation mit einer großen Anzahl an Trainingspunkten skaliert diese Methode jedoch nicht mehr und es empfiehlt sich, eine systematische Benennung der SIDs zu verwenden. So kann die Anzahl der Trainingspunkte durch ein einfaches Durchnummerieren effizient umgesetzt werden:

sich deshalb auch zur Beschreibung des vorgestellten Rasters im n-dimensionalen Raum.

[7] So wurden bei ersten Tests mit SSF QR-Codes mit symbolischen Identifiern aus Namen des Films „V wie Vendetta“ erstellt und benutzt.

$$TP = \{SID_0, SID_1, ..., SID_n\}, n = Anzahl\ der\ Trainingspunkte$$

Bei Installationen größerer Granularität auf Raumbasis kann es dagegen auch sinnvoll sein, SIDs so zu wählen, dass diese einem Objekt oder einem Ort zugeordnet sind. So kann einer Position nicht nur ein SID zugeordnet werden, sondern über den Bezug zu einem realen Objekt kann diese Position auch noch mit semantischen Informationen versehen werden. Diese semantischen Informationen können von Nutzern besser verstanden werden. So kann ein Nutzer, der die LFPT ILS verwendet, seine Trainingspunkte nach Räumen oder Objekten benennen:

$$TP = \{'Küche', 'Esstisch', 'Wohnzimmer', 'Couch', 'Bad'\}$$

Dies erleichtert die Benutzung durch den Nutzer, wenn er zum Beispiel Regeln definieren möchte, dass bestimmte Dienste automatisiert ausgeführt werden, wenn sich der Nutzer an einem dieser durch die SID bezeichneten Orte befindet.

4.10 Fazit

Die Vorteile des LFPT Verfahrens sind, dass es sehr robust ist und von komplexen Umgebungen profitiert. Zudem kann es passiv eingesetzt werden, denn dank der heutigen flächendeckenden Funkabdeckung durch GSM und allgegenwärtiger WLAN Netze im innerstädtischen Bereich ist es nicht unbedingt notwendig, eine eigene Funk-Infrastruktur aufzubauen. Werden zusätzlich einfache Algorithmen wie die Bestimmung der Distanzen anhand der euklidischen Signaldistanz verwendet, kann die aktuelle Position auf dem ME berechnet werden, was eine hohe Sicherheit und Datenschutz gewährleistet. Auch denkbar ist die Berechnung der Position auf einem zentralen Server, was wiederum den Vorteil hat, mit komplexeren Algorithmen die von den MEs übermittelten Daten auszuwerten.

Letztendlich ist es wesentlich, pro Anwendungsfall abzuwägen, welche Technologien, Techniken und Methoden zur Datenverarbeitung, welche Algorithmen, Architektur und Infrastruktur für die Installation ausgewählt werden sollen, um den Anforderungen an das LFPT-ILS am besten gerecht zu werden. Anforderungen sind hierbei alle gewünschten, nichtfunktionalen Eigenschaften eines ILS wie Genauigkeit, Zuverlässigkeit, Erweiterbarkeit, Skalierbarkeit und Sicherheit.

Die Nachteile des LFPT liegen in seiner geringeren Genauigkeit im Vergleich zu den laufzeit- und markerbasierten Verfahren, die im vorherigen Kapitel vorgestellt wurden. Die Genauigkeit beträgt je nach Umgebung in etwa 2 bis 5 Meter und hängt im Wesentlichen von der für die TRP gewählten Rastergröße ab.

5 Smartspace Framework

Das Smartspace Framework (SSF) ist ein ILS für die Google Android Plattform, welches im Rahmen der Diplomarbeit konzipiert und implementiert wurde. Das SSF soll Entwicklern eine einfache Möglichkeit bieten, auf ME der Android Plattform Indoor-Lokalisierungsdienste in ihren Applikationen verwenden zu können.

Dieses Kapitel gibt einen Überblick über die dem SSF zugrundeliegenden Konzepte und untersucht die Potentiale der Android Plattform zur Indoor-Lokalisierung. Im ersten Teil des Kapitels werden die Beweggründe und das Konzept zur Erstellung des SSF erläutert. Der zweite Teil des Kapitels befasst sich anschließend mit den aus der Konzeption resultierenden allgemeinen Anforderungen. Zum Abschluss des Kapitels wird dann das theoretische Modell des SSF vorgestellt und anhand der Anforderungen die Vor- und Nachteile sowie die Eignung der einzelnen Sensoren im Hinblick auf die Realisierung einer adäquaten Indoor-Lokalisierung auf Basis von Google Android diskutiert.

5.1 Konzept

Bei der Konzeption des SSF stand vor allem der Gedanke im Vordergrund, wie ein ILS wohl aussehen würde, das jeder Nutzer eines Android basierten ME zu Hause oder an einem Ort seiner Wahl installieren und einsetzen kann. Dies ist deshalb von so großer Bedeutung, weil wir uns einen Großteil unseres täglichen Lebens in geschlossenen Räumen aufhalten und es angesichts der aktuellen Entwicklungen gerade der ortsbezogenen Dienste immer wichtiger wird, die Position eines ME und somit den Kontext seines Nutzers zu kennen. Leider gibt es – wie schon erwähnt – momentan keine ubiquitär verfügbare Lösung für dieses Problem.

Der Aufbau eines privaten ILS ist ein Szenario, welches vor einigen Jahren noch undenkbar war. MEs waren vorrangig zum Telefonieren geeignet und andere Formfaktoren wie PDAs wurden ausschließlich zum Kontakt- und Terminmanagement verwendet. Digitalkameras wurden zur Aufnahme von Bildern und zur Aufzeichnung von Videos benutzt. Weiterhin war die umgebende Funkinfrastruktur zur Lokalisierung einfach nicht flächendeckend verfügbar, während heute WLAN- und GSM-Technologie weit verbreitet und in fast jedem Haushalt vorzufinden ist. Inzwischen hat eine regelrechte Konvergenz der einzelnen Technologien hin zu neuartigen MEs, die richtige Universaltalente sind, stattgefunden. Diese MEs der neuen Generation werden als Smartphones oder auch Superphones bezeichnet[1] und können keiner der erwähnten Kategorien von MEs zugeordnet werden. So verfügen die MEs der neuesten Generation über eine Vielzahl an Sensoren, unter anderem WLAN, Bluetooth, GSM, IMUs, Kamera, Touchscreens, Lichtsensoren, Temperatursensoren und Näherungssensoren. All diese Sensoren sind heute in einem einzigen ME vereint und können zum Aufbau eines ILS eingesetzt werden. Diese Vielfalt an verfügbaren Sensoren und die Möglichkeiten, diese in den neuen MEs auf-

[1] Google Inc. hat bei der Vorstellung ihres neuesten Android basierten ME, dem Nexus One, zum ersten Mal den Begriff des Superphones verwendet.

grund der enormen Rechenleistung parallel nutzen zu können, ermöglicht es ein fein granulares Sensorabbild der Umwelt zu erschaffen. All diese Informationen tragen dazu bei, mehr über den Kontext des Nutzers zu erfahren und so Dienste, wie eine Lokalisierung in geschlossenen Räumen, zu ermöglichen.

Die neuen Möglichkeiten, welche uns die aktuellen MEs bieten, sind aber nicht genug, um eine Lokalisierung für den Nutzer zu ermöglichen beziehungsweise durchzuführen. Gerade für die Ermittlung der Position eines ME ist man stark auf die in der Infrastruktur befindlichen Funknetze angewiesen. Das Problem hierbei ist, dass der Nutzer eines solchen Systems, welches jeder installieren können sollte, im Normalfall keine neue Infrastruktur wie in Kapitel 4.8 vorgestellt installieren möchte. Deshalb musste im Rahmen der Konzeption des SSF versucht werden, mit bereits vorhandener oder leicht zu errichtender Infrastruktur auszukommen. Wesentlich sind die Kosten, die bei der Installation der Infrastruktur für den Nutzer entstehen. So ist es unwahrscheinlich, dass ein Nutzer mehrere WLAN-APs aufstellen wird, nur um Indoor-Lokalisierungsdienste einer mobilen Applikation nutzen zu können. Dagegen ist aber denkbar, dass ein Nutzer einen Marker in Form eines QR-Codes oder eine Karte seiner Umgebung auf dem ME erstellt und diese – im Falle der QR-Codes – ausdruckt und in seiner Umgebung verteilt. In den meisten Fällen muss aber zur Lokalisierung nicht einmal ein QR-Code ausgedruckt beziehungsweise eine Karte auf dem ME erstellt werden. Die Lokalisierung kann in den meisten Anwendungsfällen anhand gegenwärtiger Infrastrukturen, sei es in Form von vorhandenen Funknetzen (WLAN, GSM) oder visuellen Markern (QR-Codes) oder durch andere Objekte, welche im Raum vorhanden sind, durchgeführt werden. Diese bereits vorhandene Infrastruktur muss einfach nur dazu instrumentalisiert werden, den Nutzer zu lokalisieren. Wieviel Aufwand ein Nutzer aber letztendlich zur Installation seines eigenen ILS betreibt, kommt am Ende auf den Mehrwert an, den der Nutzer aus einem ILS ziehen kann. So wird ein Nutzer keine Kosten und Mühen scheuen, QR-Codes auszudrucken und diese in seinem Haus zu verteilen, wenn er über eine Applikation, die anhand der aktuellen Position die Lichtsteuerung im Haus übernimmt, viel Geld einsparen kann.

Das Ziel des SSF ist es nicht, den besten Anwendungsfall für eine Applikation zu entwerfen und abzubilden, sondern das Bereitstellen von Indoor-Positionsdaten für die Verwendung durch Applikationen der Android-Plattform. Die heute in den ME verbauten Sensoren bilden die Grundlage, um das SSF-ILS in der Praxis umzusetzen. Eine geeignete Fusion der Informationen, die von den verschiedenen Sensoren geliefert werden, soll dafür sorgen, dass das ILS so genau wie möglich arbeitet. Das SSF soll wie ein Katalysator für die Entwicklergemeinde wirken, welche mit innovativen neuen Konzepten und Ideen die Verwendung von MEs zur Indoor-Lokalisierung und Kontexterkennung vorantreiben. Das SSF soll somit den Grundstein zu neuartigen Applikationen legen, welche bisher nur außerhalb geschlossener Räume zur Verfügung standen.

Im Gegensatz zu dem im SSF verwendeten Konzept nutzen die meisten heute bekannten ILS spezialisierte Hardwarelösungen (siehe [PCB00], [WHFG92], [WJH97]). Außerdem versucht ein Großteil der Arbeiten genau eine Technologie zur Umsetzung eines ILS zu verwenden. Manche Arbeiten verwenden auch schon eine Kombination aus mehreren Sensoren, um eine Lokalisierung durchzuführen. So verwendet das von Zhou et al. [ZYN08] beschriebene ILS schon Funksignale aus WLAN und GSM Netzen zur genaueren Lokalisierung durch Fusionierung der Sensoren. Ebenfalls verwendet das von Bolliger et al. [BPCL09] vorgestellte ILS zusätzlich zur WLAN basierten Lokaliserung die IMU Sensoren zur Erkennung von Bewegung.

Dennoch gibt es keinen allgemeinen Ansatz mit dem Ziel, eine Indoor-Lokalisierung der breiten Masse an Nutzern zur Verfügung zu stellen. Das Konzept des SSF, ein ILS bereitzustellen, dass jeder Nutzer selber installieren und nutzen kann, unterscheidet sich deshalb deutlich von den meisten in Kapitel 3 vorgestellten ILS und wurde in dieser Form noch nicht implementiert.

5.2 Anforderungen

Ausgehend von der im letzten Abschnitt vorgestellten Konzeption werden nachfolgend die Anforderungen des SSF-ILS abgeleitet und festgelegt. Die Anforderungen werden im nächsten Abschnitt im Detail vorgestellt und diskutiert.

5.2.1 Verwendung gängiger mobile Endgeräte

Die zentrale Anforderung an ein ILS, welches nach Möglichkeit von jedem eingesetzt werden kann, ist die Verwendung gängiger mobiler Endgeräte, die über keinerlei spezialisierte Hard- beziehungsweise Software[2] verfügen müssen, um eine Lokalisierung durchzuführen. Die Anforderungen an die MEs sind also vorrangig allgemeiner Art und beziehen sich auf das Vorhandensein mehrerer Sensoren, die durch geeignete Kombination und Sensor-Fusion den Aufbau eines ILS ermöglichen. Hierbei ist es wichtig, dass die verschiedenen Sensoren je nach Zustand des Gesamtsystems verwendet werden und nach Möglichkeit immer genau der Sensor eingesetzt wird, der das beste Resultat bezüglich des aktuellen Kontextes liefert. So verfolgt SSF eine strategische Verwendung der Sensoren je nach aktuellem Zustand des Systems, in dem sich das ME befindet.

5.2.2 Vorrangige Nutzung bereits vorhandener Infrastruktur

Neben der Verwendung gängiger MEs müssen die Anforderungen an die umgebende Infrastruktur festgelegt werden. Dies ist von großer Bedeutung, da die MEs es zwar ermöglichen, mit Hilfe der Sensoren ein Sensor-Profil der Umgebung, in welcher sich das ME befindet, zu erstellen, hierzu aber immer externe Infrastruktur benötigt wird. Wichtig ist zu erkennen, dass der normale Nutzer des SSF in der Regel nur beschränkte oder gar keine technischen Kenntnisse aufweist. Vor diesem Hintergrund ist es nur schwer vorstellbar, dass ein Nutzer zum Beispiel dazu in der Lage sein wird, eine geeignete WLAN Infrastruktur aufzubauen. Selbst wenn ein Nutzer dazu in der Lage ist, möchte er sich im Normalfall nicht mit solchen Details auseinandersetzen, sondern mit relativ geringem Aufwand eine Lokalisierung in seinen Räumlichkeiten durchführen. Ganz abgesehen von den durch den Aufwand entstehenden Mehrkosten, die bei dem Aufstellen der aktiven Infrastruktur entstehen würden.

Bei der Umsetzung des SSF soll deshalb soviel vorhandene passive Infrastruktur benutzt werden wie möglich. Es sollen vorrangig die Technologien zu einer Lokalisierung verwendet

[2]Bei heutigen mobilen Betriebssystem Plattformen gibt es oft unnötige Einschränkungen bei dem Zugriff auf bestimmte Funktionen. Deshalb ist ein regelrechter Kampf um das sogenannte „Rooten“ von MEs zwischen Herstellern auf der einen und Entwicklern auf der anderen Seite ausgebrochen. Die Entwickler stellen den Nutzern Software-Hacks zur Verfügung die es ihnen erlauben, Funktionalität zu nutzen, die normalerweise nicht zugänglich ist. Von Rooten spricht man deshalb, weil in der Regel Exploits im Betriebssystem ausgenutzt werden, um Administrator (Root)-Rechte auf dem Gerät zu erhalten. Bei SSF wurde deshalb ausdrücklich darauf geachtet, nur Funktionalität zu verwenden, die keinerlei Root-Rechte benötigt.

werden, die bereits zur Verfügung stehen. Dies bedeutet, wenn möglich vorhandene Funkinfrastruktur zu benutzten und bei Verfügbarkeit auch mehrere Technologien zur Lokalisierung zu verwenden, um die Genauigkeit des Systems zu erhöhen. Bei den drahtlosen Technologien macht sich das SSF den Umstand zunutze, dass vor allem im städtischen Bereich eine ubiquitäre Verfügbarkeit von Funknetzen vorherrscht. Diese bereits vorhandene WLAN- und GSM-Infrastruktur kann sehr gut zur Lokalisierung verwendet werden und hat für das ILS keinerlei Auswirkung, da zu keiner Zeit eine aktive Verbindung bestehen muss.

Weiterhin kann die in den MEs integrierte Kamera dazu benutzt werden, optische Marker in der Infrastruktur zu erkennen. Optische Marker können zum Beispiel QR-Codes, Raumnummern oder auch Computerbezeichnungen sein, die an den jeweiligen Räumen beziehungsweise Geräten bereits angebracht sind. Die Erkennung der Marker erfolgt zum Beispiel über einen in dem ME integrierten Barcodescanner oder mittels OCR, welche anhand eines Algorithmus zur Bilderkennung versucht, Texte aus Bildern zu extrahieren. Wird den Markern dann jeweils eine feste Position im Raum zugewiesen, kann über eine optische Identifizierung dieser Marker eine Lokalisierung durchgeführt werden.

Passivität, Adaptivität und Erweiterbarkeit

Aufgrund der Tatsache, dass das SSF hauptsächlich von Endnutzern eingesetzt werden soll, ist es sinnvoll, dass das SSF einen starken passiven Charakter aufweist. Passivität hat in diesem Zusammenhang zweierlei Bedeutung. Passivität bedeutet zum einen, dass alle Daten über die verfügbaren Sensoren passiv gemessen werden. Das bedeutet beim Messen sowohl der WLAN als auch der GSM Feldstärken keine Verbindung zum jeweiligen AP beziehungsweise zur jeweiligen Zelle herzustellen, sondern jeweils alle empfangbaren Signale passiv auszuwerten (siehe hierzu auch Kapitel 4.8). Zum anderen bedeutet Passivität, dass keinerlei Infrastrukturkomponenten der Architektur, wie in Kapitel 4.7 beschrieben, verwendet werden. Die Berechnung der aktuellen Position soll ebenfalls ausschließlich auf dem ME durchgeführt werden.

Das SSF soll sich nach Möglichkeit der Umgebung, in der es eingesetzt wird, dynamisch anpassen und eine Art Adaptivität an geänderte Umstände bereitstellen. Dies bedeutet vor allem in Hinblick auf die Funktechnologien, dass das System ständig versucht, sich an die neuen Gegebenheiten der Raumcharakteristik anzupassen. Das Innenraum-Modell soll sich nach Möglichkeit autonom an die in den vorherigen Kapiteln vorgestellten Fluktuationen bezüglich der Signalausbreitung anpassen und lernen, ohne dass der Nutzer aktiv eingreifen muss.

Die Erweiterbarkeit des SSF ist ebenfalls von großer Bedeutung. Der technische Fortschritt der MEs schreitet mit hoher Geschwindigkeit voran. Die MEs verfügen über immer mehr Sensoren, die das Potential haben, zur Indoor-Lokalisierung eingesetzt zu werden. Gerade weil es noch keine allgemeingültige Lösung zur Indoor-Lokalisierung gibt, ist es wichtig, die Integration immer neuerer Sensoren, die entweder eine eigene unabhängige Lokalisierung bereitstellen beziehungsweise eine Ergänzung bereits vorhandener Daten durch Fusionierung ermöglichen, so einfach wie möglich zu gestalten. Dazu muss das SSF von Anfang an Schnittstellen anbieten, welche die Integration neuer Sensoren in das Framework erlauben.

5.2.3 Leicht benutzbar

Die Tatsache, dass SSF später als Grundlage für andere Applikationen dient, welche direkt vom Nutzer bedient werden, führt dazu, dass ein solches System so einfach wie möglich im Alltag zu verwenden sein sollte. Dies gilt sowohl für das Erstellen des Innenraum-Modells als auch bei der späteren Benutzung des System. Das Deployment durch den Nutzer muss sehr einfach und klar verständlich umgesetzt sein und darf keine komplizierten Aufgaben und Eingaben erfordern. So müssen einige wenige Konzepte ausreichen, um ein solches System einzusetzen zu können. Der Nutzer muss wissen, welche Schritte auszuführen sind, um ein Innenraum-Modell zu erstellen, aber zum Beispiel nicht genau wissen, wie der Mechanismus zur Adaption durch autonomes Lernen funktioniert oder wie die verwendete Bewegungserkennung in das System eingreift. Wichtig ist aber auch, dass das SSF durch den Entwickler der Applikation, aber auch durch den Nutzer konfigurierbar ist. SSF muss eine zentrale Stelle zur Konfiguration des Systems anbieten, so dass einzelne Sensoren oder berechnungsaufwändige Algorithmen einfach abgeschaltet werden können.

5.3 Theoretisches Modell

5.3.1 Verwendete Lokalisierungsverfahren

Das SSF definiert verschiedene unabhängige Lokalisierungsverfahren, die miteinander kombiniert werden können. Lokalisierungsverfahren sind in sich geschlossene Systeme, denen genau ein oder mehrere Sensoren zu Grunde liegen und die mit Hilfe dieser eine unabhängige Lokalisierung durchführen können. Unterstützt werden markerbasierte, displaybasierte und kontinuierliche Lokalisierungverfahren. Das markerbasierte Lokalisierungsverfahren verwendet im Raum verteilte Marker, wie zum Beispiel QR-Codes, um die Lokalisierung eines Nutzers durchzuführen. Die Nutzer können mit der im ME vorhandenen Kamera einen Marker scannen und sich so zu der vom Marker kodierten Position lokalisieren lassen. Das displaybasierte Lokalisierungsverfahren ermöglicht dem Nutzer, seine Position durch Eingabe seiner aktuellen Position über den Touchscreen dem System mitzuteilen. Die verwendeten kontinuierlichen Lokalisierungsverfahren nutzen dagegen drahtlose Technologien wie WLAN und GSM, um die aktuelle Position des Nutzers zu bestimmen.

Die höchste Genauigkeit bei der Lokalisierung bieten die in Kapitel 3.5.6.1 vorgestellten markerbasierten Ansätze. Bei einem markerbasierten Ansatz wird durch ein geeignetes Datenformat einem vorher festgelegten Punkt im Raum eine eindeutige ID zugewiesen, über die sowohl der Punkt und also auch die kodierte Position im Raum festgestellt werden kann. Alternativ können, wie beschrieben, auch schon in der Umgebung vorhandene Marker benutzt werden. Der Vorteil von markerbasierten Ansätzen besteht darin, dass sie die Position sehr genau bestimmen können. Der Nachteil ist, dass eine Lokalisierung nur dann durchgeführt werden kann, wenn der Nutzer sich zu den fixen Markern im Raum lokalisiert. Für die Verwendung des markerbasierten Ansatzes verwendet das SSF den im ME vorhandenen Kamerasensor. SSF verwendet zur markerbasierten Lokalisierung die in Kapitel 3.5.7 vorgestellten QR-Codes.

Die displaybasierte Lokalisierung wurde in Kapitel 3.5.7 eingeführt und verwendet das in allen MEs der Android Plattform integrierte Touchscreen-Display als Eingabegerät der aktuellen Position eines Nutzers. Dieses Verfahren bietet die Möglichkeit, auf einfache Art und Weise die aktuelle Position des Nutzers mit hoher Genauigkeit zu bestimmen und nutzt zudem eine

Eingabemethode, mit der die Benutzer eines ME vertraut sind. Dem Nutzer des SSF wird eine Karte der aktuellen Indoor-Umgebung angezeigt. Durch Berühren des Displays kann der Nutzer dem SSF seine aktuelle Position manuell mitteilen. Da ein Nutzer in der Regel ungefähr weiß, wo er sich gerade befindet, ist diese manuelle Eingabe der Position eine effiziente Art und Weise, die aktuelle Position eines Nutzers zu bestimmen. Um dem Nutzer die Benutzung zu erleichtern, werden zusätzlich die vorgestellten POI, Labeling- und Tagging-Verfahren benutzt. Trotzdem arbeitet dieses Verfahren in der Regel nicht so genau, wie ein markerbasiertes Verfahren; vor allem, wenn der Nutzer seine Position ohne die vom System bereitgestellten Hilfsmittel eingibt. Es liefert aber vor allem beim Einsatz von POI zur Lokalisierung gute Resultate. Das SSF unterstützt die Eingabe beliebiger Ortsdaten in das Framework, solange sich diese an das definierte Indoor-Geo-Format aus Kapitel 5.3.3 hält.

Die kontinuierliche Lokalisierung ist ein weiteres unabhängiges Lokalisierungsverfahren zur Bestimmung der aktuellen Position eines MEs. Die kontinuierliche Lokalisierung wird mittels der Funktechnologien WLAN und GSM umgesetzt. Zur Lokalisierung mittels drahtloser Technologien findet im SSF das LFPT Verfahren Anwendung, welches in Kapitel 4 ausführlich beschrieben wurde. In einem ersten Schritt wird bei diesem Verfahren ein Modell zur Signalausbreitung anhand von gemessenen Feldstärken erstellt. In einer TRP werden an vordefinierten TPs in den Räumlichkeiten der Installation Messungen bezüglich der an diesem Punkt empfangbaren Feldstärken durchgeführt und in Form eines LFPTs in einer Datenbank, der RM, abgespeichert. In der RTP werden zur Lokalisierung erneut die Feldstärken an der aktuellen Position des ME aufgenommen und mit allen in der Datenbank abgelegten LFPTs verglichen. Dieses Vorgehen ermöglicht es, nach initialer Aufzeichnung der RM kontinuierlich die Position des MEs zu bestimmen.

SSF implementiert momentan zwei Module, die eine kontinuierliche Lokalisierung bereitstellen: ein WLAN-LFPT Modul und ein GSM-LFPT Modul. Beide Module können unabhängig voneinander oder in Kombination eingesetzt werden. Zusätzlich verwendet das SSF ein sogenanntes Motion-Detector Modul, welches die IMUs zur Bewegungserkennung nutzt und über eine Schnittstelle bereitstellt, was es den WLAN-LFPT und GSM-LFPT Modulen erlaubt, bei der Aufnahme der LFPTs durch Bewegung, Orientierung und die Lage eines Nutzers für eine Steigerung der Genauigkeit der kontinuierlichen Lokalisierung in der TRP und RTP zu sorgen.

5.3.2 Fixe Position und Bewegungserkennung

SSF unterscheidet als Eingabeparameter zwischen fixer Position und Bewegung. Fixe Positionen sind physikalische Referenzpositionen innerhalb des ILS, die mit hoher Genauigkeit vorliegen. Bei diesen handelt es sich um Positionen, die entweder durch die QR-Code, Display oder Dialog-Eingabemethode des SSF gemacht wurden. Je nachdem, welche Module zur Lokalisierung verwendet werden, dient die Eingabe einer fixen Position auch als Eingabeparameter des Zustandsautomaten (FSM) des SSF, welcher anhand der Eingabeparameter der fixen Position und der Bewegung über den aktuellen Zustand des SSF entscheidet. Liegt am Eingang der Zustandsmaschine eine fixe Position an, dann beginnt das SSF, wenn keine Bewegung des MEs vorliegt, kontinuierlich LFPTs aufzunehmen. Neben dem Eingabeparameter der fixen Position gibt es noch einen zweiten Eingabeparameter für die FSM, die Bewegung des ME. Je nach anliegenden Eingabeparameteren trifft die FSM dann eine Entscheidung, wann LFPTs oder RFPTs aufgenommen werden sollen und führt die Zustandswechsel transparent für den Nutzer des Systems durch. So werden unter anderem während der RTP nicht nur RTFPT von

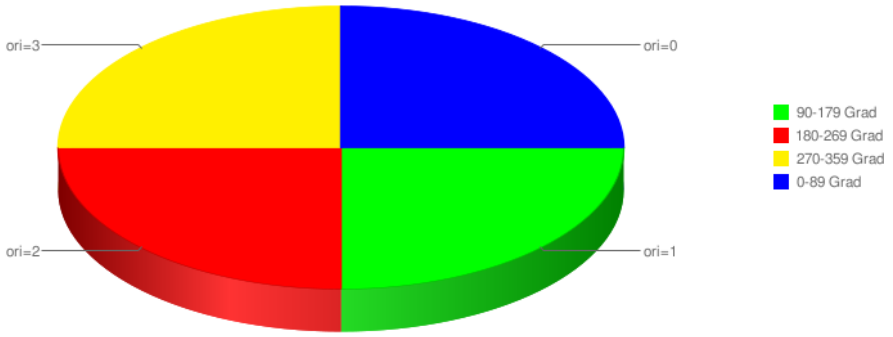


Abbildung 5.1: Quantisierung der vom elektrischen Kompass gelieferten Orientierung in der RTP in 4 Intervalle

WLAN und GSM erfasst und mit den LFPT in der RM verglichen, sondern diese darüber hinaus noch zur Autoadaption der RM verwendet. Liegt Bewegung als Eingabeparameter an, so wird die Aufnahme gestoppt. Die Schwellenwerte für die Bewegungserkennung in der TRP und RTP müssen allerdings vorab konfiguriert werden. Die FSM wird in Kapitel 6.2.2.5 genauer beschrieben.

Bewegungserkennung

Zur Bewegungserkennung eines ME werden mittlerweile die in allen Android basierten ME verbauten Micro-Electro-Mechanical Systems (MEMS) verwendet. Durch die Verwendung dieser IMU Sensoren wie elektrischer Kompass, Beschleunigungsmesser und Lagesensor kann auf die Bewegung eines ME geschlossen werden. Die Bewegungserkennung ist eine Entweder-oder-Entscheidung: entweder bewegt sich das ME im Raum oder es befindet sich gerade in Ruhelage. Im weiteren Verlauf der Arbeit werden hierfür die von Bollinger et al. [BPCL09] eingeführten Begriffe „dynamischer Zustand“ und „statischer Zustand“ verwendet. Die Bewegungserkennung ist wichtig, da es wie in Kapitel 4 beschrieben leicht zu Interferenzen der Signalcharakteristik kommen kann. Dies würde zu einer Verfälschung der Aufnahme der LFPTs in der TRP führen. Dies wiederum führt zu einer Verfälschung der in der RM abgelegten LFPT und beeinträchtigt die Lokalisierung in der RTP signifikant. Aus diesem Grund wurde im SSF ein Modul implementiert, welches Bewegung erkennen und die Aufnahme der LFPTs entsprechend stoppen kann, bis sich das Gerät wieder in Ruhelage befindet oder die Beschleunigung so gering ist, dass die zu erwartenden Interferenzen gering sind. So wird gewährleistet, dass Interferenzen durch Bewegungen keinen nachhaltigen Einfluss auf die Qualität der RM haben. Die Bewegungserkennung wird aber nicht nur in der TRP, sondern auch in der RTP angewendet. So wird eine Lokalisierung in der RTP nur dann ausgeführt, wenn sich die absolute Beschleunigung des ME unter einem vom Nutzer einstellbaren Schwellenwert befindet und anhand dieser Werte eine Entscheidung bezüglich eines dynamischen und statischen Zustands getroffen werden kann. Die Zustandsmaschine definiert darum auch einen eigenen Zustand für Bewegung.

5.3.3 Datenformat

Als Datenformat zur Beschreibung von Indoor-Geo-Positionen verwendet das SSF sowohl SIDs als auch ein 4-Tupel Format zur Kodierung der Koordinaten. SIDs werden durch einen URI-String der Form: `name:<SID>` kodiert. Zur Beschreibung der Positionen in Koordinaten verwendet SSF einen Quadrupel $L = \{x, y, floor, ori\} \mid x, y, floor, ori \in \mathbb{R}^4$. x und y kodieren beliebige Punkte in der x-y-Ebene, $floor$ kodiert das Stockwerk und ori ist eine quantisierte Orientierung, welche die Werte 0 ($0-89°$), 1 ($90-179°$), 2 ($180-269°$), ($270-359°$) annehmen kann. Die Orientierung muss nicht zwingend angegeben werden und ist standardmäßig mit dem Wert 0 initialisiert. Wird die Orientierung verwendet, wird die Quantisierung in der RTP anhand der mit dem elektrischen Kompass gemessenen Orientierung erfasst und quantisiert. Die Quantisierung ist in Abbildung 5.1 dargestellt und kann die vier beschriebenen Wert 0, 1, 2, 3 annehmen.[3] Die Darstellung des Quadrupels innerhalb des verwendeten Positions-Strings orientiert sich an der von der IETF vorgeschlagenen Erweiterung des URI-Standards für geografische Informationen [IETa]. Zusätzlich wird die vorgeschlagene Erweiterung um die Orientierung ergänzt. Eine physikalische Indoor-Geo-Position wird dann letztendlich anhand seiner Koordinaten als URI-String der Form `igeo:<x>,<y>,<floor>,<ori>` kodiert. Das endgültig von SSF verwendete Format zur Beschreibung einer Indoor-Geo-Position setzt sich dann aus einer Konkatenation der SID-URI, des Trennzeichens — und der Geo-URI zusammen. Das verwendete Format sieht folgendermaßen aus:

```
name:<SID>|igeo:<x>,<y>,<floor>,<ori>
```

Genauere Details und einen Überblick über Formate, die sich zur Repräsentation von Indoor-Geo-Positionen in ILSs eignen und eingesetzt werden, finden sich im Design Kapitel unter Kapitel 4.9.5.

5.3.4 Eingabe-Methoden

SSF unterstützt drei verschiedene Eingabemethoden zur Bekanntmachung einer physikalischen Position an das System: QR-Codes, dialog- und displaybasierte Eingabe. Bis auf die dialogbasierte Eingabe sind die Eingabemethoden nicht Kernbestandteil des SSF, sondern separat implementierte Module, die mit dem Framework ausgeliefert werden oder zusätzlich über den Android App Market installiert werden können. Im System vorhandene Eingabemethoden werden vom SSF automatisch erkannt und können dann als Eingabemethode für physikalische Positionen benutzt werden.

Die Eingabemethode der QR-Codes werden der vorgestellten markerbasierten Lokalisierung zugeordnet und ermöglichen eine automatisierte, robuste und performante Möglichkeit zur Lokalisierung eines ME. QR-Codes werden im SSF nicht nur dazu verwendet, eine Lokalisierung durchzuführen, sondern dienen aufgrund ihrer hohen Genauigkeit auch als Eingabeparameter einer fixen Position an die FSM, die für die Steuerung der kontinuierlichen Lokalisierung verantwortlich ist. Da die vorgestellte TRP die Grundlage zur kontinuierlichen Lokalisierung anhand der in SSF verwendeten LFPT Technik bildet, ist es zur Aufnahme von LFPTs an vorab definierten TPs und deren Verwendung zur Erstellung der initialen RM nötig, dass das SSF

[3] Die vier Werte für die Quantisierung können auch individuell angepasst werden, in dem das Konfigurationsobjekt des SSF mit einem entsprechenden Wert parametrisiert wird. Die abgedeckten Winkel ändern sich dann dementsprechend.

name:TP01|igeo:1,0,1,1

Abbildung 5.2: QR-Code zur Bekanntmachung einer fixen Position an das SSF. Der QR-Code kodiert hierbei das vom SSF verwendete Format zur Beschreibung von Positionen anhand eines SIDs (name) und einer physikalischen Position (igeo), bestehend aus X- und Y-Position, Stockwerk und der Orientierung.

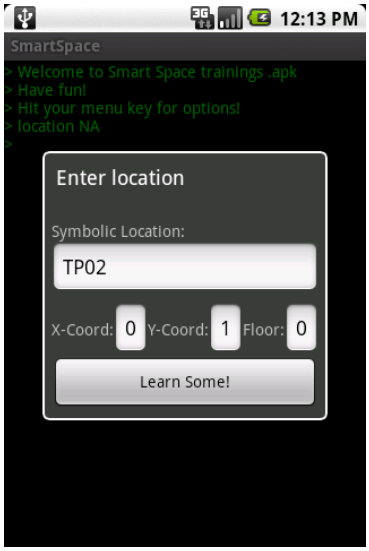


Abbildung 5.3: Screenshot des SSF auf einem HTC G1. Angezeigt wird ein Dialog zur Eingabe einer fixen Position. Die Eingabe setzt sich aus einem SID und einer physikalischen Position, bestehend aus X- und Y-Koordinate, und einer Angabe für das Stockwerk zusammen.

Kenntnis über den aktuellen Aufenthaltsort des ME hat. Wesentlich ist hier die Tatsache, dass die aktuelle physikalische Position innerhalb des Referenzsystems mit hoher Genauigkeit bestimmt werden kann, da es ansonsten zu einer Verfälschung der in der TRP aufgenommenen LFPT in der RM kommen würde. Dies würde sowohl die Genauigkeit wie auch die Zuverlässigkeit des Systems signifikant beeinflussen. Die Eingabe von Positionen mit Hilfe von QR-Codes wird durch den Nutzer vorgenommen. Um dem SSF eine Position bekannt zu machen, kann der Nutzer durch einfaches Abscannen eines QR-Codes dem SSF den aktuellen Standort mitteilen.[4] Ein typischer QR-Code, wie er zur Bekanntmachung einer fixen Position in das SSF verwendet wird, ist in Abbildung 5.2 dargestellt. Der dargestellte QR-Code kodiert eine Position anhand des vom SSF definierten Formats zur Repräsentation von Indoor-Geo-Positionen, welches unterhalb des QR-Codes nochmals als String dargestellt ist. QR-Codes sind eine sehr effiziente Weise zur Generierung von Markern zur Eingabe von Positionen in das SSF.

QR-Codes können schnell und ohne größeren Aufwand automatisiert erstellt werden.[5] Der Nutzen bei der Verwendung von QR-Codes liegt aber nicht nur in der automatisierten Erstellung der Marker. Ebenfalls wird auch die Zeit bei der Aufnahme der RM durch die Verwendung von QR-Codes, vor allem in großen Installation mit mehr als 10 TPs signifikant verkürzt. In kleineren Installation, die keine markerbasierte, sondern eine rein kontinuierliche Lokalisierung anwenden, lohnt sich der Aufwand zur Erstellung der QR-Codes in der Regel nicht. Wird jedoch sowohl markerbasierte als auch kontinuierliche Lokalisierung verwendet, ergibt sich ein zusätzlicher Nutzen aus der Verwendung von QR-Codes. Wie beschrieben liegt durch das Abscannen des QR-Codes auch eine fixe Position an der FSM an. Solange der Nutzer sich nicht bewegt, kann diese Information dazu benutzt werden, ein automatisiertes Training, eine Autoadaption der RM vorzunehmen.

Die zweite vom SSF unterstützte Eingabemethode ist die der Dialogeingabe. Bei dieser Methode wird dem Nutzer ein Dialog zur Eingabe der physikalischen Position angezeigt. Die Eingabe erfolgt manuell durch den Nutzer des SSF. Abbildung 5.3 zeigt einen Screenshot des verwendeten Testgerätes, welches einen Dialog zur Eingabe einer Position präsentiert. Der Nutzer kann einen symbolischen Identifier für die Position, eine Lage in der x-y-Ebene und ein Stockwerk angeben. Hat der Nutzer die Daten eingegeben, muss er den eingegebenen Punkt bestätigen. Die Eingabe des Nutzers wird nach Bestätigung der Positionseingabe vom SSF intern in das SSF interne Datenformat konvertiert und kann sofort verwendet werden. Die Eingabemethode über einen Dialog in der Benutzerschnittstelle ist von Nutzer einfach zu bedienen und hat im Vergleich mit der Eingabe Methode durch QR-Codes den Vorteil, dass keinerlei Vorarbeit in Form von Erstellung mehrerer QR-Codes geleistet werden muss. Sie eignet sich sehr gut in kleineren Installation mit wenigen TPs zur Erstellung des Innenraum-Modells zur kontinuierlichen Lokalisierung. Einen Nachteil bietet diese Eingabemethode bei der Verwendung einer markerbasierten Lokalisierung: Die Eingabe der aktuellen Position ist wesentlich zeitintensiver als das Scannen eines QR-Codes und macht nur Sinn, wenn eine Lokalisierung in einer Umgebung durchgeführt werden soll, in der keinerlei umgebende Funkinfrastruktur vorhanden ist.

[4] SSF verwendet hierzu den populären und frei verfügbaren Barcodescanner ZXING [ZXIa] für die Android Plattform. Dieser kann einfach über das Android-Intent-Subsystem mitbenutzt werden, ohne eine eigene Lösung zu implementieren.

[5] Im Laufe der Arbeit wurde neben der eigentlichen Android Software noch eine Google Chrome Extension zur Generierung von QR-Codes implementiert. Die Extension nutzt die Google Charttools API [Gooa] und generiert eine konfigurierbare Anzahl an QR-Codes im Indoor-Geo-Format des SSF. Die Extension liegt ebenfalls dem Quellcode dieser Arbeit bei.

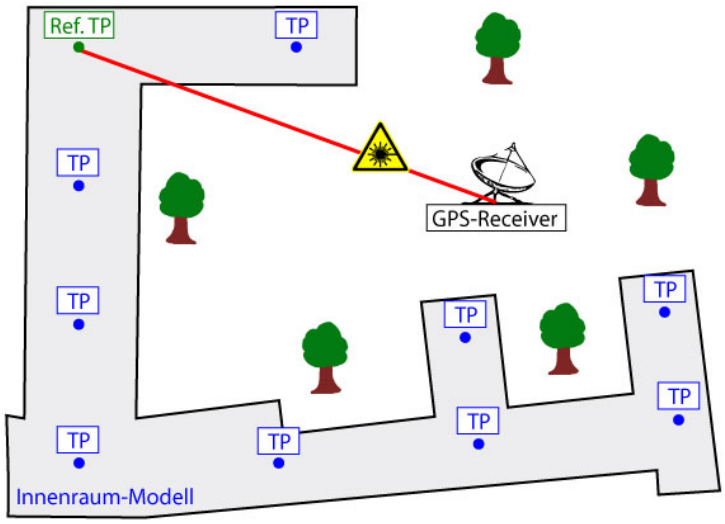


Abbildung 5.4: Geo-Mapping Verfahren im SSF. Zeigt die Position eines ME relativ zu der oberen linken Ecke des Innenraum-Modells. Hierzu wird die linke obere Ecke des Innenraum-Modells mit Hilfe eines GPS-Receivers vermessen.

Die letzte vom SSF unterstütze Eingabemethode von physikalischen Positionen ist die displaybasierte Eingabemethode, bei der die Bekanntmachung der aktuellen Position an das SSF durch das Erstellen eines POI erfolgt. Bevor diese Eingabemethode verwendet werden kann, müssen die Nutzer aktive Vorarbeit leisten. In einem ersten Schritt muss eine Karte in Form eines .png oder .jpg Bildes auf das ME kopiert werden und anschließend vom Nutzer mit eigenen POI versehen werden. Die POI müssen vom Nutzer selbst durch Eingabe über den Touchscreen des ME erstellt werden. Es kann sich hierbei um beliebige Positionen handeln, zu denen später eine Lokalisierung durchgeführt werden soll. Zum Beispiel kann ein Nutzer der ein SSF-ILS in seinen Räumlichkeiten installieren möchte einen Grundriss seines Hauses oder seiner Wohnung erstellen, auf das ME kopieren und dann Räume oder Orte in seinen Räumlichkeiten als POI, wie Bad, Küche, Wohnzimmer, Arbeitszimmer, Kinderzimmer angeben. Der Nutzen dieser Eingabemethode ist, dass sie durch die visuelle Darstellung der Indoor-Umgebung für den Nutzer sehr intuitiv und einfach bedienbar ist. Sind die Positionen in Form von POI einmal angelegt, können diese zum einen zu einer rein displaybasierten Lokalisierung genutzt werden, zum anderen als TPs in der TRP als Eingabeparameter einer fixen Position für die FSM zum Erstellen eines Innenraum-Modells verwendet werden. Ein Nachteil bei der Verwendung dieser Eingabemethode ist jedoch die aufwendige Erstellung einer Karte der Räumlichkeiten der Installation und die manuelle Eingabe der Position. Durch die relativ geringe Auflösung der Displays von MEs und einer gewissen Ungenauigkeit bei der Eingabe von Positionen über den Touchscreen, kann es zu einer gewissen Ungenauigkeit bei der Eingabe der physikalischen Positionen kommen, welche im Normalfall aber keine Auswirkungen auf die Performance des Systems haben.

5.3.5 Geo-Mapping

Mit Geo-Mapping wird eine im SSF verwendete Technik bezeichnet, die es ermöglicht, das relative Koordinatensystem eines SSF Innenraum-Modells[6] auf das wohl bekannte WGS 84 Koordinatenreferenzsystem zu übertragen. Hintergrund ist die Darstellung eines geeigneten Innenraum-Modells unter der Verwendung von Industriestandards wie GPX oder KML. Um ein Geo-Mapping durchführen zu können, müssen alle in Smartspace verwendeten physikalischen Positionen relativ zu einer Referenzposition innerhalb des Innenraum-Modells angeben werden. Abbildung 5.4 zeigt, wie die Position eines ME relativ zu der oberen linken Ecke des Innenraum-Modells angegeben wird. Hierzu wird die linke obere Ecke des Innenraum-Modells mit Hilfe eines GPS-Receivers vermessen. Die Messung sollte eine hohe Genauigkeit aufweisen und deshalb an mehreren Tagen durchgeführt und gegebenenfalls ein Mittelwert gebildet werden. Die Verlässlichkeit dieser Messung ist wichtig, weil alle Positionen des Innenraum-Modells sich auf die vermessene Referenzposition beziehen und sich der Fehler auf jede Position innerhalb des SSF Referenzsystems auswirken würde. Diese relative Angabe einer physikalischen Position kann dazu benutzt werden, die Positionen des Innenraum-Modells in Geo-Koordinaten nach WGS 84 zu übersetzen und es so ermöglichen, diese in einem Geo-Daten Format wie zum Beispiel KML zu exportieren und in einer beliebigen Applikation oder zur Darstellung in der Benutzerschnittstelle, zum Beispiel innerhalb von Google Mobile Maps, zu verwenden. Bei der Verwendung von KML und GPX gibt es durch die Möglichkeit, einen Namen mit einem Geo-Objekt zu verbinden, zusätzlich noch die Möglichkeit, die symbolische Position mit aufzunehmen, was gerade bei der Darstellung in Google Earth oder Google Maps sehr hilfreich sein kann.

Geo-Mapping ist deshalb von großer Bedeutung, weil es erlaubt, Daten eines Innenraum-Modells in ein standardisiertes Format zu transformieren und exportieren zu können. Somit kann das Innenraum-Modell auch außerhalb der ME erstellt und verwendet werden. Ein weiterer wichtiger Aspekt ist, dass es durch die Übersetzung der physikalischen Position in Koordinaten des WGS 84 Referenzsystems möglich wird, eine saumenlose Integration von Outdoor Lokalisierungssystemen, wie zum Beispiel GPS, zu realisieren. Dies wiederum ermöglicht die Implementierung von Diensten und Applikationen, welche nicht mehr zwischen einer Indoor- und einer Outdoor-Umgebung unterscheiden müssen. Zusätzlich können die Daten durch Verwendung eines standardisierten Formats wie GPX oder KML auch in anderen Applikation, die nicht auf ein ME beschränkt sind, benutzt werden.

Die Probleme bei der Verwendung von Geo-Mapping bestehen in der Vermessung des Referenzpunkts in absoluten Koordinaten nach WGS 84. Der gegebenenfalls bei der Vermessung des Referenzpunkts entstandene Fehler wird bei der Transformation der Koordinaten für jede Position innerhalb des Innenraum-Modells übernommen. Somit ist die gesamte Genauigkeit des ILS in Bezug auf die Indoor-Umgebung, zum Beispiel bei der Darstellung auf der Google Mobile Map, abhängig von der Genauigkeit des vermessenen Referenzpunkts.

[6] SSF verwendet ein ein eigenes Format, welches in Kapitel 5.3.3 beschrieben wird.

6 Smartspace Framework Implementierung

Dieses Kapitel stellt die Implementierung des SSF anhand des beschriebenen theoretischen Modells vor. Im Anschluss wird ein detaillierter Überblick über die wichtigsten Design-, Architektur- und Implementierungskonzepte gegeben. Zusätzlich ist dieser Teil für Entwickler gedacht, welche die interne Arbeitsweise des SSF verstehen und erweitern möchten. Bei der Beschreibung der Implementierung werden an vielen Stellen objektorientierte Softwareentwurfsmuster zur Erklärung der Konzepte verwendet. Diese werden aber zu einem besseren Verständnis des Textes nicht im Detail erklärt. Im Anhang B findet sich deshalb eine Übersicht und Erklärung der verwendeten Entwurfsmuster. Im Anhang C wird außerdem eine kurze Einführung in die grundlegenden Konzepte der zur Umsetzung des SSFs ausgewählten Google Android Plattform gegeben.

6.1 Design

Beim Design des SSF wurde stark darauf geachtet, die im Theorieteil erarbeiteten Erkenntnisse bezüglich der verwendeten Technologien und Verfahren bei der Implementierung der Software so gut wie möglich abzubilden. Das Framework soll eine angebrachte Abstraktion der Problemdomäne modellieren. Deshalb würde ein rein sensorbasierter Ansatz die Möglichkeiten der Entwickler beschränken, da die Art der Anwendungen, die sie so in der Lage sind zu entwickeln, begrenzt wären. Das Framework muss deshalb vor allem eine gute Abstraktion bieten und darf nicht von verfügbaren Sensoren ausgehen, sondern muss darauf basieren, Lokalisierungsinformationen unabhängig von den darunterliegenden Sensoren zu liefern. Schichtenmodelle oder Komponentenmodelle mit guten Abstraktionen und einer sauberen API sind der einzige Weg, dieses Ziel zu erreichen. Sie führen zu einem wiederverwendbaren und erweiterbaren Framework. Das Design wurde vor allem vor diesem Hintergrund entworfen, mit besonderem Augenmerk auf die nichtfunktionellen Eigenschaften der Softwareentwicklung wie Erweiterbarkeit, Robustheit, Skalierbarkeit, Wartbarkeit und Wiederverwendbarkeit. Vor allem die Erweiterbarkeit der Software stand im Vordergrund. So musste von Anfang an ein Design entworfen werden, welches eine Abstraktion der Anforderungen darstellt und nur den Rahmen sowie die Kommunikation zwischen den konkreten Komponenten wie Sensoren, Filter zur Datenverarbeitung, Positionierungsalgorithmen und Persistenz in Form von klar definierten Schnittstellen vorgibt.

Ein weiterer wichtiger Punkt beim Entwurf des Designs war das Leistungsverhalten des ILS. Um ein gutes Leistungsverhalten zu gewährleisten, baut das SSF auf einer Multithreading-Architektur auf. Das heißt, jeder im System verwendete Sensor und alle an der Verarbeitung und Lokalisierung beteiligten zentralen Komponenten werden in einem eigenen Thread ausgeführt. Zudem werden ankommende Daten jeweils nicht direkt an die übergeordnete Komponente ausgeliefert, sondern in die Warteschlange der übergeordneten Komponente geschrieben.

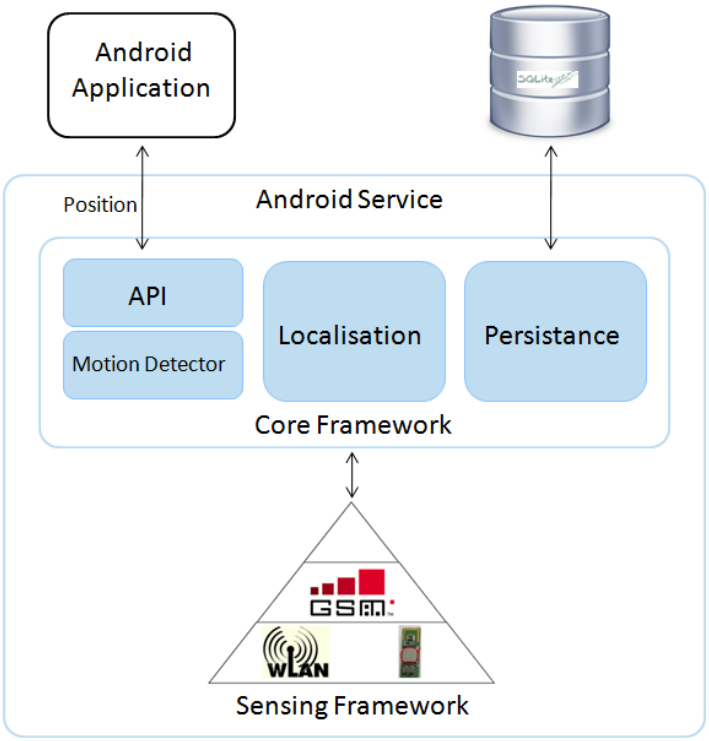


Abbildung 6.1: Architektur des SSF, bestehend aus den zwei Subsystemen Sensing- und Core-Framework. Zusätzlich dargestellt sind die Komponenten innerhalb der beiden Subsysteme und die zu externen Applikationen bereitgestellte API.

Diese entnimmt die Daten aus der Warteschlange und führt die weitere Prozessierung durch. Wichtig für das Leistungsverhalten ist hierbei, dass die Komponenten-Threads blockieren, solange sich keine Daten in der Warteschlange befinden und somit keine Rechenzeit in Anspruch nehmen. Weil dieses Prinzip den aus Unix Betriebssystemen bekannten „Pipes“ sehr ähnlich ist, wird dieses auch gemeinhin als „Pipes-and-Filters„ Entwurfsmuster bezeichnet. Ein Beispiel für dieses Verhalten sind zum Beispiel die „Sensing-Threads“, welche ihre Sensordaten in die Warteschlange des Reactors schreiben. Der Reactor läuft ebenfalls in einem eigenen Thread, entnimmt die Sensordaten und arbeitet diese sequentiell ab. Anschließend leitet er die Daten an einen passenden Event-Handler weiter, welcher die Daten in die Warteschlange des Sensor-Synchronizer einstellt. Dieser läuft ebenfalls in einem eigenen Thread und nimmt unter Zuhilfenahme eines Threadpools eine effiziente und parallele Weiterverarbeitung der Sensordaten vor. Das Ergebnis wird als LFPT in der RM persistiert.

6.2 Architektur und Implementierung

Dieser Abschnitt beschreibt die Umsetzung vom theoretischen Modell hin zu einer konkreten Implementierung. Es soll vor allem dazu dienen, einerseits dem Leser einen Überblick über den Aufbau des SSF zu vermitteln und andererseits dem Entwickler, der das SSF in einer seiner Applikationen einsetzen möchte, eine Dokumentation und ein Nachschlagewerk bereitzustellen.

Der grundlegende Aufbau des SSF ist durch zwei Subsysteme gegeben, welche in mehreren Komponenten organisiert sind. Die Einteilung und die Aufgaben der einzelnen Komponenten

orientieren sich grob an der Arbeit von Hightower et al. [HBB02] „The Location Stack: A Layered Model for Location in Ubiquitous Computing“. Die Autoren führten in ihrer Arbeit eine Untersuchung mehrerer ILS Implementierungen durch, um daraus Prinzipien für ein Software-Engineering-Modell abzuleiten. Softwareentwickler sollen so mit einem geeigneten Standardmodell und dem dazugehörigen Vokabular zu der Implementierung eines ILS ausgestattet werden. Das Ziel ist es vor allem, Entwicklern dabei zu helfen, Implementierungskonzepte besser diskutieren, sich mit anderen Entwickler besser verständigen und leichter auf dieser Arbeit aufbauen zu können. Das vorgestellte Modell verwendet ein ähnlich dem für Rechnernetze bekannten Open System Interconnect (OSI) Modell, also eine Aufteilung des Software-Stacks in fünf verschiedene Schichten. Es definiert weiterhin eine geeignete Terminologie zur Benennung der einzelnen Schichten. Im Folgenden wird eine kurze Auflistung und Beschreibung der fünf Schichten des Modells präsentiert:

Sensor-Schicht Beinhaltet die eigentliche Sensor-Hardware und die dazugehörigen Treiber und exportiert die von den Sensoren gelieferten Rohdaten an die $n+1$ Measurement-Schicht in einer Vielzahl von Formaten.

Measurement-Schicht Enthält Algorithmen zum Transkribieren der rohen Sensordaten und exportiert einen Datenstrom aus zum Beispiel Abstands-, Winkel- und Näherungsmessungen, geschätzten Positionen oder nichtgeometrische Messungen, wie gemessene Feldstärken.

Fusions-Schicht Definiert eine allgemeine Methode, um einen kontinuierlichen Strom von Messungen mit Zeitstempeln zu versehen und die Berechnung der aktuellen Position und Orientierungen des Objektes durchzuführen. Diese Schicht ist auch für die Fusion mehrerer Messungen aus unterschiedlichen Sensoren zuständig, um eine Verringerung der Unzuverlässigkeit beziehungsweise der Ungenauigkeit zu erreichen. Falls erforderlich, übernimmt die Fusion-Schicht auch die Vergabe von eindeutigen Kennungen. Anschließend wird an die $n+1$-Schicht, entweder eine Schnittstelle zur Abfrage oder ein Event, welcher die Daten kapselt, exportiert. Die Schnittstelle oder das Event müssen dabei mindestens den aktuellen Standort und die aktuelle Genauigkeit/Zuverlässigkeit der einzelnen Objekte beinhalten. Je nach verwendeter Technologie können der Arrangement-Schicht noch zusätzliche Informationen zum Zustand des mobilen Objektes wie Geschwindigkeit, Beschleunigung, ID und Name des Objekts bereitgestellt werden.

Arrangement-Schicht Stellt eine Engine bereit, welche für die Bestimmung von Beziehungen unter den lokalisierten Objekten wie zum Beispiel Nähe, Containment oder auch geometrische Formationen zuständig ist. Die Arrangement-Schicht exportiert wiederum eine Schnittstelle oder einen Event, welcher Informationen über Beziehungen zwischen zwei oder mehreren Objekten enthält.

Contextual-Fusion Stellt eine Fusion der dem System vorliegenden Standortdaten und deren Beziehungen mit anderen Kontextinformationen bereit. Erweiterte Kontextinformationen können sich je nach System unterscheiden und können persönliche Daten wie Kalender, E-Mail, Kontakte, To-Do-Listen etc. sein. Exportiert wird von dieser Schicht eine Schnittstelle, die direkt von den Anwendungen abgefragt werden kann, um es dem System zu ermöglichen, interessante Situationen zu erkennen, um angemessen und wenn möglich automatisiert auf das Ereignis zu reagieren.

Ausgehend von dem beschriebenen Modell wurde die Architektur des SSF-ILS entworfen. Es besteht aus zwei Subsystemen, dem Sensing- und dem Core-Framework. Das Sensing-Framework beinhaltet Komponenten zur Koordination der Sensoren und zur Aufzeichnung und Verarbeitung der Sensordaten. Das Core-Framework enthält die Komponenten zur Positionsbestimmung, Persistenz, Bewegungserkennung und stellt zusätzlich eine Schnittstelle zur Verwendung des SSF in Applikationen bereit. Die Koordination und Steuerung der Subsysteme wird von einem Zustandsautomaten, der Finite State Machine (FSM), übernommen, welche eine Zustandssteuerung der Subsysteme und der einzelnen Komponenten anhand von Umgebungsparametern[1] vornimmt.

Der grundlegende Aufbau der Architektur wird in Abbildung 6.1 dargestellt. Sie zeigt die beiden Subsysteme Sensing-Framework und Core-Framework. Zusätzlich dargestellt sind die Komponenten innerhalb der Subsysteme. Die Kommunikation zwischen den Subsystem wird über Schnittstellen entkoppelt, welche sich das Strategy-Entwurfsmuster zu Nutze machen, um je nach Zustand des Systems ein anderes Verhalten durch eine andere Strategie zu realisieren. Die beiden Subsysteme und ihre einzelnen Komponenten werden im Anschluss im Detail beschrieben und die wichtigsten Stellen in Form von Code-Beispielen verdeutlicht. Zusätzlich werden Angaben gemacht, in welchem Paket die angegebenen Klassen zu finden sind. Allgemein gilt aber, dass alle Klassen, die in irgendeiner Weise Klassen des Android-Application-Frameworks (AAF) nutzen oder Referenzen auf Objekte des AAFs halten, sich im Paket `de.ambisense.smartspace.android` befinden. Der Hintergrund ist die Portierbarkeit des SSF auf andere Java-Plattformen. Auf diesen muss dann lediglich die konkrete Implementierung der Klassen im `de.ambisense.smartspace.android` Paket ausgetauscht werden, um das Framework benutzen zu können.

6.2.1 Sensing-Framework (SeF)

Das Sensing-Framework (SeF) ist für die Kapselung und Koordination der verfügbaren Sensoren, der Daten-Prozessierung und der Fusion der von verschiedenen Sensoren gelieferten Daten zuständig. Es setzt sich aus den nachfolgenden Komponenten zusammen:

- Sensing,
- Synchronisation,
- Data-Processing,
- Sensor-Fusion.

Abbildung 6.2 gibt einen Überblick über die einzelnen SeF-Komponenten und die Einordnung des SeF in der Gesamtimplementierung des SSFs. Hierbei wird dargestellt, dass die einzelnen Komponenten eine Kondensierung der Sensordaten von Rohdaten hin zu aufbereiteten Datenstrukturen, die zur Lokaliserung verwendet werden können, durchführt. Weiterhin dargestellt ist die Ansteuerung des Sensing-Frameworks durch die FSM des Core-Frameworks, welche die Steuerung des Sensing-Frameworks anhand des aktuellen Systemzustands durchführt. Zusätzlich sind noch die vom SSF unterstützen Eingabemethoden zur Eingabe fixer Positionen

[1] Umgebungsparameter können die nicht beeinflussbaren Eingabeparameter fixe Position oder Bewegung sein. Andere Parameter können über das Konfigurationsobjekt gesetzte Systemparameter sein.

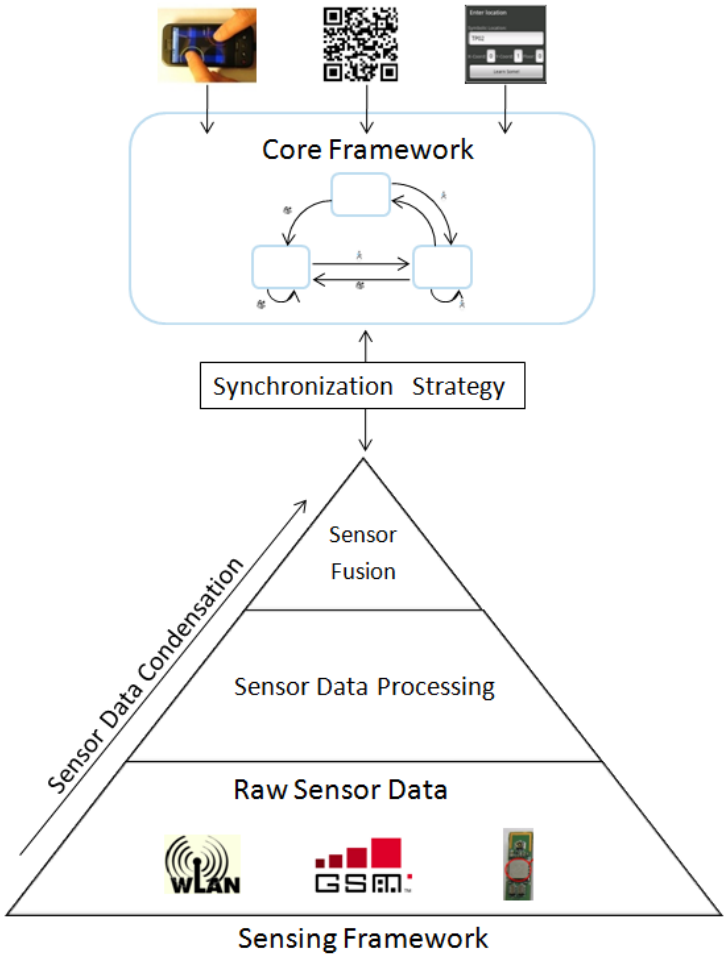


Abbildung 6.2: Architektur und Funktionsweise des Sensing-Frameworks. Dargestellt ist die Kondensierung der von den Sensoren gelieferten Rohdaten hin zu aufbereiteten Datenstrukturen, die zur Lokaliserung verwendet werden können. Dies geschieht durch die einzelnen Komponenten des Sensing, Data-Processing und der Sensor-Fusion. Weiterhin dargestellt ist die Ansteuerung des Sensing-Frameworks durch die FSM des Core-Frameworks, welche die Steuerung des Sensing-Frameworks anhand des aktuellen Systemzustands durchführt. Zusätzlich sind noch die vom SSF unterstützen Eingabemethoden zur Eingabe fixer Positionen und Bewegung dargestellt.

und Bewegungen dargestellt. Die dem Sensing-Framework zugehörigen Klassen befinden sich im Paket `de.ambisense.smartspace.sensing.*`.

6.2.1.1 Sensing

Die Sensing-Komponente entspricht der in „The Location Stack“ vorgestellten Sensor-Schicht und ist für die Kapselung der Sensoren und das Bereitstellen eines kontinuierlichen Stromes an Sensor-Rohdaten an die Data-Processing Komponente zuständig. Die Kapselung der Sensoren muss deshalb erfolgen, um innerhalb des Frameworks mit einer eigenen Sensor-Schnittstelle arbeiten zu können, die unabhängig von der darunterliegenden API des AAFs angesprochen und verwendet werden kann. Dies ist deshalb so wichtig, weil die meisten APIs in Android zwar als stabil markiert sind, diese aber nicht wie zum Beispiel beim iPhone eingefroren wurden. Das heißt, dass man sich nie sicher sein kann, ob sich eine bestimmte Schnittstelle in der Zukunft ändert oder um neue Funktionen ergänzt wird, welche die alte API ablösen.[2] Würde man also die konkreten API Schnittstellen direkt im SSF verwenden, müsste man gegebenenfalls bei jeder neuen Android Version Anpassungen am Framework-Code vornehmen. Es ist deshalb sinnvoll, eine Abstraktion zu wählen, die eine Kapselung der konkreten Android Schnittstellen vornimmt. Die Verwaltung der Sensoren wird deshalb über einen `SensorManager` implementiert. Beim Bootstraping des Frameworks wird auf diesem die Methode `initSensors()` aufgerufen, welche alle verfügbaren und im Konfigurationsobjekt zur Verwendung markierten Sensoren auswählt und initialisiert. Die Methoden `startSensors()` und `stopSensors()` des `SensorManager` werden jeweils von der FSM aufgerufen, um die Sensoren zu starten oder zu stoppen. Ein Sensor wird innerhalb des SSFs durch die Klasse `AbstractSensorDevice` repräsentiert. Er kann über mehrere Scanner verfügen, welche über die Hardware einen Strom an Sensordaten anfordern und das Ergebnis in ein internes Format für Sensor-Events umwandeln und den Strom an die nächste Schicht weiterleiten. Beide Klassen werden als nächstes beschrieben.

6.2.1.1.1 AbstractSensorDevice Wie bereits angedeutet, unterstützt das SSF mehrere Sensoren. Zur Zeit implementierte Sensoren basieren auf den Technologien WLAN, GSM und IMU. Jede dieser Technologien wird im SSF durch einen eigenen Sensor repräsentiert. Das Sensor-Konzept stellt eine Abstraktion über die vom AAF bereitgestellten `SensorManager`, `TelephonyManager` und `WifiManager` Klassen bereit. Hier sieht man sehr anschaulich, warum eine Sensor-Abstraktion überhaupt eingeführt werden muss. Die API, mit denen die einzelnen Sensoren im AAF angesprochen werden, sind unterschiedlich. Vor allem die WLAN- und GSM-Implementierungen, die im Rahmen dieser Arbeit auch als Sensoren angesehen werden, sind im AAF jeweils über einen eigenen Manager in Android repräsentiert. Die IMUs müssen über den `SensorManager` von Android angesprochen werden.[3] Deshalb muss jeder im SSF verwendete Sensor von der abstrakten Klasse `AbstractSensorDevice` erben, um innerhalb des Frameworks verwendet werden zu können. `AbstractSensorDevice` kann vom Stereotyp als Interfacer Objekt nach [WBM02] bezeichnet werden. Wichtig ist, den Unterschied zwischen dem `AbstractSensorDevice` und einem von ihm verwendeten Scanner zu kennen. Repräsentiert ein `AbstractSensorDevice` nur einen Sensor, wie zum Beispiel WLAN an sich, findet die eigentliche Kapselung und Übersetzung der von Android gelieferten

[2] Seit Android 1.0 wird zwar gewährleistet, dass Änderungen in den Android API rückwärtskompatibel sind, trotzdem ist die neue Funktionalität nur über die Schnittstelle verfügbar.

[3] Dies macht aus Sicht des AAFs Sinn, ist aber aus Sicht des SSF nicht optimal.

Sensor-Daten in einem Scanner statt.

Ein Scanner ist eine Klasse, die immer innerhalb eines `AbstractSensorDevice` implementiert wird und von der Java `Runnable` Schnittstelle erbt. Ein Sensor kann einen oder mehrere Scanner haben. So können verschiedene Scanner für unterschiedliche Anwendungsfälle implementiert werden. Ein Scanner kapselt die Logik für den Zugriff auf die AAF-API, übernimmt die Übersetzung der empfangenen Sensordaten in ein SSF internes Event-Format und fügt sie der Warteschlange des `SensingReactor` hinzu, welcher die Sensordaten weiterverarbeitet. Wichtig ist, dass er die Java `Runnable` Schnittstelle implementiert, damit der Scanner vom `SensorManager` in einem eigenen Thread gestartet werden kann, um das Leistungsverhalten des Systems bei der Verwendung mehrerer Sensoren zu gewährleisten. So führt der im `SensorDevice80211` implementierte Scanner `Scanner80211Passive` einen Scan nach verfügbaren WLANs durch. Anschließend wandelt er das vom Android `WifiManager` erhaltene Array mit allen Scan-Ergebnissen in Form von (`ScanResult`) Objekten in das SSF interne Format `ScanResult80211` um. Als letztes generiert der Scanner einen `SensorEvent`, welcher eine Liste an `ScanResult80211` Objekten enthält und diese als Sensor-Ereignis in die systemweite Warteschlange des `SensingReactor` schreibt.

Die abstrakte Klasse `AbstractSensorDevice` ist in Listing 2 dargestellt. Die Methoden `getID()`, `getName()` und `getSensorRunnables()` sind direkt in der abstrakten Schnittstelle implementiert. Die abstrakten Methoden müssen beim Hinzufügen eines neuen Sensors in das SSF implementiert werden. Sie werden beim Laden der Sensoren vom `SensorLoader` aufgerufen, um den Sensor zu initialisieren und anschließend dem `SensorManager` hinzugefügt. Im `SensorLoader` wird hierfür das Template-Method-Entwurfsmuster verwendet, um nacheinander alle Sensoren automatisiert zu initialisieren. In der Methode `initSensorID()` und `initSensorName()` müssen jeweils die Felder `SENSOR_ID` und `SENSOR_NAME` gesetzt werden, um Name und ID zu initialisieren.[4] In der Methode `deviceAvailable()` muss ein Boolean-Wert zurückgegeben werden, welcher angibt, ob der Sensor verfügbar ist. In der Methode `initDevice()` muss der Sensor initialisiert werden. Im Falle von WLAN, unter Verwendung des Android `WifiManager`, könnten diese beiden Methoden wie in Listing 1 implementiert werden. So wird in der `deviceAvailable()` Methode eine Referenz auf den Android-System-Service `WifiManager` angefordert und eine Überprüfung durchgeführt, ob dieser vorhanden ist. Diese Methode wird sozusagen als Vorbedingung aufgerufen und stellt sicher, dass eine Referenz auf den `WifiManager` vorhanden ist. In der `initDevice()` Methode wird auf dem `WifiManager` geprüft, ob das WLAN auf dem ME auch aktiviert ist. Wenn nicht wird das WLAN über `enableWifi()` aktiviert. Die Methode `createRunnables()` muss so implementiert werden, dass sie Instanzen aller Scanner-Objekte erstellt, die verwendet werden sollen. Beim Starten der Sensoren werden die Instanzen vom `SensorManager` über die `getSensorRunnables()` angefordert. Die weiteren Methoden `registerEvents()`, `registerScanSamples()`, `registerFilters()` und `registerDBPersistance` dienen der Registrierung von Datenstrukturen in der SSF Registry.[5]

Die Vorteile bei der Verwendung der `AbstractSensorDevice` Klasse liegen vorrangig

[4] Es empfiehlt sich hierbei, die Werte aus dem Konfigurationsobjekt, welches später vorgestellt wird, zu verwenden.

[5] Die Registry Implementierung wird nicht im Detail vorgestellt und wurde nach dem Registry-Entwurfsmuster entworfen.

Algorithmus 1 Verfügbarkeitsprüfung innerhalb eines AbstractSensorDevice

```
@Override
public boolean deviceAvailable() {
  this.wifiManager = (WifiManager)
  androidCtx.getSystemService(Context.WIFI_SERVICE);
  return wifiManager == null ? false : true;
}

@Override
public void initDevice() {
  if (!wifiManager.isWifiEnabled())
    enableWifi();

    createRunnables();
}
```

in der Flexibilität und Erweiterbarkeit, die sie in das Framework einbringen. Durch Implementierung des abstrakten Sensors können effizient neue Sensoren zum SSF hinzugefügt werden, ohne sich um das interne Sensing-Subsystem kümmern zu müssen. Ein weiterer Vorteil ist, dass sich bei Änderungen der Schnittstelle des AAPs oder beim Hinzufügen neuer Funktionalitäten der Code nur an einer zentralen Stelle, dem Sensor, geändert werden muss, was die Wartbarkeit des Systems erleichtert. Weiterhin ist durch die Verwendung von einem Thread pro Sensor ein gutes Leistungsverhalten und die Skalierbarkeit im Hinblick auf eine große Anzahl verschiedener Scanner und ein hoher Durchsatz an Sensordaten gewährleistet.

6.2.1.1.2 SensingReactor Der `SensingReactor` ist ein Event-Demultiplexer und -Dispatcher, implementiert nach dem Reactor-Entwurfsmuster. Er übernimmt das Demultiplexing der ankommenden Sensor-Events und leitet den Event an den sensorspezifischen Event-Handler zur weiteren Verarbeitung der Sensordaten weiter. Der `SensingReactor` läuft in einem eigenen Thread und blockiert so lange, bis ein `SensorEvent`-Objekt von einem Scanner der Sensoren in die Warteschlange des `SensingReactor` geschrieben wird. Die Warteschlange muss aufgrund der diversen Sensing-Threads verschiedener Sensoren synchronisiert sein. Um dies zu gewährleisten, wird die im Paket `java.util.concurrent` bereits implementierte synchronisierte Warteschlange `ConcurrentLinkedQueue` verwendet. Scanner liefern einen kontinuierlichen Strom an Sensordaten an den `SensingReactor`, der die ankommenden Events aus der Warteschlange entnimmt und verarbeitet. Die Verarbeitung der Events findet im Reactor synchron statt und besteht darin, anhand des angelieferten Event-Typs einen geeigneten Event-Handler zur Daten-Prozessierung zu identifizieren und den Event an diesen weiterzuleiten (dispatchen).

6.2.1.1.3 AbstractSensorHandler Der `AbstractSensorHandler` ist die abstrakte Superklasse, die den Rahmen für einen Event-Handler vorgibt. Event-Handler werden direkt beim `SensingReactor` registriert und können direkt verwendet werden. Registriert werden die Event-Handler in der vorgestellten abstrakten Methode `registerEvents()` des `AbstractSensorDevice`. Listing 3 zeigt anschaulich die Registrierung der Event-Handler `EventHandlerGSM` und `RawDataHandlerGSM` der konkreten GSM-Sensor-Implemen-

Algorithmus 2 AbstractSensorDevice Klasse im Überblick

```
public abstract class AbstractSensorDevice {
  // ...Felder...
  public String getID() {
    // ...Implementierung...
  }

  public String getName() {
    // ...Implementierung...
  }

  public List<Runnable> getSensorRunnables() {
    // ...Implementierung...
  }

  public abstract boolean deviceAvailable();

  public abstract void initDevice();

  public abstract void createRunnables();

  public abstract void registerEvents(Dependencies dep);

  public abstract void registerScanSamples(ScanSampleProvider reg);

  public abstract void registerFilters(FilterProvider fProv);

  public abstract void registerDBPersistance(
      ScanSampleDBPersistanceProvider splDBPers);

  public abstract void initSensorName();

  public abstract void initSensorID();
}
```

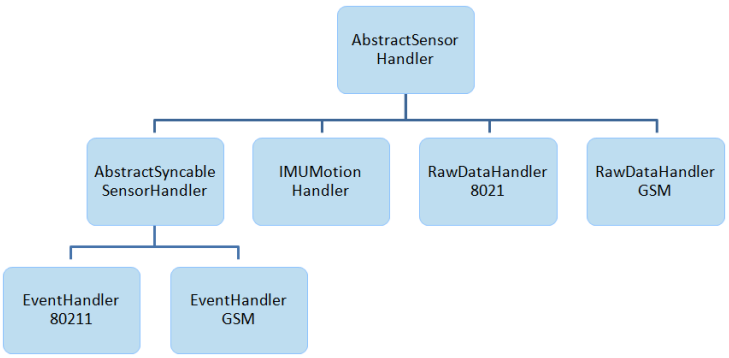


Abbildung 6.3: Klassendiagramm aller bereits im SSF implementierten Event-Handler.

Algorithmus 3 Registrieung eines Sensor-Event-Handlers

```
public void registerEvents(Dependencies dep) {
  ArrayList<AbstractSensorHandler> sensorHandlers = new ArrayList<
     AbstractSensorHandler>();
  if (Configuration.getInstance().sensorGSM.scannerGSMCells.isActive
     ) {
    sensorHandlers.add(new EventHandlerGSM(SENSOR_ID,
        GSM_CELL_SCAN_EVENT_ID,
        dep.sensorDependencies.eventSnychronizer));
  }
  if (Configuration.getInstance().persistence.mode ==
     SmartSpaceFramework.SCIENCE_MODE) {
    sensorHandlers.add(new RawDataHandlerGSM(SENSOR_ID,
        GSM_CELL_SCAN_EVENT_ID, dep.persistanceManager,
        dep.positionManager));
  }
  dep.sensorDependencies.reactor.registerHandler(
     GSM_CELL_SCAN_EVENT_ID,
      sensorHandlers);
}
```

tierung `SensorDeviceGSM`. Da beim Reactor mehrere Sensoren auf einmal registriert werden können, wird zuerst eine `ArrayList` erstellt und dieser dann den einzelnen Event-Handler hinzugefügt. Weiterhin wird in diesem Fall zusätzlich geprüft, welche Event-Handler im Konfigurationsobjekt zur Verwendung markiert sind und die einzelnen Handler werden dann beim Reactor registriert.

Konkrete Event-Handler, wie zum Beispiel `EventHandlerGSM`, müssen die abstrakte Methode `handleEvent()` implementieren, welche vom Reactor beim dispatchen der Events aufgerufen wird, um über die weitere Vorgehensweise zu entscheiden. Der `RawDataHandlerGSM` schreibt zum Beispiel alle in den Events enthaltenen Rohdaten direkt in einen eigenen Puffer und persistiert diesen periodisch in eine Datei auf dem Dateisystem. Der `IMUMotionHandler` des `SensorDeviceIMU` leitet dagegen die erhaltenen IMU-Sensor-Events direkt an den `MotionDetector` zur Bewegungserkennung weiter. Eine weitere Art der im SSF verwendeter Event-Handler sind `AbstractSyncableSensorHandler`, welche eine Synchronisation mit anderen Event-Handlern ermöglichen und im nächsten Abschnitt über Synchronisation besprochen werden. Abbildung 6.3 zeigt ein Klassendiagramm aller bereits im SSF implementierten Event-Handler und deren Zuordnung.

6.2.1.2 Synchronisation

SSF stellt eine Synchronisation der von den Sensoren gelieferten Daten bereit. Event-Handler, welche von `AbstractSyncableSensorHandler` erben, können bezüglich der Anzahl der von den Sensoren gelieferten Events oder der Messzeit synchronisiert werden. Dies bedeutet, dass pro Event-Handler, der eine Synchronisierung unterstützt, eine Obergrenze bezüglich der Anzahl der Messungen und/oder der Messzeit angegeben werden kann. Die Werte können pro Scanner im Konfigurationsobjekt gesetzt werden. So können Kriterien definiert werden, die erfüllt sein müssen, bevor die Sensordaten in den Data-Processing Klassen weiterverarbei-

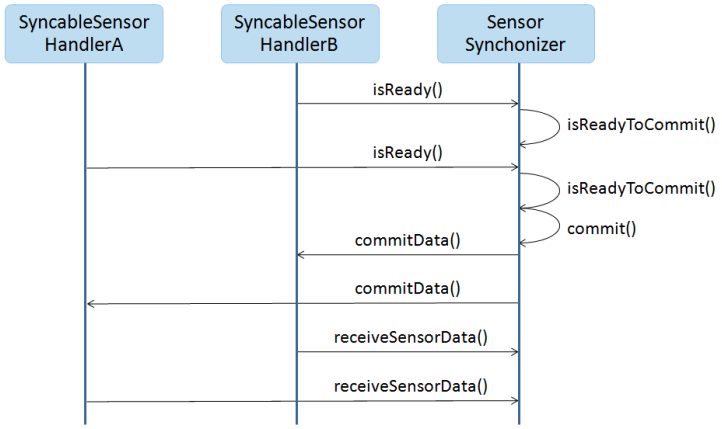


Abbildung 6.4: Sequenzdiagramm der Kollaboration von AbstractSyncableSensorHandler und EventSynchronizer.

Algorithmus 4 Syncable Schnittstelle zur Kollaboration zwischen AbstractSyncableSensorHandler und EventSynchronizer

```
public interface Syncable {
  public void setSyncronizer(EventSynchronizer sensorSynchronizer);
  public void commitData();
}
```

tet werden können. Regeln für Kriterien können wie nachfolgend definiert werden: Die Daten können erst dann weiterverarbeitet werden, wenn gilt:

- die Anzahl der empfangenen Events von $\langle SensorA \rangle$ an $\langle SyncableSensorHandlerA \rangle$ muss größer zehn sein,
- die Anzahl der empfangenen Events von $\langle SensorB \rangle$ an $\langle SyncableSensorHandlerB \rangle$ muss größer fünf und der Messzeitraum von $\langle SensorB \rangle$ größer sechzig Sekunden sein.

Es können also pro Event-Handler sowohl Regeln bezüglich der Messanzahl als auch der Messzeit definiert werden. Um aber eine sensorübergreifende Synchronisation nach den definierten Kriterien zu ermöglichen, muss eine Kollaboration zwischen den synchronisierbaren Event-Handlern und dem `EventSynchronizer` stattfinden. Damit die Synchronisierung stattfinden kann, muss vom Event-Handler die in Listing 4 dargestellte Schnittstelle `Syncable` implementiert werden. Das Sequenzdiagramm der Kollaboration der beiden Objekte ist in Abbildung 6.4 dargestellt. Gesteuert wird der gesamte Vorgang vom `EventSynchronizer`. Dieser wartet, bis die zu synchronisierenden Event-Handler signalisiert haben, dass die jeweiligen Kriterien erfüllt sind. Dies geschieht durch den Aufruf der Methode `isReady()` des `EventSynchronizer`. Sobald alle Event-Handler bereit sind, die Daten weiterzureichen, ruft der `EventSynchronizer` auf allen zu synchronisierenden Event-Handlern die Methode `commitData()` der `Syncable` Schnittstelle auf. Woraufhin alle Event-Handler die gepufferten Sensordaten über die `receiveSensorData()` Methode des `EventSynchronizer` an diesen übermitteln. Allgemein kann dann ein Kriterium K zur Synchronisation der Event-Handler folgendermaßen ausgedrückt werden:

$$K = (R_{a1} \wedge R_{a2} \wedge ... \wedge R_{an}) \wedge (R_{b1} \wedge R_{b2} \wedge ... \wedge R_{bn}) \wedge ... \wedge (R_{m1} \wedge R_{m2} \wedge ... \wedge R_{mn})$$

Dabei steht R für Regel, mit $R \in \{\#Messungen, Messzeitraum\}$, n für die Anzahl der Regeln und m für die Anzahl der zu synchronisierenden Event-Handler.

6.2.1.3 Data-Processing

Das Data-Processing im SSF entspricht von den Verantwortlichkeiten in etwa der vorgestellten Measurement-Schicht des „Location Stacks“. So enthält diese Komponente Strukturen und Algorithmen zum Transkribieren der rohen Sensordaten und exportiert diese anhand eines Datenstroms an die Komponenten Motion-Detector, Persistenz und Positionierung.

Algorithmus 5 Schnittstelle der SynchronizerStrategy

```
public abstract class SynchronizerStrategy {

  // ...Felder...

  // ...Konstruktor...

  public List<String> getSyncSet() {
    // ...Implementierung...
  }

  public abstract HashMap<String, ScanSampleList> processData(
      Map<String, List<List<?>>> sensorData);

  public abstract SensorDataSample fusionate(
      HashMap<String, ScanSampleList> resultSamples);

  public abstract void deploySampleDataSample(SensorDataSample sDSpl
     );
}
```

6.2.1.3.1 EventSynchronizer und SynchronizerStrategy Das Prozessieren der vom `SensingReactor` angelieferten Sensordaten ist neben der Synchronisierung der Event-Handler die zweite Verantwortlichkeit des `EventSynchronizer`. Die Strategie zur Prozessierung der Sensordaten ist vom aktuellen Zustand des SSF und somit der FSM abhängig und wird deshalb von dieser gesteuert. Für die Umsetzung der Strategie auf dem `EventSynchronizer` wurde deshalb das Strategy-Entwurfsmuster gewählt. Die allgemeine Schnittstelle der Strategie ist durch die abstrakte Klasse `SynchronizerStrategy` realisiert und in Listing 5 aufgeführt. Eine `SynchronizerStrategy` setzt sich aus drei Methoden zusammen: `processData()`, `fusionate()` und `deployDataSample()`. Aktuell gibt es zwei konkrete Implementierungen der `SynchronizerStrategy` zur Verarbeitung der Sensordaten: eine Strategie für den „Learning-State“, die `LFPTSyncStrategy` und eine Strategie für den „Realtime-State“, die `LFPTSyncStrategy`. Da beide zahlreiche Gemeinsamkeiten haben, erben sie nicht direkt von `SynchronizerStrategy`, sondern von `SensingStrategy`.

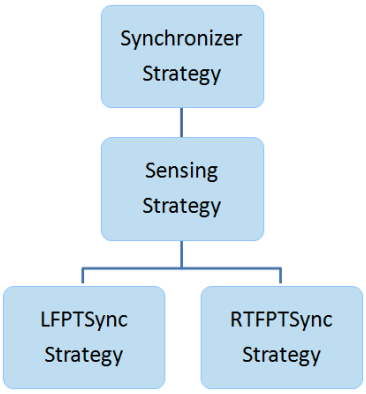


Abbildung 6.5: Vererbungshierarchie der Data-Processing Strategien.

Die Vererbungshierarchie der Strategie Klassen sind zur Verdeutlichung nochmals in Abbildung 6.5 dargestellt. Die Entscheidung, welche Strategie gewählt wird, trifft die FSM anhand ihres aktuellen Zustandes. So sieht die Strategie der FSM im „Learning-State“ folgendes vor:

- Verarbeitung der Sensordaten (`processData()`). In der Regel durch Berechnung von Mittelwert und Standardabweichung. Kann aber auch durch zusätzliche Filter wie Kalman Filter oder SSD ergänzt werden.
- Gegebenenfalls aufbereitete Sensordaten aus mehreren Sensoren zusammenführen (`fusionate()`).
- Persistenz des aktuellen LFPT über den `PersistanceManager` in der Datenbank.

Das beschriebene Verhalten muss von der konkreten Strategie, in diesem Falle der `LFPTSyncStrategy`, implementiert werden. Eine konkrete Strategie wird bei einem Zustandswechsel durch FSM über die Methode `setStrategy()` in den `EventSynchronizer` injiziert und von der FSM unter Verwendung des Template-Method-Entwurfsmusters aufgerufen. Der Vorteil bei der Verwendung einer Strategie wird deutlich, wenn zum Vergleich die `RFPTSyncStrategy` herangezogen wird. Diese Unterscheidet sich zum Beispiel in der `deployDataSample()` Methode. In der RTP muss im Gegensatz zur TP keine Persistenz[6] durchgeführt werden, sondern eine Lokalisierung. Deshalb muss in der `deployDataSample()` Methode zuerst die aktuelle RM aus der Datenbank geladen werden und anschließend die Positionierung über den `LocationProvider` initiiert werden. Eine Strategie ist somit ein sehr mächtiges Tool zur Beeinflussung des Verhalten des SeFs und ermöglicht es, das SSF leicht um neue Funktionalität zu erweitern. So kann eine Strategie einfach von einem Entwickler ausgetauscht und durch eine eigene Strategie ersetzt werden. Dabei muss nur von `SynchronizerStrategy` geerbt werden und die Strategie über die FSM in den `EventSynchronizer` injiziert werden. Zum Beispiel könnte die aktuelle für die Persistenz verwendete Komponente, welche die aufgenommenen LFPTs in der lokalen SQLite Datenbank speichert, auf einer Backend Struktur auf einem Server in der Cloud stattfinden.

[6]Wenn im Konfigurationsobjekt so festgelegt, kann in der aktuellen Implementierung der RFPTSyncStrategy, bei Ruhelage des MEs, auch eine Persistenz zur Autoadaption der RM durchgeführt werden.

Algorithmus 6 DataFilter Schnittstelle zur Realisierung von Filtern auf Sensordaten.

```
public interface DataFilter<I, O> {
  public void execute();
  public void setStdIn(I stdIn);
  public void setStdOut(O stdOut);
}
```

6.2.1.3.2 DataFilter Zur Verarbeitung der Daten können diverse Filter[7] implementiert werden. Listing 6 stellt die Schnittstelle dar, die implementiert werden muss, um einen Daten-Filter im SSF zu realisieren. Die DataFilter Schnittstelle umfasst drei Methoden. Die `execute()` Methode wird vom Framework im Laufe der Datenverarbeitung aufgerufen. In dieser Methode müssen die eigentliche Logik und die Algorithmen, die angewendet werden sollen, definiert werden. Die Methoden `setStdIn()` und `setStdOut()` werden ebenfalls vom Framework aufgerufen und setzen jeweils die aktuellen Sensoreingans- und Sensorausgangsdaten der Filter. Um die Schnittstelle besser zu verstehen, muss kurz der Aufbau des Data-Processing im SSF beschrieben werden.

Das SSF erlaubt es, mehrere Filter auf die gleichen Daten anzuwenden, diese nacheinander zu schalten und auf vorher schon prozessierte Daten auszuführen. Dies geschieht unter Zuhilfenahme des Command-Processor-Entwurfsmusters in Verbindung mit dem bereits beim Threading verwendeten „Pipes-and-Filters“-Prinzips. Es wurde eine Struktur entworfen, welche es erlaubt, Gruppen von Filtern in sogenannten „DataCommandChains“ zu definieren und diese sequentiell auszuführen. Der `DataProcessor` läuft in einem eigenen Thread und arbeitet intern mit einem `ThreadPoolExecutor` des `java.util.concurrent` Pakets. Der `DataProcessor` arbeitet dann alle Filter und Filtergruppen ab. Filter und Filtergruppen müssen bei der Initialisierung eines Sensors in der bereits in Kapitel 6.2.1.1.1 vorgestellten Methode `registerFilters()`, des `AbstractSensorDevice`, in der Registry registriert werden. Eine konkrete Implementierung eines Filters ist das `MeanDataCommand`, welches von WLAN und GSM in Form von `MeanCommand80211` und `MeanCommandGSM` Anwendung findet.

6.2.1.3.3 ScanSample und ScanSampleList `ScanSample` und `ScanSampleList` sind weitere bedeutende Klassen bei der Implementierung von Filtern. Filter beziehungsweise Filtergruppen müssen am Ende ihrer Datenverarbeitung immer eine Datenstruktur vom Typ `ScanSampleList`, eine Baumstruktur von `ScanSample` Objekten, zurückliefern. Die Eingangsdaten eines Filter oder einer Filtergruppe sind hingegen die vom `SensingReactor` gelieferten Rohdaten in Form von `ScanResult` Objekten. Jeder Sensor muss seine eigenen `ScanSample` Prototypen bei Initialisierung des `AbstractSensorDevice` über die Methode `registerScanSamples()` vorab bei der Registry registrieren. Eine Kopie des sensorspezifischen Prototyps wird vom `DataProcessor` aus der Registry abgerufen und in den Filter/Filtergruppe injiziert. Konkrete Implementierungen sind `ScanSample80211` oder `ScanSampleGSM`.

[7] In der aktuellen Version 1.4 des SSF heißen die Filter noch Data-Commands, dies soll aber mit einem Refactoring in Version 2.0 im Anschluss an diese Diplomarbeit geändert werden. Zum besseren Verständnis wird im Rahmen dieser Ausarbeitung deshalb schon von Filtern und nicht von Data-Commands gesprochen.

Ein `ScanSample`-Objekt modelliert letztendlich einen einzigen AP oder eine GSM-Zelle mit dazugehörigen sensorspezifischen Attributen[8] bei einem WLAN- oder GSM-Scan. Da aber, zum Beispiel bei einem WLAN-Scan, in der Regel mehr als nur ein AP empfangen wird, müssen die Filter immer eine `ScanSampleList` zurückgeben, auch wenn diese nur ein einziges `ScanSample` enthält. Die interne Organisation von `ScanSample` und `ScanSampleList` erfolgt durch eine Baumstruktur und die Klassen erben jeweils von der `Composite` Schnittstelle. Der Hintergrund, warum eine `ScanSampleList` ihre `ScanSample` in einer Baumstruktur hält, ist die spätere Verwendung bei der Datenfusionierung und der Verwendung innerhalb des LFPT-Verfahrens. Details und die Struktur der `Composite` Schnittstelle werden im nächsten Abschnitt zur Sensor-Fusion beschrieben.

6.2.1.4 Sensor-Fusion

Die Sensor-Fusion hat zwei Verantwortlichkeiten. Zum einen die Verantwortlichkeit, Daten von verschiedene Sensoren in einem LFPT zu kombinieren, zum anderen die Erstellung des LFPT in Form eines `SensorDataSample` Objekts. Die Fusion der Daten wird im SSF konkret in der Methode `fusionate()` der `SynchronizerStrategy` Schnittstelle realisiert. Der Vorgang ist sehr komplex und ist mit Hilfe des Composite und des Visitor Entwurfsmusters realisiert worden. Im Folgenden soll das zur Erstellung eines LFPTs verwendete Prinzip kurz vorgestellt werden.

Algorithmus 7 Composite Schnittstelle zur Umsetzung einer flexiblen Baumstruktur für dynamische LFPTs.

```
public interface Composite extends Component {
  public void add(String key, Component value);
  public Component get(String key);
}
```

6.2.1.4.1 SensorDataSample Ein `SensorDataSample` ist eine Datenstruktur, die beliebige Daten in Form der von den Filtern gelieferten `ScanSampleList` Objekte aufnehmen kann. Dies ist möglich, weil das `SensorDataSample` ebenfalls von der in Listing 7 dargestellten `Composite` Schnittstelle erbt. Einem `SensorDataSample` muss bei seiner Initialisierung im Konstruktor eine ID, ein Zeitstempel und der Datentyp bekannt gemacht werden. Sie ist somit eine sehr generische Datenstruktur, in der viele verschiedene Datentypen, wenn sie die `Composite` Schnittstelle implementieren, abgelegt werden können. Zur Modellierung eines LFPTs wird aber eine Spezialisierung benötigt. Diese ist durch die abstrakte Superklasse `FPT` umgesetzt und erweitert das `SensorDataSample` um allgemeine Fingerprinting spezifische Attribute. Die konkreten Implementierungen der für das LFPT-Verfahren verwendeten Klassen finden sich dann in `LFPT` und `RTFPT`.

Zu weiteren Details der Implementierung von Filtern, der Organisation und Hierarchie eines LFPT und den allgemeinen Konzepten des Sensing-Frameworks soll an dieser Stelle auf den Quellcode, der dieser Arbeit beiliegt, verwiesen werden. Anhand der Beispielimplementierungen der aktuellen WLAN- GSM- und IMU-Sensoren kann die Funktionsweise sehr schön nachvollzogen und gegebenenfalls erweitert werden.

[8] Dies sind in der Regel die Feldstärken, SSID, BSSID, CID und so weiter.

6.2.2 Core-Framework (CoF)

Das Core-Framework (CoF) beinhaltet die wichtigsten Komponenten des Frameworks außerhalb des SeFs. Es ist für die Persistenz der LFPTs, der unabhängigen und fusionierten Lokalisierung, der Bewegungserkennung und der Kommunikation mit den Anwendungen, die das Framework verwenden, zuständig. Es setzt sich aus den nachfolgenden Komponenten zusammen:

- Persistance,
- Localisation,
- Motion Detector,
- API.

6.2.2.1 Persistance

Die Persistenz-Komponente ist für das Datenbank- und Dateisystem-Management verantwortlich. So kapselt diese Komponente den Zugriff auf die darunterliegenden Subsysteme, wie das Dateisystem oder die von Android verwendete SQLite Datenbank. Die Subsysteme werden im weiteren Verlauf als Gateways bezeichnet. Alle an der Persistenz beteiligten Klassen liegen im Paket `de.ambisense.smartspace.android`. Die eigentliche Steuerung der Persistenz wird durch den `PersistanceManager` übernommen. Die Aufgaben der Persistenz-Komponente bestehen in der TRP in der Speicherung der LFPTs in der RM und in der RTP darin, ein Caching der RM vorzunehmen und diese bei Bedarf neu zu laden. Das Caching dient dazu, einen schnellen Zugriff für den Abgleich der RTFPTs bereitzustellen.

Algorithmus 8 Persistance Schnittstelle zum Laden, Speichern und Setzen einer Persistenz Strategie.

```
public interface Persistance {
  public void save(int pGateway, Object);
  public void load(int pGateway, String query);
  public Object get(int pGateway);
  public void setStrategy(int gateway, Object strategy);
}
```

6.2.2.1.1 PersistanceManager Der `PersistanceManager` ist nach dem Fassaden-Entwurfsmuster implementiert und nimmt die Stellung eines Vermittlers [WBM02] ein, welcher den Zugriff auf die darunterliegenden Persistenz Subsystem überwacht und steuert. Der `PersistanceManager` implementiert die `Persistance` Schnittstelle. Diese ist in Listing 8 dargestellt und stellt die Methoden `load()` und `save()` zum Laden und Speichern von Daten durch Angabe eines Gateways bereit. Zusätzlich kann ein Gateway durch den Aufruf der Methode `get()` direkt angefordert werden. Darüber hinaus kann in mit `setStrategy()` eine optionale Gateway Persistenz Strategie auf einem Gateway gesetzt werden.

Algorithmus 9 Gateway Schnittstelle, welche von allen Subsystemen wie Datenbanken oder dem Dateisystem implementiert werden muss.

```
public interface Gateway {
  public void setStrategy(Object Strategy);
  public boolean isLoaded();
}
```

6.2.2.1.2 Gateway Ein Gateway ist die im SSF verwendete Repräsentation genau eines Subsystems. Ein Gateway muss, um innerhalb des `PersistanceManager` verwendet werden zu können, die `Gateway` Schnittstelle aus Listing 9 implementieren. Bereits implementierte konkrete Gatewayumsetzungen sind unter anderen der für das LFPT-Verfahren verwendete `LFPTGateway` oder der `FileGateway` zum Speichern in Dateien des Dateisystems. Letzterer wird zum Beispiel zum Schreiben der gemessenen Rohdaten auf das Dateisystem verwendet.

Ein Gateway kann auch eine eigene Strategie zur Persistenz verwenden und ist mit Hilfe des Strategy-Musters implementiert. Ein Datenbank-Gateway kann durch eine geänderte Strategie je nach Situation ein anderes Verhalten bezüglich der auf der Datenbank ausgeführten Operationen aufweisen. Um eine eigene Persistenz-Strategie für einen Gateway zu verwenden, muss die in Listing 10 dargestellte `GatewayStrategy` Schnittstelle implementiert werden. Die Verwendung einer Strategie macht vor allem dann Sinn, wenn auf dem Gateway Beispiel SQL-Befehle ausgeführt werden müssen, die sich je nach Zustand dynamisch ändern.

Algorithmus 10 GatewayStrategy Schnittstelle

```
public interface GatewayStrategy {
  public void save(Object data);
  public void loadEntity();
  public Object getEntity();
}
```

Eine konkrete Implementierung einer solchen Strategie findet sich in `AndroidLocalDB-PersistanceStrategy`. Diese Strategie hat das Ziel, das Schreiben und Lesen von LFPTs in der Datenbank durchzuführen. Damit sie aber mit verschiedenen LFPT-Techniken (GSM, WLAN) umgehen kann, verwendet sie nicht konkret implementierte SQL-Befehle, sondern lädt diese dynamisch anhand des LFPT-Typs aus der Registry. Diese in der Registry hinterlegten Objekte zur Kapselung der SQL-Befehle müssen direkt im `AbstractSensorDevice` in der Methode `registerDBPersistance()` registriert werden. Die Klasse `ScanSample-80211DBAdapter` zeigt eine konkrete Implementierung eines Adapters zur Kapselung der SQL-Befehle, die zum Erzeugen und Speichern von WLAN-LFPT benötigt werden. Für weitere Details sei auch an dieser Stelle auf den Quelltext der Implementierung verwiesen.

6.2.2.2 Localisation

6.2.2.2.1 LocationProvider Die Komponenten im SSF, welche die eigentliche Lokalisierung nach dem LFPT-Verfahren durchführen, werden im SSF durch Klassen verwirklicht, welche die in Listing 11 dargestellte Schnittstelle `LocationProvider` implementieren. Die

Algorithmus 11 LocationProvider Schnittstelle

```
public interface LocationProvider {
  public void setRadioMap(RadioMap rm);
  public void calculateLocation(Object data);
  public List<?> getEstimatedLocations();
}
```

Schnittstelle umfasst drei Methoden. Über `setRadioMap()` kann die aktuelle RM geladen werden. In der RTP wird dann vom SSF auf allen im System vorhandenen `LocationProvider` Objekten die Methode `calculateLocation()` aufgerufen, welcher ein RTFPT übergeben wird. Anhand der RM und des übergebenen RTFPT kann in dieser Methode jeder beliebige Lokalisierungsalgorithmus ausgeführt werden. Die Methode `calculateLocation()` muss allerdings eine Liste an geschätzten Positionen zurückgeben, auch wenn nur eine Position vorliegt. Die Traversierung der Daten kann aufgrund der verwendeten Baumstruktur effizient mit einem Visitor-Entwurfsmuster durchgeführt werden. Schließlich wird, wenn die Methode `calculateLocation()` zurückkehrt, die Methode `getEstimatedLocations()` aufgerufen, um die aktuellen Positionen abzufragen. Die Steuerung dieser Vorgänge wird durchgängig von der FSM durchgeführt, die zu jeder Zeit die volle Kontrolle über den Zustand des SSF behält.

6.2.2.2.2 iLocationManager Der `iLocationManager` hat die Verantwortlichkeiten, die von den unterstützten markerbasierten, displaybasierten und kontinuierlichen Lokalisierungsmodulen gelieferten Positionen zu verwalten und dient darüber hinaus als Vermittler zwischen den API-Interfacer-Objekten [WBM02] und dem CoF.

Alle Module müssen, nachdem sie eine Position als Eingabe erhalten oder berechnet haben, diese dem `iLocationManager` bekannt machen. Dies kann auf zwei Wegen geschehen: Liegt nur eine Position vor, wie zum Beispiel beim Scannen eines QR-Codes, kann die aktuelle Position auf dem `iLocationManager` über die Methode `setPosition()` direkt gesetzt werden. Liefert ein Modul dagegen eine Rangliste an Positionen zurück, so wie dies beim LFPT-Modul der Fall ist, kann die Liste an Positionen auch über die `enqueueEstimatedLocations()` auf dem `iLocationManager` gesetzt werden. Sobald der `iLocationManager` eine neue Position aus einem der Module erhält, wird diese Position umgehend an alle Klassen weitergeleitet, die sich für Positionsupdates beim `iLocationManager` registriert haben. Dieser Mechanismus ist über ein Listener-Konzept nach dem Observer-Entwurfsmuster realisiert. So registriert sich der von der API bereitgestellte `SSFLocationManager` vorab beim `iLocationManager`, um über Positionsupdates informiert zu werden und informiert anschließend die Applikation, die den SSF-Android-Service zur Indoor-Lokalisierung nutzt. Genauere Details hierzu befinden sich im Kapitel 6.2.2.6 zu der vom SSF bereitgestellten API.

6.2.2.3 Motion Detection

Für die Bewegungserkennung im SSF ist der `MotionDetector` zuständig. Dieser führt eine Bewegungserkennung anhand der auf allen Achsen gemessenen absoluten Beschleunigungen durch. Der `MotionDetector` läuft wie alle Komponenten des CoF in einem eigenen Thread. Die Beschleunigungsdaten des IMU Sensors werden dabei direkt vom `IMUMotionHandler`, nachdem dieser die Rohdaten vom `SensingReactor` erhalten hat, in die Warteschlange des

`MotionDetector` eingefügt. Dieser entnimmt die Werte aus der Warteschlange und führt in den Methoden `processMotionData()` und `detectMotion` die eigentliche Bewegungserkennung durch.

Zur Steuerung der Bewegungserkennung müssen über das Konfigurationsobjekt die zwei Schwellenwerte „Motion-Sensivity“ und „Queue-Capacity“ gesetzt werden. Die Motion-Sensivity *mtnSens* gibt an, ab welchem Schwellenwert der absoluten Beschleunigungen *absAcc*, Bewegung erkannt werden soll. Über die Queue-Capacity *n* kann zusätzlich die Größe der Warteschlange zur Bewegungserkennung festgelegt werden. Bewegung *mtn* wird vom `MotionDetector` genau dann erkannt, wenn gilt:

$$mtn = \frac{\sum_{i=0}^{n} absAcc}{n} > mtnSens$$

Wurde erkannt, dass sich das ME aktuell in Bewegung befindet, wird dies der FSM mitgeteilt. Diese hat sich beim Bootstraping des SSFs beim `MotionDetector` als `MotionListener` registriert, um über Bewegungen des ME informiert zu werden. Die FSM wechselt dann direkt in den Motion-State, indem nur der IMU Sensor zur Bewegungserkennung verwendet wird und verbleibt in diesem Zustand, bis sich das ME wieder in Ruhelage befindet. Die Aufnahme von LFPTs und die kontinuierliche Lokalisierung werden in diesem Zustand gestoppt, um Interferenzen vor allem im Learning-State zu vermeiden, welche unweigerlich zu einer Verfälschung der LFPTs in der RM führen würde. In zukünftigen Versionen des SSF ist geplant, in diesem Zustand eine relative Positionierung unter Zuhilfenahme der IMUs, anhand Schritterkennung und dem Dead-Reckoning Verfahren vorzunehmen.

6.2.2.4 Konfigurationsobjekt

SSF verwendet ein eigenes Konfigurationsobjekt `Configuration`. In diesem Objekt können die Umgebungsparameter für die Komponenten des CoF und für alle im System verfügbaren Sensoren gesetzt werden. Dabei ist jede konfigurierbare Komponente durch ein eigenes Objekt modelliert. Tabelle 6.2.2.4 gibt einen Überblick über die Konfiguration der CoF-Komponenten und deren Parametern. Die zahlreichen Parameter der einzelnen Sensoren sind nicht dargestellt und können direkt in der Implementierung des `Configuration` Objekts nachgeschlagen werden. Die gesetzten Parameter des Konfigurationsobjekts werden zur Laufzeit ausgelesen und ermöglichen so eine flexible und dynamische Lösung zur Steuerung des Verhaltens des SSF. Im nächsten Abschnitt zur API wird auch eine Möglichkeit vorgestellt, wie das Konfigurationsobjekt in Verbindung mit dem von Android bereitgestellten Preferences-Framework verwendet werden kann und so eine Konfiguration über die `SharedPreferences` von Android möglich ist.

6.2.2.5 Finite State Machine (FSM)

Die Steuerung des SSF erfolgt durch eine Finite State Machine (FSM). Ihre Verantwortlichkeiten entsprechen in etwa der vorgestellten Arrangement-Schicht des „Location Stacks“. Die FSM ist ein Zustandsautomat, welcher die Steuerung des SSF anhand der zwei Eingangsparameter, fixe Position und Bewegung, dynamisch steuert. Die Verwendung der FSM gewährleisten je nach Zustand der Umgebung eine adäquate Lokalisierung und übernimmt vor allem die Steuerung der kontinuierlichen Lokalisierung. Die dynamische Steuerung des SSF anhand

Komponente	Parameter	Verwendung
ConfigLocalization	euklideanDist	Euklidean-Distance Algorithmus
ConfigPersistence	dbName	Datenbankname
	mode	Modus
	bufferSize	LFPT-Puffer Größe
	refreshInterval	Ladeintervall der RM in der RTP
ConfigSensing	oriQuantCount	Quantisierung der Orientierung
ConfigSensing.LFPT	SYN_THREADPOOL_SIZE	Threadpool Größe Event-Synchronizer

Tabelle 6.1: Konfigurationen der CoF Komponenten und ihrer Parameter

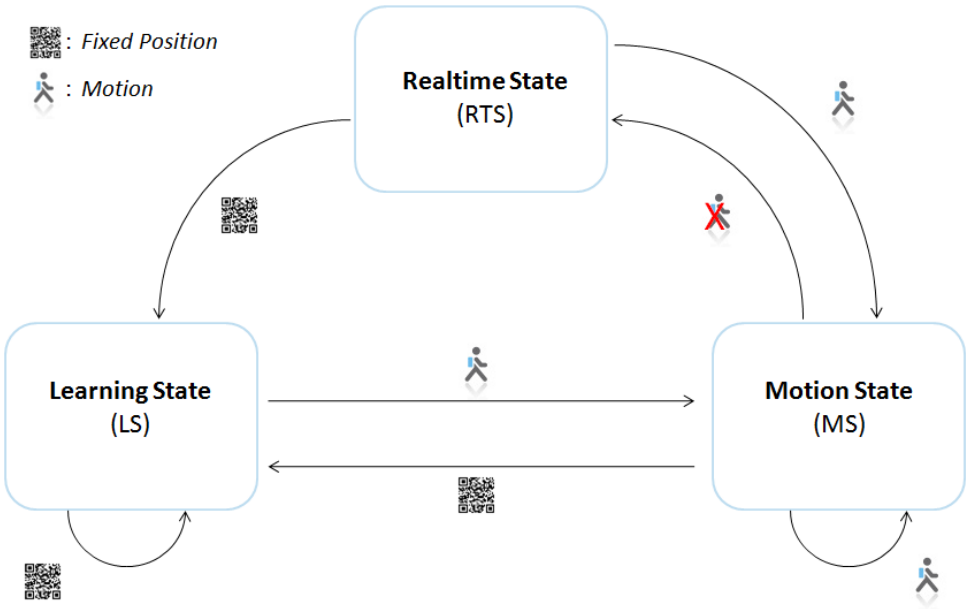


Abbildung 6.6: Funktionsweise der FSM zur Steuerung des SSF. Abgebildet sind die drei Zustände Realtime-State, Learning-State und Motion-State, sowie die Übergänge zwischen den einzelnen Zuständen anhand der Eingabeparameter der fixen Position und Bewegung.

von Umgebungsparamtern ist von großer Bedeutung, um den in Kapitel 4 vorgestellten Eigenschaften der RSS-Werte im Bezug auf optimiertes LFPT Rechnung zu tragen. So ermöglicht die Verwendung der FSM ein flexibles und trotzdem robustes Verhalten im Bezug auf Änderungen der Signalcharakteristik.

Die genaue Funktionsweise der FSM ist in Abbildung 6.6 dargestellt. Es werden insgesamt drei Zustände Unterschieden. Learning-State (LS), Realtime-State (RTS) und Motion-State (MS). Weiterhin gibt es zwei Eingabeparameter, die durch die Umgebung, in der sich das ME befindet, vorgegeben werden. Die Eingabeparameter sind die in Kapitel 5.3.2 beschriebenen Parameter fixe Position und Bewegung. Anhand dieser Parameter wird über einen Zustandsübergang innerhalb der FSM entschieden. Die Übergänge zwischen den einzelnen Zuständen sind abhängig von den anliegenden Eingabeparametern. Hierbei ist es wesentlich zu verstehen, dass die Parameter nicht von der FSM beeinflusst werden können, sondern die Zustandsübergänge nur eine Reaktion auf Signale aus der Umwelt sind, in der sich das ME befindet. Ist keine Position bekannt, wechselt die FSM in den RTS und versucht mit der vorgestellten LFPT Technik eine Lokalisierung des ME durchzuführen. Voraussetzung ist, das bereits eine RM vorhanden ist, die vom SSF zur Lokalisierung im RTS verwendet werden kann. Immer wenn eine Position mit hoher Genauigkeit bekannt ist, zum Beispiel durch Eingabe einer fixen Position über eine der Eingabemethoden Dialog, QR-Codes oder Display, wechselt die FSM in den LS und startet die TRP zur Aufnahme von LFPTs der aktiven Technologien. Voraussetzung hierfür ist, das keine Bewegung anliegt. In diesem Zustand verweilt das SSF solange, bis sich das ME bewegt oder ein Stop-Signal an die FSM gesendet wird. Wird vom Motion Detector dagegen Bewegung erkannt, wechselt die FSM in den MS und verweilt dort, bis sich das ME wieder in Ruhelage befindet. Sobald die FSM dies erkennt, wird wieder in den RTS gewechselt, weil durch die Bewegung davon ausgegangen werden muss, dass sich das ME nicht mehr an der eingegebenen fixen Position befindet und eine weitere Aufnahme von LFPTs die RM verfälschen könnte.

Algorithmus 12 State Schnittstelle

```
public interface StateMachine {
  public boolean checkForTransition();
  public void doTransition();
}
```

Die `FSM` implementiert die drei Schnittstellen `StateMachine`, `iLocationListener` und `MotionListener`. Die `StateMachine` Schnittstelle ist in Listing 12 dargestellt und enthält die zwei Methoden `checkForTransition()` und `doTransition()`. Die Methode `checkForTransition()` überprüft aufgrund des Zustands der anliegenden Eingabeparameter `fPos` und `mtn`, ob ein Zustandswechsel durchgeführt werden soll und stellt die Vorbedingung beim Aufruf von `checkForTransition()` dar. Wird festgestellt, dass ein Zustandswechsel ausgeführt werden soll, wird der nächste Zustand anhand einer Fallunterscheidung im aktuellen Zustand bestimmt und eine Referenz auf den nächsten Zustand in das Feld `nextState` gespeichert. Danach kann die Methode `doTransition()` aufgerufen werden. Diese ersetzt dann einfach die Referenz im Feld `currentState` durch `nextState` und ruft dann mehrere Methoden der `State` Schnittstelle auf, um einen Zustandswechsel durchzuführen. Die beiden Methoden `checkForTransition()` und `doTransition()` werden immer dann aufgerufen, wenn die FSM über Positions- und Bewegungsupdates informiert

wird. Dies geschieht genau dann, wenn einer der beiden Listener für die Eingabeparameter zurückgerufen wird und auf der FSM intern die Methoden fixedPositionAction() oder motionAction() aufgerufen werden. Die beiden verwendeten Listener werden beim Bootstraping des SSF von der FSM beim iLocationManager und dem MotionDetector initial registriert. Alle Klassen der FSM finden sich im Paket de.ambisense.smartspace.fsm.

Algorithmus 13 State Schnittstelle

```
public interface State {
  public void enterState(Dependencies dep);
  public void doActivity();
  public void exitState();
  public State switchNextState(boolean pos, boolean mtn);
}
```

State Die Arbeit der FSM besteht darin, den aktuellen Zustand des SSF anhand von Umgebungsparamtern zu bestimmen. Zustände werden im SSF durch das Implementieren der State Schnittstelle umgesetzt. Die Schnittstelle umfasst vier Methoden, welche die FSM in der Folge aufruft, wie Sie in Listing 13 dargestellt sind. Die Methoden enterState() und exitState() werden immer genau dann aufgerufen, wenn die FSM in einen neuen Zustand wechselt beziehungsweise einen Zustand verlässt. Diese Methoden eignen sich für eine Initialisierung und zum Shutdown von Komponenten. Die Methode doActivity() wird nach enterState() aufgerufen und ist dazu gedacht, Aktivitäten in einer Schleife auszuführen, zum Beispiel das Aufnehmen von LFPT in der TRP. Die letzte Methode switchNextState() wird von der FSM immer vor einem Zustandswechsel aufgerufen, um den nächsten Zustand zu bestimmen.

Durch implementieren der State Schnittstelle können Entwickler auch neue, eigene Zustände in das SSF integrieren. Hierzu können die bereits implementierten Zustände InitialState, LearningState, RealtimeState, MotionState oder FinalState als Referenz herangezogen werden.

6.2.2.6 API

Das SSF Framework ermöglicht es, mit Applikationen der Android Plattform Indoor-Lokalisierungs Dienste zu nutzen, ohne sich um interne Details der Lokalisierung kümmern zu müssen. Das SSF ist als ein Android-Service implementiert, der im Hintergrund seine Arbeit verrichtet. Entwickler können das SSF einfach durch binden des Services in ihrer Applikation beziehungsweise ihrer Activity verwenden. Die bereitgestellte API des SSF ist sehr schlicht und ist einfach zu bedienen. Die folgenden Erläuterungen sind vor allem als Dokumentation für Entwickler gedacht, die das SSF in einer ihrer Applikationen benutzen möchten.

Um das SSF in einer Applikation verwenden zu können, müssen in einem ersten Schritt die Listener-Schnittstellen iLocationListener und LogListener in Form von anonymen inneren Klassen implementiert werden. Momentan können zwei Listener, iLocationListener für Positionsupdates und LogListener zum Empfangen von Debug- und Log-Informationen, registriert werden. Die Position wird jeweils in Form eines IGeoPoint zurück-

geliefert.

6.2.2.6.1 IGeoPoint `IGeoPoint` ist eine Datenstruktur, welche das vom SSF verwendete Datenformat in einem Objekt kapselt und die interne Repräsentation einer Indoor-Geo-Position darstellt. Über den `IGeoPoint` lassen sich die aktuelle x- und y-Position, das Stockwerk und der Name der aktuellen Position abfragen. Zudem verfügt die `IGeoPoint` Klasse über drei Klassenmethoden zum Importieren und Exportieren des SSF-Daten Formats. Unterstützt werden Strings und zusätzlich ein JSON-Export, um die aktuelle Position an den REST-Service eines Servers zu schicken.

Algorithmus 14 iLocationListener Schnittstelle. Diese Schnittstelle muss implementiert werden, damit Positionsupdates vom SSF empfangen werden können.

```
public interface iLocationListener {

  void onLocationChanged(IGeoPoint location, Accuracy acc);

  void onLocationChanged(List<IGeoPoint> list, Accuracy acc);

  void onStateChanged(int state);
}
```

6.2.2.6.2 iLocationListener Der `iLocationListener` ist die wichtigste Komponente, um Indoor-Lokalisierungs-Informationen über das SSF zu erhalten. Damit der `iLocationListener` innerhalb des SSFs verwendet werden kann, muss sein Verhalten vorab in einer anonymen inneren Klasse implementiert werden. Die Schnittstelle des `iLocationListener` ist in Listing 14 dargestellt und verfügt über drei Methoden. Die `onLocationChanged()` Methode wird genau dann aufgerufen wenn ein der `SSFLocationMananger` eine neue Position vom SSF erhalten hat. Es gibt zwei `onLocationChanged()` Methoden, welche wahlweise mit nur einer Position in Form eines `IGeoPoint`-Objektes aufgerufen werden oder mit einer Liste von `IGeoPoint`-Objekten. Beide Methoden bekommen zusätzlich noch die Genauigkeit in Form eines `Accuracy` Objekts übergeben. Die dritte Methode `onStateChanged()` wird aufgerufen wenn die FSM ihren Zustand gewechselt hat und ermöglicht eine dynamisches Verhalten bezüglich des aktuellen Zustands der FSM zu implementieren.

Algorithmus 15 LogListener Schnittstelle. Diese Schnittstelle muss implementiert werden, um Debug- und Log-Informationen des SSF zu erhalten.

```
public interface LogListener {
  public void onLogMessage(String tag, String msg);
}
```

6.2.2.6.3 LogListener Der `LogListener` erlaubt es, sich für Debug- und Log-Informationen beim SSF zu registrieren. Um den `LogListener` verwenden zu können, muss wie beim

iLocationListener das Verhalten in einer anonymen inneren Klasse implementiert werden. Die LogListener Schnittstelle findet sich in Listing 15 und besitzt nur die Methode onLogMessage(), welche den Typ der Nachricht (Warnung, Fehler, Debug) und eine Inhalt übergeben bekommt. Ein LogListener wird direkt auf der Service-Instanz des SSF durch Aufruf der Methode registerLogListener() registriert und kann anschließend verwendet werden.

Algorithmus 16 Android-ServiceConnection Schnittstelle. Diese Schnittstelle muss implementiert werden, um das Verhalten bei Auf- und Abbau einer ServiceConnection zwischen der Activity und dem SSF-Service festzulegen.

```
private ServiceConnection ssfServiceConnection = new
   ServiceConnection() {
  @Override
  public void onServiceDisconnected(ComponentName name) {
    locationMngr.unregisterListener(iLocL);
    ssf.unregisterLogListener(logL);
    ssf.kill();
    ssfBinder = null;
    ssf = null;
  }

  @Override
  public void onServiceConnected(ComponentName name, IBinder service
     ) {
    ssfBinder = (SmartSpaceFramework.SSFBinder) service;
    ssf = ssfBinder.getSSF();
  }
};
```

6.2.2.6.4 ServiceConnection Nach Implementierung der Listener muss das Verhalten der Activity, die das SSF verwendet, bezüglich der Android ServiceConnection Schnittstelle, ebenfalls in einer anonymen inneren Klasse definiert werden. Wie dies genau auszusehen hat, wird in Listing 16 gezeigt. Hierbei wird die ServiceConnection in der Klassenvariable ssfServiceConnection abgespeichert. In der onServiceConnected() Methode wird das Verhalten beim Aufbau einer Verbindung zum Service implementiert. Mit der Zuweisung ssfBinder = (SmartSpaceFramework.SSFBinder) service; bekommt man eine Referenz auf den SSF-Binder. Anschließend wird direkt eine Referenz auf das SSF angefordert und in der Klassenvariable ssf gespeichert. Diese kann im weiteren Verlauf dazu benutzt werden, mit den API Methoden des SSF zu interagieren.

6.2.2.6.5 Verbindungsaufbau und Abbau Nachdem das Verhalten der ServiceConnection implementiert wurde, kann nun aktiv eine Verbindung zum SSF-Service aufgebaut werden. Wie dies genau funktioniert, wird in Listing 17 gezeigt. Zum Aufbau einer Verbindung wird in der Lebenszyklus Methode onCreate() der Activity die Methode bindService() aufgerufen. Nach Aufruf dieser Methode wird eine Verbindung zum SSF-Service aufgebaut

Algorithmus 17 Zeigt den Verbindungsaufbau zum SSF-Service in der onCreate() Methode einer Activity.

```
public void onCreate(Bundle savedInstanceState) {
  // ...Implementierung...
  Intent ssfServiceIntent = new Intent(getApplicationContext(),
      SmartSpaceFramework.class);
  bindService(ssfServiceIntent, ssfServiceConnection,
      Context.BIND_AUTO_CREATE);

    ssf.registerLogListener(logL);
    ssf.start();
    locationMngr = ssf
        .getLocationManager(SmartSpaceFramework.
           INDOOR_POSITION_PROVIDER);
    locationMngr.registerListener(iLocL);
  } catch (Exception e) {
      e.printStackTrace();
  }
}
```

und aufgrund der `ServiceConnection` Implementierung eine Referenz auf das SSF in der Klassenvariable `ssf` gespeichert.

Das SSF ist nun bereit, verwendet zu werden. Möchte man Debug- und Log-Informationen erhalten, muss nun zuerst der bereits implementierte `LogListener` registriert werden. Anschließend muss die Methode `start()` aufgerufen werden, um die FSM des SSF zu starten. Diese übernimmt nun die Kontrolle und steuert das SSF je nach Systemzustand. Schließlich muss zum Erhalt von Positionsänderungen noch der `iLocationListener` registriert werden. Hierzu muss über die Methode `getLocationManager()` eine Referenz auf den `SSFLocationManager` bezogen werden und dann auf diesem durch Aufruf von `registerListener()` der `iLocationListener` registriert werden. Sobald diese Schritte erledigt wurden, wird je nach den im Konfigurationsobjekt gesetzten Parametern die Lokalisierung ausgeführt. Die Details zum Verbindungsaufbau werden an dieser Stelle nicht genauer erläutert und können ebenfalls Listing 16 und Listing 17 entnommen werden.

6.2.2.6.6 ConfigTranslator Entwickler haben die Wahl, ob sie die Optionen zur Konfiguration der Sensoren und anderer Framework-Parameter durch den Nutzer veränderbar machen möchten oder nicht. Ist keine Modifikation gewünscht, können die Parameter direkt in der `Configuration` Klasse gesetzt werden. Ist eine Modifikation der Parameter durch den Nutzer gewünscht, ist es sinnvoll, diese in die Android `SharedPreferences` zu integrieren und einen Übersetzer zu schreiben, welcher die Werte der Android-Preferences in Parameter übersetzt, die dann direkt auf dem `Configuration` Objekt gesetzt werden. Um einen eigenen Übersetzer zu schreiben, muss von der in Listing 18 dargestellten abstrakten Klasse `ConfigTranslator` geerbt und die `translate()` Methode implementiert werden. Innerhalb der `translate()`-Methode müssen die Parametern auf den im Framework vorhandenen Konfigurationsobjekten gesetzt werden und in den Feldern der Superklasse abge-

speichert werden. Danach muss die `create()` aufgerufen werden, um die Werte in das SSF `Configuration` Objekt zu schreiben. Eine konkrete Implementierung eines Übersetzers findet sich in der Klasse `AndroidConfigTranslator`.

Algorithmus 18 Abstrakte Basisklasse von der geerbt werden muss, um einen ConfigTranslator für die Android SharedPreferences zu implementieren.

```
public abstract class ConfigTranslator {

  protected ConfigLocalization algos;
  protected ConfigPersistence persConf;
  protected ConfigSensing sensingConf;
  protected ConfigSensor80211 snsCfg80211;
  protected ConfigSensorGSM snsCfgGSM;
  protected ConfigSensorIMU snsCfgIMU;

  public void create() {
    Configuration.createConfiguration(algos, persConf, sensingConf,
        snsCfg80211, snsCfgGSM, snsCfgIMU);
  }
  public abstract void translate();
}
```

6.2.2.6.7 Verwendung von Eingabemodulen SSF unterstützt mehrere Eingabemethoden zur Eingabe fixer Positionen. Hierzu zählen die Module QR-Code, Dialog und Display.[9] Die Eingabemethoden für fixe Positionen sind wesentlich für die Verwendung der kontinuierlichen Lokalisierung im SSF. Die FSM braucht zur Aufnahme der initialen RM die exakte Position, um einen LFPT an einem Trainingspunkt aufnehmen zu können. Nachfolgend wird deshalb die Verwendung von Eingabemodulen am Beispiel des QR-Code Moduls vorgestellt. Alle Eingabemodule liegen im Paket `de.ilimitado.smartspace.android` und sind als eigene Activities realisiert.

Algorithmus 19 Zeigt die Verwendung der QR-Code Eingabemethode innerhalb einer Activity.

```
public void useQRInputMethod() {
  Intent intent = new Intent(
      "de.ambisense.smartspace.android.QRInputMethod");
  startActivity(intent, 0);
}
```

Die Verwendung von Eingabemodulen ist sehr einfach und in Listing 19 dargestellt. Da die Module als Activities implementiert sind, müssen diese einfach über einen Intent gestartet werden. Im Falle des `QRInputMethod`-Moduls wird der ZXing Barcodescanner [ZXIa] gestartet und das Ergebnis nach erfolgreichem Scan zunächst dem `iLocationManager` und schließlich der FSM bekannt gemacht, die daraufhin mit der Aufnahme von LFPTs beginnt.

[9] Die Eingabemethode über den Touchscreen eines Android basierten ME ist zum Zeitpunkt der Abgabe noch in Entwicklung und wird erst in Version 2.0 des SSF verwendet werden können.

6.2.2.6.8 Schlussbemerkungen zur Verwendung der API Um die kontinuierlichen Lokalisierung mittels des LFPT-Verfahrens in einer Applikation nutzen zu können, müssen folgende Schritte durchgeführt werden: Als erstes muss die initiale RM erstellt werden. Um die RM aufzunehmen, muss an verschiedenen Trainingspunkten, zu denen die Applikation sich später lokalisieren soll, ein LFPT aufgenommen werden. Zum Beispiel könnte man, wenn man eine Lokalisierung in den eigenen Räumlichkeiten durchführen möchte, LFPTs in der Küche, im Bad, im Arbeitszimmer und im Wohnzimmer aufnehmen. Zum Aufnehmen eines Trainingspunkts muss dem SSF eine Position mit einer Eingabemethode bekannt gemacht werden. Hierfür kann zum Beispiel über die Dialog oder QR-Code Eingabemethode eine Position eingegeben werden. Sobald die Eingabe der fixen Position erfolgt ist und in der Konfiguration entweder das WLAN oder GSM-LFPT-Verfahren ausgewählt ist, wechselt die FSM in den Learning-State und beginnt mit der Aufnahme der LFPTs. Nach erfolgreicher Aufnahme eines oder mehrerer TPs kann mit der eigentlichen Lokalisierung in der RTP begonnen werden.

Die komplette Beispielimplementierung der im API Abschnitt vorgestellten Code-Beispiele findet sich als ein Art Activity-Template in Listing D.1, im Anhang D. Diese kann von Entwickler in ihren eigenen Applikationen verwendet werden.

6.2.2.6.9 SmartSpace Demo-Applikation Um das SSF zu testen, kann die im Rahmen der Arbeit entwickelte Trainings-Applikation verwendet werden. Diese liegt dieser Arbeit in Form einer .apk Datei bei oder kann aus dem Quellcode zu dieser Arbeit selber compiliert und installiert werden. Mit ihr können beliebige LFPTs aufgenommen und nach Aufnahme der RM in der RTP eine Lokalisierung zu diesen durchgeführt werden. Über die Android `SharedPreferences` kann die Demo-Applikation konfiguriert und die verschiedenen Technologien wie WLAN, GSM und IMU ausprobiert werden. Die Demo-Applikation hat ein einfaches Command-Line-Interface, welches die Ausgaben des `LogListeners` auf dem Bildschirm ausgibt.

6.3 Fazit

Die hier vorgestellten Details der Implementierung geben einen Überblick darüber, wie die grundsätzlichen Ideen zum System-Design umgesetzt wurden. Des weiteren wurden die Grundlagen vermittelt, wie man das SSF für eine eigene Applikation zur Indoor-Lokalisierung verwenden kann. Dennoch wurden nur die wichtigsten Klassen und Konstrukte präsentiert. Für alle weiteren Details sei auf den Quellcode des SSF verwiesen, denn immerhin umfasst das SSF momentan ca. 9950 Lines of Code (LOC) davon etwa 6000 LOC Produktivcode und 4000 LOC Testcode.

Es bleibt festzuhalten, dass die Architektur des SSF aus zwei Subsysteme besteht, dem SeF und dem CoF. Das SeF dient zunächst der Kapselung der Sensoren als `AbstractSensorDevice`, welche alle in einem eigenen Thread Daten aufzeichnen. Sie stellen einen kontinuierlichen Strom an Rohdaten als Events in der Warteschlange des `SensingReactor` bereit, der auch in einem eigenen Thread läuft. Dieser dispatched die Daten zunächst an sensorspezifische Event-Handler, wo sie verarbeitet werden können. Die Event-Handler werden bei Bedarf nach konfigurierbaren Regeln im `EventSynchronizer` synchronisiert, um anschließend prozessiert, fusioniert und dann verwendet werden zu können. Dies steuert die FSM aus dem CoF Parameter abhängig und entscheidet wann und wie Daten aufgezeichnet, wie diese prozessiert und

ob sie dann zur Positionierung verwendet oder persistiert werden sollen. Für die Verarbeitung, Fusionierung und Verwendung der Sensordaten wird von der FSM eine Strategie, welche vom aktuellen Zustand des SSF abhängig ist, vorgegeben. Es können sowohl einfache als auch komplexere Algorithmen zur Datenverarbeitung verwendet werden, ohne dass das SSF blockiert wird, da all diese Algorithmen in Threads aus einem Pool von Worker-Threads abgearbeitet werden. Die Daten aus mehreren Sensoren müssen dann zu einem `SensorDataSample` fusioniert werden. Im letzten Schritt muss die Strategie der FSM darüber entscheiden, was mit dem `SensorDataSample` geschehen soll, also ob die nun fertig aufgezeichneten und aufbereitete Sensordaten vom `PersistanceManager` des CoF persistiert werden sollen oder ob das Sample zur Positionsbestimmung zum `LocationProvider` des CoF geleitet werden soll.

Zusammenfassend: das SeF zeichnet Daten, auf Wunsch synchronisiert, mit mehreren Sensoren auf, prozessiert die Daten, fusioniert sie zu einem `SensorDataSample` und stellt diesen zur weiteren Verwendung im CoF bereit. Das CoF besteht dabei zunächst aus der `FSM` zur Steuerung von zustandsabhängigem Verhalten, einem Konfigurationsobjekt zum dynamischen Anpassen des SSF an eigene Bedürfnisse, dem `PersistanceManager`, welcher den Zugriff auf sogenannte `Gateways`, welche wiederum den Datei- oder Datenbankzugriff abstrahieren, bereitstellt. Dazu kommt der `LocationProvider`, der den eigentlichen Lokalisierungs-Algorithmus ausführt und dem `iLocationManager` die aktuelle Position in Form eines `IGeoPoint` mitteilt. Eine weitere Komponente ist der `MotionDetector`, der feststellen kann, ob sich ein ME in Bewegung befindet oder nicht, was Auswirkungen auf die Zustände der `FSM` hat. Zur Verwendung des SSF in einer Android-Applikation können dann über die Registrierung des `iLocationListeners` und des `LogListerens` Positions-, Debug- und Log-Informationen bezogen werden.

Die Demo-Applikation ist ein guter Ansatzpunkt, um sich mit den Konzepten des SSF auseinanderzusetzen und anhand des Quellcodes und den in diesem Kapitel beschriebenen Anleitungen nachzuvollziehen, wie das SSF funktioniert und eingesetzt werden kann.

7 Evaluation

Dieses Kapitel stellt die Ergebnisse der im Laufe dieser Arbeit durchgeführten Messungen zur Leistungsevaluation des SSF vor. In einem ersten Teil wird auf die verwendeten Messkriterien eingegangen. Im zweiten Teil werden die konkreten Messergebnisse in Form von einzelnen Messreihen präsentiert.

7.1 Messkriterien

Um aussagekräftige Messdaten über die Performanz des implementierten SSFs zu bekommen, muss ein geeignetes Verfahren und ein strukturiertes Vorgehen festgelegt werden. Nur so kann gewährleistet werden, dass die aufgenommenen Daten für eine Auswertung herangezogen werden können. So wird die Genauigkeit und Zuverlässigkeit des Systems in Abhängigkeit von verschiedenen Faktoren bestimmt:

- verwendete Technologien,
- Umgebung (Stockwerke, Alltagsszenario),
- Anzahl der APs, Zellen-IDs (CIDs) pro TP.

Als erstes sollen die Unterschiede bei der Verwendung unterschiedlicher Technologien untersucht werden. Deshalb soll sowohl die Genauigkeit und die Zuverlässigkeit als auch der durchschnittliche Fehler bei der Lokalisierung von WLAN- und GSM-LFPT bestimmt und verglichen werden.

Die verwendeten Technologien sollen vor allem in Hinblick auf das Verhalten des SSFs beim Einsatz in verschiedenen Umgebungen getestet werden. Die einzelnen Testreihen sollen die System-Performanz in Hinblick auf einstöckige und mehrstöckige Installation untersuchen. Zudem soll die Messung in der einstöckigen Umgebung in einer sehr realitätsnahen Umgebung durchgeführt werden, um die Performanz in Bezug auf ein realistisches Alltagsszenario messen zu können.

Ein weiteres Kriterium, welches erfasst werden soll, ist die Performanz des SSFs im Hinblick auf seine Lokalisierungs-Geschwindigkeit und deren Bezug zur Größe der Installation. Die Geschwindigkeit, mit der eine Lokalisierung durchgeführt werden kann, ist von großer Bedeutung, da je nach Anwendung vom Benutzer nicht erwartet werden kann, für jede Lokalisierung für einige Sekunden oder Minuten stillzuhalten. Zudem sollte die Auswirkung der Anzahl der APs auf die Genauigkeit und Zuverlässigkeit des Systems hin untersucht werden.

7.1.1 Systemparameter

7.1.1.1 Messdichte

Die Messdichte entspricht der festgelegten Rastergröße für die Installation. Sie gibt an, in welchem Abstand in der TRP die Messungen durchgeführt werden. Die Rastergröße entspricht somit auch gleich der Auflösung des ILS. Nähere Informationen zur Rastergröße wurden in Kapitel 4.9.2 beschrieben. In jeder Messreihe wird deshalb eine feste Messdichte festgelegt, nach der die Messungen durchgeführt werden.

7.1.1.2 Größe der Installation

Pro Installation muss eine Größe und die Dimensionen der Installation angegeben werden. Die Größe wird in m^2 angeben. Anschließend muss anhand der Größe der Installation und der Messdichte die Anzahl der TPs und der RPs festgelegt werden.

7.1.1.3 Messwiederholungen

Die Messwiederholung gibt die absolute Anzahl an Messungen wieder, die zur Erstellung eines TPs aufgenommen wurden. Die Anzahl gibt letztendlich die Messfrequenz des ME wieder und muss anhand der gemessenen Anzahl an LFPTs berechnet werden. Die Anzahl der Messwiederholungen L pro LFPT kann wie folgt berechnet werden:

$$L = \#samples * bS$$

samples gibt die Anzahl der Messungen an, die von der ME durchgeführt werden, um eine Mittelung der Daten (Mittelwert und Standardabweichung) vorzunehmen. *bS* gibt an, wieviele der gemittelten Messungen aus *samples* durch erneute Bildung der Mittelwerte zu einem LFPT kondensiert wurden. Die gesamte Anzahl der Messwiederholungen M an einem TP ergibt sich aus den Messwiederholungen pro LFPT L und der Anzahl der aufgenommenen LFPTs pro TP c zu:

$$M = L * c$$

Prinzipiell gilt bei den Messwiederholungen: Je mehr Messungen pro TP durchgeführt werden, desto aussagekräftiger sind die daraus resultierenden LFPTs. Dies ist vor allem vor dem Hintergrund der in Kapitel 4.4.1.1 beschriebenen Auswirkung auf die Signalcharakteristik wichtig, um Ausreißern durch kurzzeitige Fluktuation der Feldstärken entgegenzuwirken.

7.1.1.4 Messzeitraum

Der Messzeitraum gibt den Zeitraum an, der verwendet wurde, um einen TP in der TRP aufzunehmen. Der Messzeitraum pro TP ergibt sich aus der Zeit zur Aufnahme eines LFPT und dem Messintervall (der Pause) zwischen der Aufnahme der einzelnen LFPTs. Der Messzeitraum hat starke Implikationen auf die Zeit, die zur Vermessung der gesamten Installation benötigt wird. Die Messdichte ergibt sich aus der Zeit t_{RM}, die zur Erstellung der RM aufgewendet werden muss und der Zeit t_{TP}, die pro TP Messung miteinbezogen werden muss, multipliziert mit der Anzahl der aufgenommenen TPs, TP:

$$t_{RM} = t_{TP} * \#TP$$

7.1.1.5 Orientierung

Wie in Kapitel 4.4.1.1 beschrieben, kann die Orientierung erheblichen Einfluss auf die Performanz eines ILS haben. Die Orientierung gibt deshalb immer an, welche Ausrichtung des ME zum Zeitpunkt der durchgeführten Messung vorlag. Diese muss bei der Aufnahme eines TPs immer dieselbe sein. Geringen Abweichungen wirkt man mit der vorgestellten Quantisierung der Orientierung, die in Kapitel 5.3.3 beschrieben wurde, entgegen. So werden nicht die absoluten Orientierungswerte, sondern nur die quantisierten Orientierungen verwendet. Die zusätzliche Aufnahme der n quantisierten Orientierungen ist mit einem hohen Mehraufwand verbunden und ver-n-facht die Zeit zur Erstellung der RM.

Da die Messungen zur Auswirkung der Orientierung auf die Performanz des SSFs nicht im Fokus dieser Messreihe standen, wurden die Ergebnisse der durchgeführten Messungen zur Orientierung schon in Kapitel 4.4.1.1 präsentiert. Die Orientierung wurde deshalb bei allen Messreihen dieser Auswertung mit Orientierung $ori0$ $(0-90^{\circ})$ durchgeführt.

7.1.2 Messverfahren

SSF verwendet zur Lokalisierung das vorgestellte LFPT-Verfahren mit WLAN- und GSM-Feldstärken. Als Algorithmus wird in den Messreihen das NNSS-Verfahren mit euklidischer Distanz verwendet. Des weiteren wird keinerlei aktive Infrastruktur benutzt: Das ME ist zu keinem Zeitpunkt mit einem AP in seiner Umgebung assoziiert. Bei der Verwendung von GSM muss das ME zu einer aktuellen Zelle verbunden sein, damit die Feldstärken der aktuellen und der Nachbarzellen gemessen werden können.[1] Weiterhin hat das ME keinerlei Kenntnis der physikalischen Position der gemessenen APs und Zellen. Die Messungen werden also rein passiv durchgeführt und es werden nur im Gebäude bereits installierte WLAN- oder von den Mobilfunkanbietern bereitgestellte GSM-Infrastruktur verwendet.

In einer ersten Phase wurde eine Vermessung der Dimensionen der Räumlichkeiten durch optische Distanzmessung mit einem Lasermessgerät vorgenommen. Die Vermessung der Räumlichkeiten wurde durchgeführt, weil der zur Verfügung stehende Gebäudeplan nicht mehr in allen Details den Räumlichkeiten entsprochen hat. Zudem lag zum Zeitpunkt der Messungen nur ein Gebäudeplan des zweiten Stockwerks vor, welcher geringe Abweichungen zum Erdgeschoss und dem ersten Stockwerk aufwies, die dokumentiert werden mussten, um die entsprechenden Gebäudepläne für diese Stockwerke zu erstellen. Nach Vermessung der Räumlichkeiten wurden die einzelnen TPs in Rastergröße vermessen und durch das Aufkleben eines Markers gekennzeichnet. Dies wurde für alle drei Stockwerke durchgeführt und die TPs auf den entsprechenden Gebäudeplänen eingezeichnet. Für die dritte Messreihe wurde eine eigenständige, wesentlich größere Installation verwendet, für die jeweils die vorgestellten Vermessungen durchgeführt wurden und zusätzlich ein eigener Gebäudeplan erstellt wurde. In beiden Fällen wurde das in Kapitel 5.3.3 vorgestellte SSF-Format zur Kodierung von physikalischen Referenzpositionen verwendet.

In der zweiten Phase wurden an allen TPs der Messreihe LFPTs nach den definierten Messparametern aufgenommen und durch Persistenz der LFPTs in einer Datenbank die RM zur

[1] Das Messen aller verfügbarer Zellen, aller Mobilfunkanbieter, wurde bis zur Fertigstellung dieser Arbeit vom Android Applikations Framework nicht unterstützt. So konnten immer nur die Nachbarzellen desselben Anbieters gescannt werden.

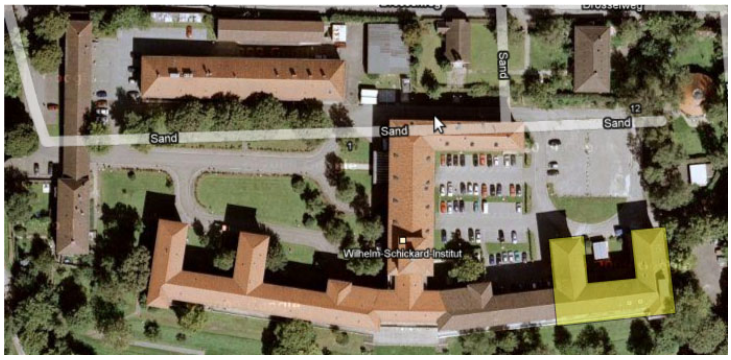


Abbildung 7.1: Luftaufnahme des Wilhelm-Schickard-Institutes der Universität Tübingen. Hervorgehoben ist der hintere Sand 13, in dem die Messreihen 1, 2 und 4 durchgeführt wurden.

späteren Lokalisierung erstellt. Das Aufnehmen eines LFPT an einem festgelegten TP wird wie folgt durchgeführt: Dem ME wird die aktuelle physikalische Referenzposition innerhalb der Messumgebung mitgeteilt, welches daraufhin mit der Aufnahme des LFPT beginnt. Um die Eingabe der Referenzpositionen der TPs zu vereinfachen, wird das vom SSF in Kapitel 5.3.4 beschriebene, mitgelieferte Modul zur Eingabe einer Position mittels eines QR-Codes verwendet. Die QR-Codes wurden hierzu mit einer eigens für diesen Zweck entwickelten Chrome-Extension, die auch auf der dieser Arbeit beiliegenden CD enthalten ist, für die komplette Installation automatisiert generiert. Zusätzlich wurde bei Durchführung der Messung jeweils der zum TP korrespondierende QR-Code anhand seines symbolischen Identifiers auf dem TP-Marker vermerkt, um eine spätere Zuordnung der Marker zu den physikalischen Positionen zu erleichtern. Zusätzlich zur Aufnahme der LFPTs wurden ebenfalls die Rohdaten zu jeder Messung aufgezeichnet und in die Auswertung mit einbezogen.

In einer dritten und letzten Phase wurden jeweils mehrere RPs zur Lokalisierung ausgewählt. Hierbei wurden sowohl Punkte zwischen den einzelnen TPs als auch direkt auf den TPs durchgeführt, um anhand dieser Daten die Performanz des Systems zu bestimmen. Anschließend wurden die Messungen zu der entsprechenden Messreihe durchgeführt und evaluiert. Zur Durchführung der Messungen bezüglich der definierten Kriterien kam die Version 1.4 des SSF auf der in der Arbeit verwendeten Hardware HTC G1 zum Einsatz. Die Messungen zur Aufnahme der RM und der anschließenden Lokalisierung wurden bei den Messreihen 1-3 jeweils in einem Abstand von $70\,cm$ vom Boden auf einem Podest aus Pappmaschee durchgeführt, um Interferenzen von den im Boden liegenden Kabeln, Rohrleitung (Abwasser, Wasser) oder eingelassenen Metallträgern zu unterbinden. In Messreihe 4 wurde das ME von einem Probanden verwendet. Hierzu stellte sich dieser über den zu untersuchenden TP und startete die Messung des RPs. Gemessen wurde immer über einen von der Messreihe vorgegebenen Messzeitraum. Anschließend wurde eine Auswertung der Positionsbestimmungen vorgenommen und anhand dieser Daten die endgültige Evaluierung im Hinblick auf die Genauigkeit und Zuverlässigkeit des Systems durchgeführt.

7.1.3 Umgebung

Als Testumgebung für die Messungen, die im Verlauf dieser Arbeit durchgeführt wurden, dienten die Räumlichkeiten des Wilhelm-Schickard-Instituts (WSI) an der Universität Tübin-

Abbildung 7.2: G1 des taiwanischen ME Herstellers High Tech Computer (HTC).

gen. Das Institut befindet sich im Tübinger Stadtteil Sand in einer ehemaligen Militärkaserne, die aus mehreren Gebäuden besteht und vor einigen Jahren von Grund auf saniert wurde. Der Gebäudekomplex verfügt über relativ dicke Decken und Mauern und ist in Abbildung 7.1 aus der Vogelperspektive dargestellt. Eingezeichnet ist der hintere Teil des Sand 13, in dem die Messreihen 1, 2 und 4 durchgeführt wurden. Die Messungen zur Messreihe 3 erstreckten sich über den gesamten Gebäudekomplex des WSIs.

Die Räumlichkeiten auf dem Sand 13 eignen sich sehr gut zur Durchführung der Testmessungen. Dieser Teil des WSIs verfügt über eine sehr gute Netzinfrastruktur in Bezug auf WLAN und GSM. So war die WLAN Abdeckungen bei den Messungen teilweise so hoch, dass pro TP bis zu 55 APs empfangen werden konnten. Des weiteren befindet sich in unmittelbarer Nähe des WSI ein Mobilfunkmast einer O2-BTS.[2] Weiterhin eignete sich das Gebäude Sand 13 für die Messreihen, weil die Messungen auf mehreren Stockwerken durchgeführt werden konnten, welche bis auf wenige Kleinigkeiten im Aufbau und der Aufteilung der einzelnen Räume gleichbleibend sind. Zudem bestand aufgrund der Tatsache, dass sich im dritten Stock des Sand 13 die Räumlichkeiten des Lehrstuhls befanden, an dem diese Arbeit verfasst wurde, die Möglichkeit, dort auch Messungen innerhalb von Räumen durchzuführen. Die Räumlichkeiten im Erdgeschoss und im ersten Stockwerk waren leider nicht zugänglich, deshalb beschränken sich die Messreihen 1 und 2, die sich auf alle drei Stockwerke beziehen, nur auf TPs in den Gängen vor den Räumen.

7.2 Verwendete Hardware

Zur Umsetzung des Frameworks und zur Durchführung der Testmessungen wurde in dieser Arbeit ein Testgerät vom Typ G1 verwendet. Das in Abbildung 7.2 dargestellte Gerät wurde vom taiwanischen Hersteller High Tech Computer (HTC) unter dem Codenamen HTC Dream 100 entwickelt, feierte im Oktober 2008 bei T-Mobile USA Weltpremiere als HTC G1 und ist seit März 2009 als erstes ME auf Android Basis auch in Deutschland erhältlich. Das G1 verfügt über mehrere Sensoren, die dazu benutzt werden können, ein ILS zu realisieren. Tabelle C.5 in Anhang C gibt einen detaillierten Überblick der Hardware-Spezifikation des G1.

[2] Zur Aufnahme der GSM Messungen wurde das O2 Netz verwendet.

7.3 Messreihen

Im Folgenden werden die Messreihen und Ergebnisse, die bei der Erhebung der Daten in Bezug auf die vorgestellten Kriterien entstanden sind, präsentiert. Um Aussagen über die Genauigkeit und Zuverlässigkeit eines ILS zu machen, benötigt man Metriken, die es erlauben, fundierte Aussagen zu treffen. Im Vergleich zu anderen Lokalisierungsverfahren wie der Trilateration basiert das LFPT Verfahren nicht auf der Berechnung von absoluten Entfernungsmaßen, die zur Performanz-Analyse verwendet werden können. Die Performanz-Analyse für LFPT Verfahren verfolgt deshalb einen quantitativen Analyseansatz. Die quantitative Analyse bezüglich der Genauigkeit und Zuverlässigkeit der RPs werden in der RTP durchgeführt. Zuerst werden die Treffer an den RPs quantitativ ausgewertet und anhand dieser Werte die Genauigkeit und Zuverlässigkeit bei der Positionsbestimmung der Messreihe bestimmt. Zusätzlich wird die mittlere Abweichung pro RP und der absolute Lokalisierungsfehler in Metern berechnet.

Da sich die Messungen zu den Messreihen 1 und 2 jeweils auf die gleichen Gebäudepläne mit denselben TPs und RPs beziehen, werden die Gebäudepläne für die Messreihen 1 und 2 hier vorab dargestellt. Als Grundlage für die Abbildungen auf allen drei Stockwerken dient ein Gebäudeplan des zweiten Stockwerks des hinteren Sand 13. Die Pläne des Erdgeschosses und des ersten Stockwerks lagen nicht vor. Deshalb mussten die Gebäudepläne für diese beiden Stockwerke grafisch angepasst werden, indem am unteren Ende der Gebäudeplan um einen Gang ergänzt wurde. Der Gang aus Stockwerk 2 existiert auf diesen Stockwerken nicht.

Die Gebäudepläne werden im Folgenden für alle drei Stockwerke des hinteren Sand 13 dargestellt und enthalten jeweils die TPs, die WLAN-RPs und die GSM-RPs in einer Grafik. Die verschiedenen Arten von Punkten unterscheiden sich in ihrer Farbe, wobei TPs in rot, WLAN-RPs in grün und GSM-RPs in blau dargestellt werden. Tabelle 7.3 zeigt eine Übersicht über die Zuordnung der Farben. Das Raster der Gebäudepläne ist jeweils in Orange über den Plan gelegt und ermöglicht so eine Zuordnung der Abstände der RPs und der gesamten Installation. Da sich das Raster je nach Messreihe unterscheidet, wird es in jeder Messreihe separat angegeben. Bei den Messreihen 3 und 4 werden die Gebäudepläne direkt im Abschnitt der Messreihe eingebunden, weil hier andere Gebäudepläne und unterschiedliche Referenzpositionen der TPs und RPs verwendet werden.

Punkt Typ	**Farbe**
TP	Rot
WLAN-RP	Grün
GSM-RP	Blau

Tabelle 7.1: Zuordnung der Farben zu ihren im Gebäudeplan korrespondierenden TPs, WLAN-RPs und GSM-RPs.

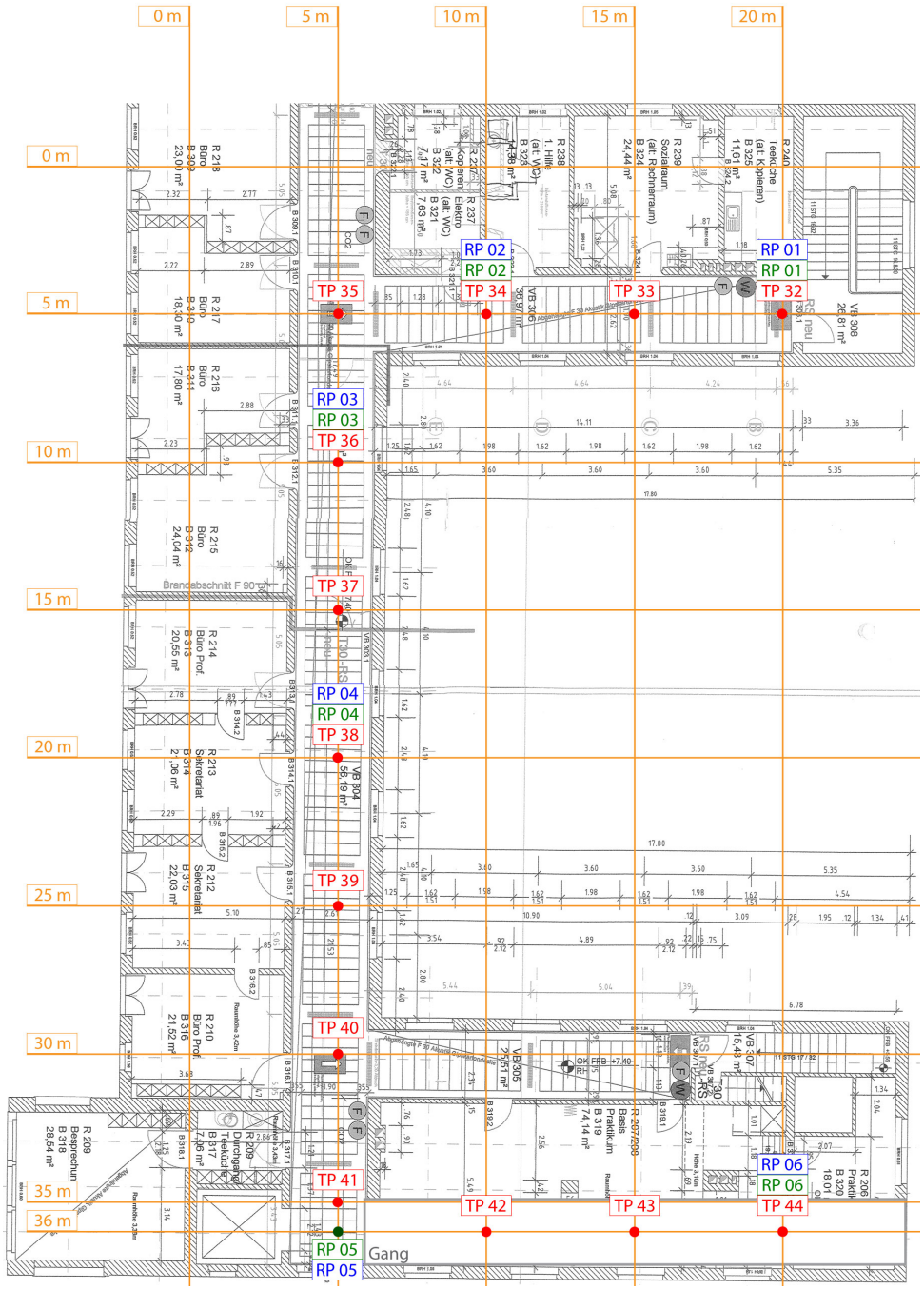


Abbildung 7.3: Gebäudeplan für die WLAN und GSM Messreihen auf allen Stockwerken. Zu sehen ist der Gebäudeplan des Erdgeschosses des hinteren Sand 13. Eingezeichnet sind die TPs (rot), WLAN-RPs (grün) und GSM-RPs (blau). Das in der Messreihe verwendete Raster ist durch die orangenen Linien gekennzeichnet. Im unteren Bereich wurde ein neuer Gang eingezeichnet, weil der Gebäudeplan im Erdgeschoss an dieser Stelle leicht abweicht.

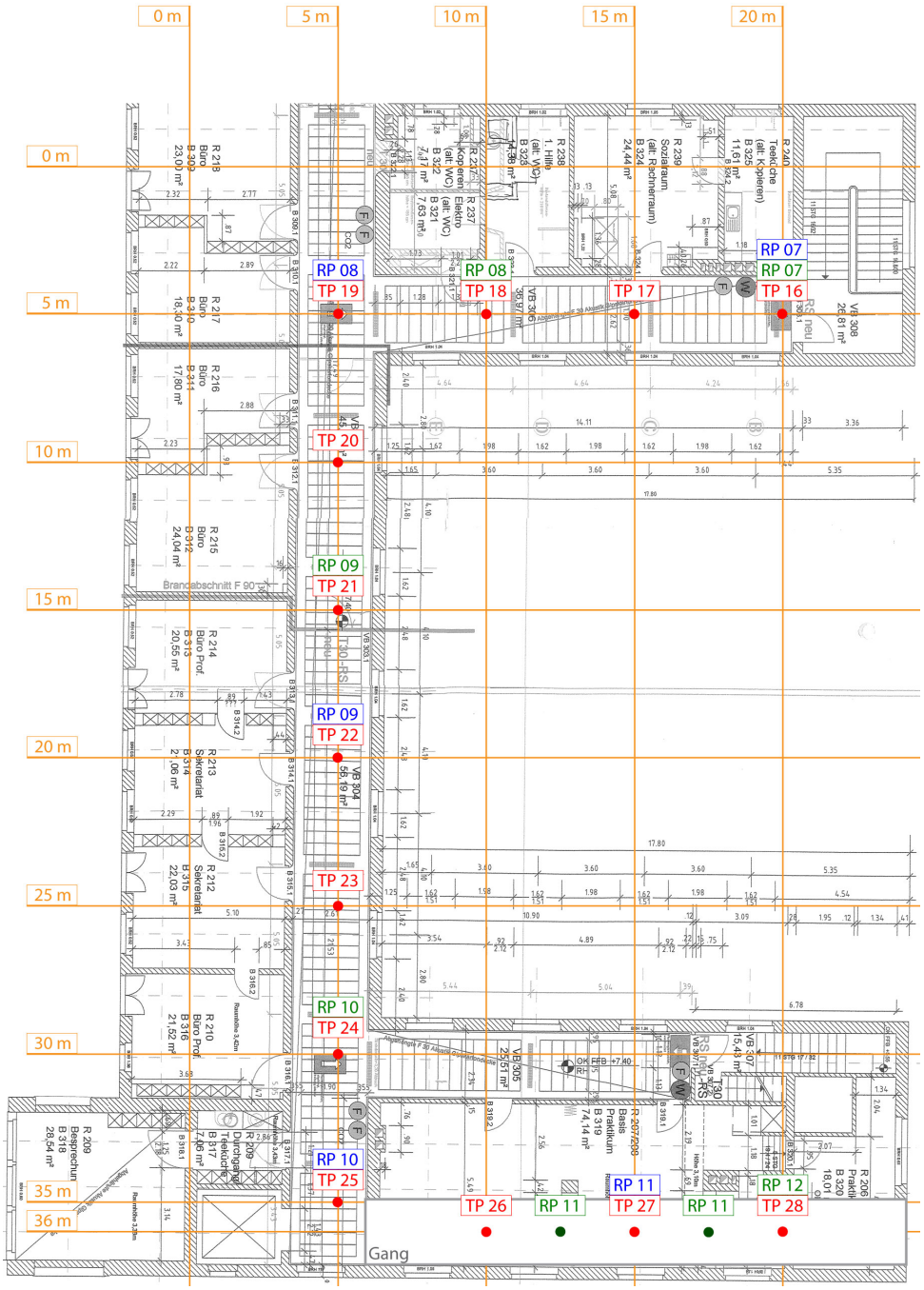


Abbildung 7.4: Gebäudeplan für die WLAN und GSM Messreihen auf allen Stockwerken. Zu sehen ist der Gebäudeplan des ersten Stockwerks des hinteren Sand 13. Eingezeichnet sind die TPs (rot), WLAN-RPs (grün) und GSM-RPs (blau). Das in der Messreihe verwendete Raster ist durch die orangenen Linien gekennzeichnet.

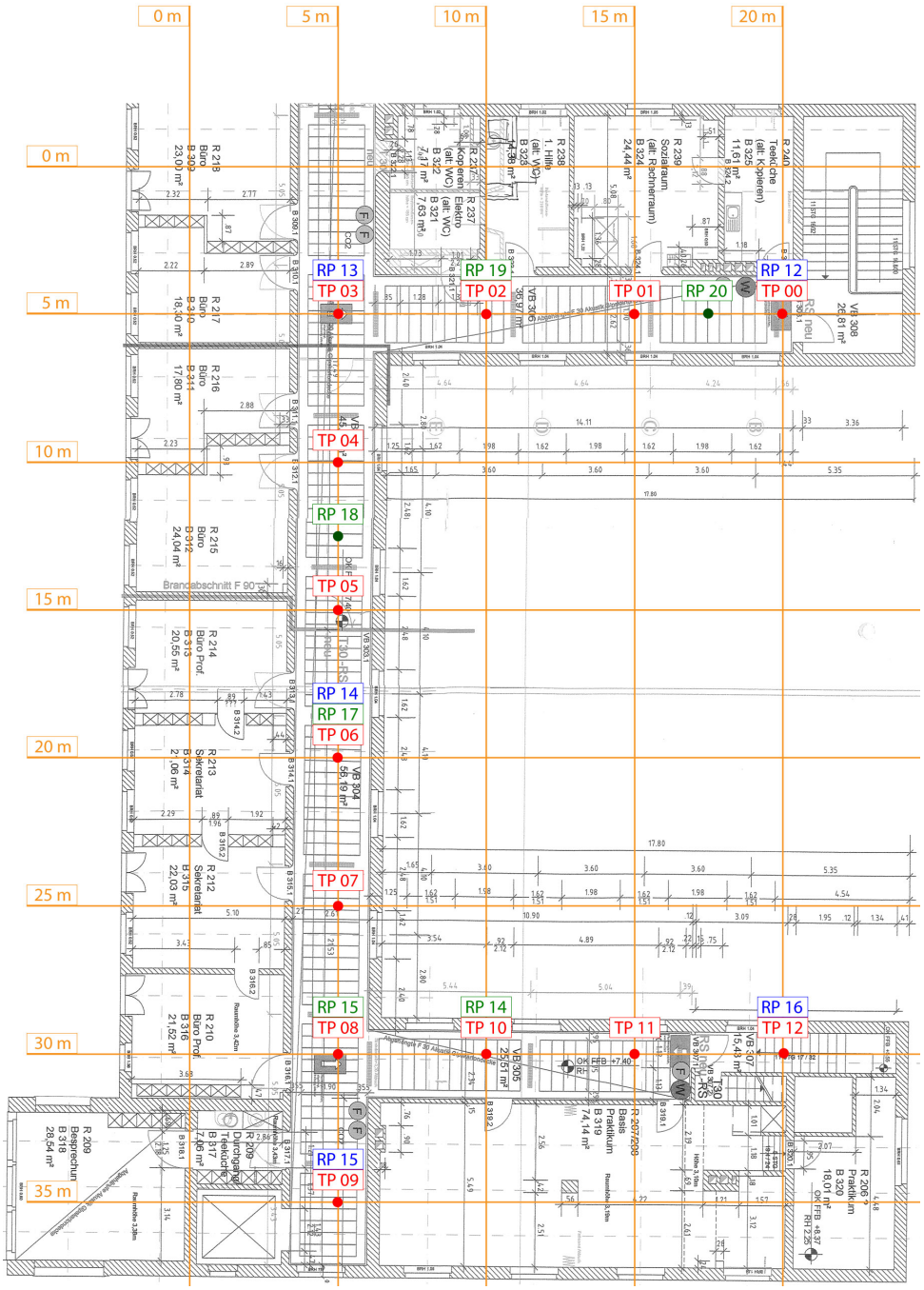


Abbildung 7.5: Gebäudeplan für die WLAN und GSM Messreihen auf allen Stockwerken. Zu sehen ist der Gebäudeplan des zweiten Stockwerk des hinteren Sand 13. Eingezeichnet sind die TPs (rot), WLAN-RPs (grün) und GSM-RPs (blau). Das in der Messreihe verwendete Raster ist durch die orangenen Linien gekennzeichnet.

7.3.1 Messreihe 1 – WLAN, 39 WLAN-TPs, 5x5 m, 3 Stockwerke

Die erste Messreihe wurde mit WLAN-LFPT auf allen drei Stockwerken des WSIs durchgeführt. Mit dieser Messreihe sollte die Performanz des SSF bezüglich des WLAN-LFPT-Verfahrens in mehrstöckigen Installationen getestet werden. Für die Messreihe wurden die folgenden Systemparameter gewählt:

7.3.1.1 Aufbau

7.3.1.1.1 Installation Die Größe der Installation betrug $133\,m^2$, als Messdichte wurde ein Raster von $5x5\,m$ gewählt. Aufgezeichnet wurde die WLAN-RSS aller verfügbarer APs in *dBm*.

7.3.1.1.2 TRP Die Anzahl der WLAN-TPs in der TRP betrug 39, an jedem WLAN-TP wurden 5 LFPTs aufgezeichnet. Das ergibt für die Anzahl der Messungen 125 ($M = \#LFPT * \#samples * bS$, mit $\#LFPT = 5, samples = 5, bS = 5$). Das Messintervall betrug $30\frac{s}{LFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0-90°)$ aufgezeichnet.

7.3.1.1.3 RTP Die Anzahl der WLAN-RPs in der RTP betrug 19, davon lagen 14 direkt auf einem WLAN-TP und 5 WLAN-RP befanden sich jeweils zwischen 2 WLAN-TPs. Das Messintervall betrug $60\frac{s}{RTFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0-90°)$ aufgezeichnet.

7.3.1.2 Ergebnisse

Die folgenden Abbildungen zeigen die Ergebnisse der Messreihe. Abbildung 7.6 zeigt die Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 19 WLAN-RPs für alle Stockwerke. Auf der x-Achse sind jeweils die Indizes der in den Gebäudeplänen (Abbildung 7.3, Abbildung 7.4, Abbildung 7.5) dargestellten WLAN-RPs aufgetragen. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch „Total“ (blau), die Anzahl erfolgreicher Positionsbestimmungen durch „Hit“ (grün) und die Anzahl falsch bestimmter Positionen „Miss“ (rot) dargestellt. Die Werte zeigen eine hohe Zuverlässigkeit bezüglich der Positionsbestimmung. Auffällig ist die gehäufte Anzahl von Fehlern bei der Positionsbestimmung zwischen WLAN-RP 07 und WLAN-RP 12. Dies kann dadurch erklärt werden, dass sich all diese WLAN-RPs im ersten Stockwerk befinden und diese somit APs aus allen Stockwerken empfangen können. Während WLAN-RPs, die im Erdgeschoss oder im zweiten Stockwerk aufgezeichnet wurden, die APs des am weitest entfernten Stockwerks nur noch vereinzelt empfangen können. Abbildung 7.7 zeigt die Anzahl der Treffer bei der Stockwerkerkennung an den aufgenommenen 19 WLAN-RPs. Die gesamte Anzahl durchgeführter Stockwerkerkennungen wird durch „Total“, die Anzahl erfolgreicher Stockwerkerkennungen durch „Hit“ (grün) und die Anzahl falsch bestimmter Stockwerke „Miss“ (rot) dargestellt. Die Werte zeigen, dass die Zuverlässigkeit bei der Unterscheidung des Stockwerks mit dem WLAN-LFPT-Verfahren sehr hoch ist. Auch hier ist die erhöhte Fehlerrate im ersten Stockwerk zu sehen, welche deutlich höher ist als die in den anderen Stockwerken. Die Stockwerkerkennung allgemein hat dagegen sehr gut funktioniert.

Die Abbildungen 7.8 und 7.9 zeigen jeweils die Zuverlässigkeit der durchgeführten Positionsbestimmungen in Prozent bezogen auf die gesamte dreistöckige Installation und die Zuverlässigkeit auf Stockwerkbasis. Die Zuverlässigkeit der Positions-/Stockwerkerkennung ist ebenfalls dargestellt und beträgt $72,45\,\%$ für die Positions- und $96,10\,\%$ für die Stockwerkerkennung. Sichtbar ist jedoch ein Ausreißer an WLAN-RP 3. Da sich dieser jedoch genau vor einer Tür befindet, besteht die Möglichkeit, dass es bei der Aufnahme des WLAN-TPs oder in der RTP zu Interferenzen aufgrund von Mitarbeitern des Lehrstuhls gekommen ist, die den Raum betreten oder verlassen haben. Darüber hinaus lässt sich feststellen, dass die Zuverlässigkeit im ersten Stockwerk aufgrund der beschriebenen geringen Distanz zu den anderen Stockwerken im Vergleich zum Erdgeschoss oder dem zweiten Stockwerk vermindert ist.

Die Genauigkeit des Systems ist dargestellt durch die mittlere Abweichung. Hierzu wurde jeweils die Distanz zwischen der physikalischen Position des aktuellen WLAN-RPs und der vom System erkannten Position berechnet. Die Distanz ist Null, wenn die Position des aufgenommenen WLAN-RP richtig bestimmt werden konnte. Im Falle eines Fehlers bei der Positionsbestimmung des aktuellen WLAN-RPs wurde die Distanz zu dem WLAN-TP bestimmt, der vom System erkannt werden sollte.[3] Abbildung 7.10 zeigt die mittlere Abweichung bei der Positionsbestimmung der WLAN-RPs in der RTP in Metern. Auf der x-Achse aufgetragen sind nur WLAN-RPs die direkt an einem zufällig ausgewählten WLAN-TP aufgenommen wurden. Die mittlere Abweichung für diese 11 WLAN-RPs beträgt $2,55\,m$. Die hohe Ungenauigkeit bei den WLAN-RP 2 und WLAN-RP 17 deutet auf Interferenzen im Bereich dieses Teils des Sands zum Zeitpunkt der Messung hin. Die beiden WLAN-RPs befinden sich fast genau übereinander und wurden in einem Zeitfenster von weniger als zehn Minuten aufgenommen. Abbildung 7.11 zeigt dagegen die mittlere Abweichung bei der Positionsbestimmung der WLAN-RPs in der RTP in Metern für WLAN-RPs, deren Position beliebig gewählt wurde, also zwischen oder abseits der WLAN-TPs aufgenommen wurden. Die mittlere Abweichung bei der Messung dieser fünf WLAN-RPs beträgt nur $1,65\,m$. Dies ist ein unerwartet gutes Ergebnis. Der einzige Ausreißer bei den Messdaten findet sich bei WLAN-RP 18.

Eine weiteres Kriterium ist die Anzahl der empfangenen WLAN APs pro WLAN-TP. Dies ist in Abbildung 7.12 dargestellt, welche die Anzahl der empfangenen APs aller gemessenen 19 WLAN-TPs für alle Stockwerke in der TRP aufzeigt. Auf der x-Achse aufgetragen sind die WLAN-TPs. Auffällig ist hierbei die mit bis zu 55 AP pro WLAN-TP sehr hohe Dichte an APs an WLAN-TP 00. Weiterhin ist die Dichte der APs an den WLAN-TPs 25–28 und 42–44 mit teilweise nur 20 APs eher gering. Dies lässt sich aber dadurch erklären, dass sich diese WLAN-TPs allesamt auf der Außenseite des hinteren Sand 13 im Erdgeschoss und dem ersten Stockwerk befinden und an diesem Ende keine weiteren Gebäude mehr anschließen. In Stockwerk zwei dagegen tritt dies nicht so stark auf, weil sich der Gang hier auf der Innenseite des Gebäudes befindet und so auch noch APs von der gegenüberliegenden Seite empfangen werden können.

[3] Die Berechnung des Abstands wurde stockwerkübergreifend durchgeführt. Das heißt, es wurde immer die Distanz zum vom System erwarteten WLAN-TP berechnet, auch wenn dieser in einem anderen Stockwerk verortet war.

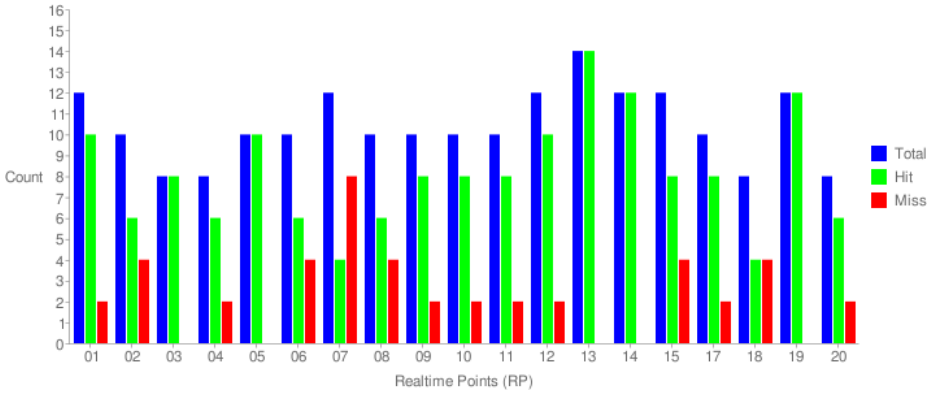


Abbildung 7.6: Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 19 WLAN-RPs für alle Stockwerke. Auf der x-Achse aufgetragen sind die WLAN-RPs. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen Miss (rot) dargestellt.

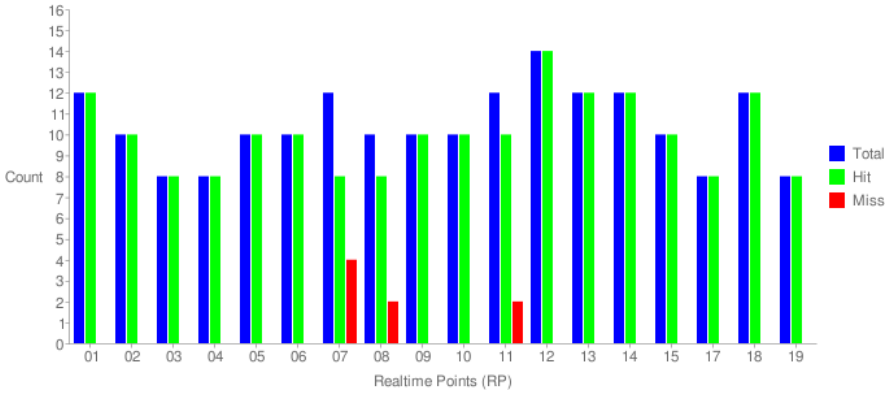


Abbildung 7.7: Anzahl der Treffer bei der Stockwerkerkennung an den aufgenommenen 19 WLAN-RPs. Auf der x-Achse aufgetragen sind die WLAN-RPs. Die gesamte Anzahl durchgeführter Stockwerkerkennung wird durch Total (blau), die Anzahl erfolgreicher Stockwerkerkennung durch Hit (grün) und die Anzahl falsch bestimmter Stockwerke Miss (rot) dargestellt.

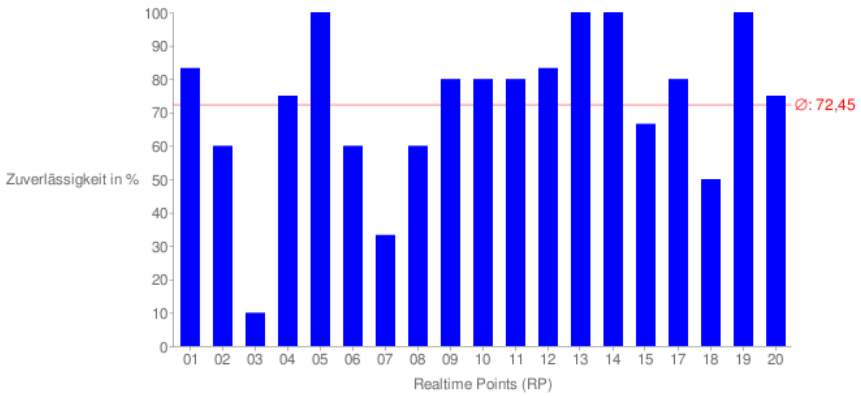


Abbildung 7.8: Zuverlässigkeit der Positionsbestimmung in Prozent für alle 19 WLAN-RPs auf allen Stockwerken. Auf der x-Achse aufgetragen sind die WLAN-RPs. Der Durchschnitt der Zuverlässigkeit der Positionsbestimmung aller WLAN-RPs ist als horizontale Linie eingetragen.

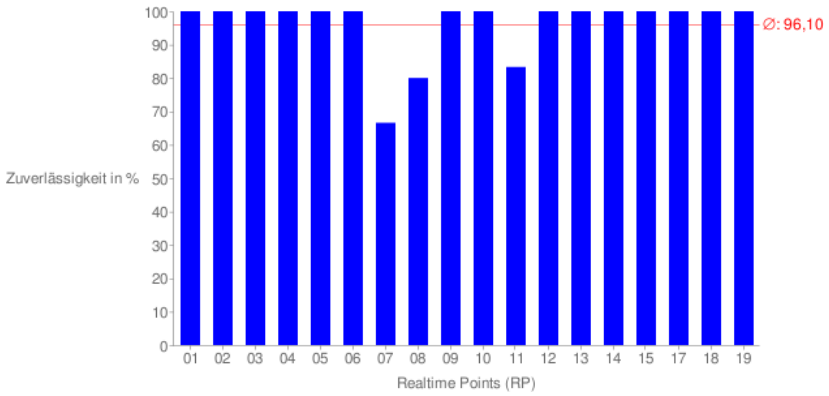


Abbildung 7.9: Zuverlässigkeit der Stockwerkerkennung in Prozent für alle 19 WLAN-RPs auf allen Stockwerken. Auf der x-Achse aufgetragen sind die WLAN-RPs. Der Durchschnitt der Zuverlässigkeit der Stockwerkerkennung aller WLAN-RPs ist als horizontale Linie eingetragen.

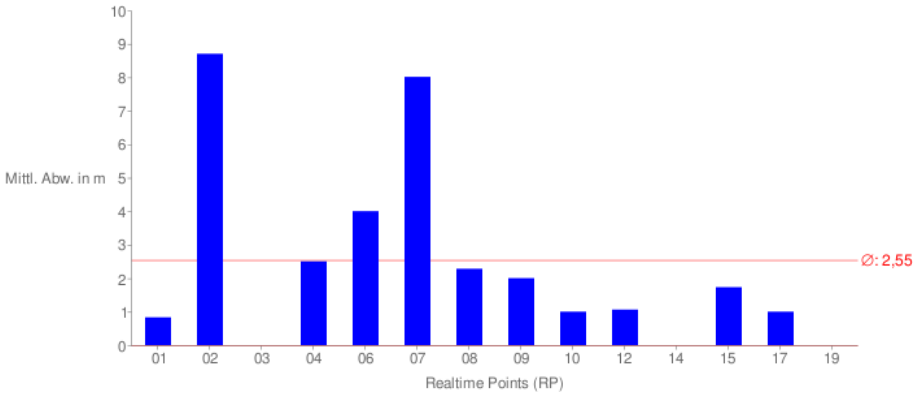


Abbildung 7.10: Mittlere Abweichung bei der Positionsbestimmung der WLAN-RPs in der RTP in Metern. Auf der x-Achse aufgetragen sind nur WLAN-RPs die direkt an einem WLAN-TP aufgenommen wurden. Die mittlere Abweichung ist als horizontale Linie eingetragen.

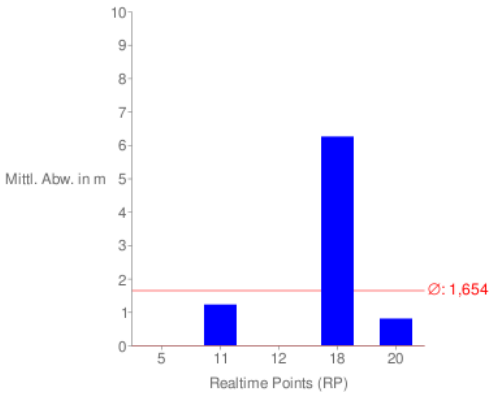


Abbildung 7.11: Mittlere Abweichung bei der Positionsbestimmung der WLAN-RPs in der RTP in Metern. Auf der x-Achse aufgetragen sind nur WLAN-RPs, deren Position beliebig gewählt wurde, dass heißt zwischen den WLAN-TPs aufgenommen wurden. Die mittlere Abweichung ist als horizontale Linie eingetragen.

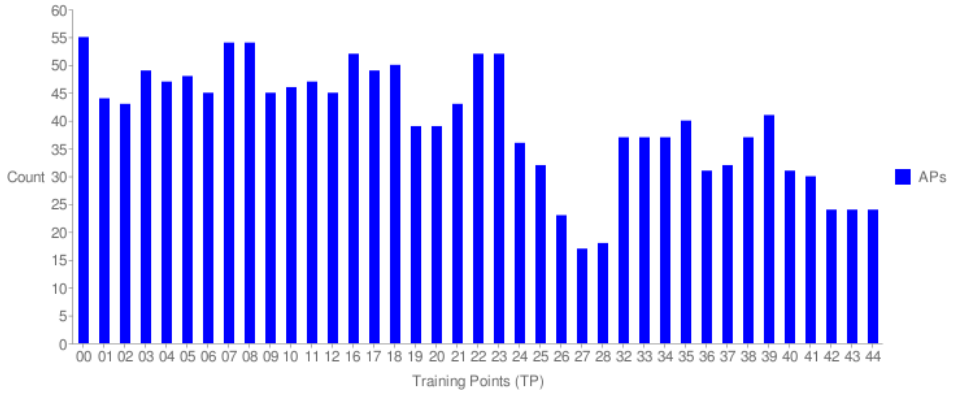


Abbildung 7.12: Anzahl der empfangenen APs (blau) aller gemessenen 19 WLAN-TPs für alle Stockwerke in der TRP. Auf der x-Achse aufgetragen sind die WLAN-TPs.

7.3.2 Messreihe 2 – GSM, 39 GSM-TPs, 5x5 m, 3 Stockwerke

Die zweite Messreihe entspricht bis auf wenige Änderungen der Systemparameter der ersten Messreihe. So wurde anstatt des WLAN-LFPT das GSM-LFPT-Verfahren verwendet und die Anzahl der durchgeführten Messungen pro GSM-TP sowie die Anzahl der gemessenen GSM-RPs variiert. Die Messungen wurden ebenfalls in allen drei Stockwerken des WSIs durchgeführt, um die Performanz des SSF bezüglich des GSM-LFPT Verfahrens in mehrstöckigen Installationen zu testen. Für die Messreihe wurden die folgenden Systemparameter gewählt:

7.3.2.1 Aufbau

7.3.2.1.1 Installation Die Größe der Installation betrug $133\,m^2$, als Messdichte wurde ein Raster von $5x5\,m$ gewählt. Aufgezeichnet wurde die GSM-RSS der aktuellen CID, sowie die RSS aller verfügbarer Nachbar CIDs in dBm.

7.3.2.1.2 TRP Die Anzahl der GSM-TP in der TRP betrug 39. An jedem GSM-TP wurden 4 LFPTS aufgezeichnet. Für die Anzahl der Messungen ergibt sich daher 100 ($M = \#LFPT * \#samples * bS$, mit $\#LFPT = 4, samples = 4, bS = 5$). Das Messintervall betrug $30\frac{s}{LFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0-90°)$ gemessen.

7.3.2.1.3 RTP Die Anzahl der GSM-RP in der RTP belief sich auf 16. Dabei befanden sich alle GSM-RPs direkt auf einem GSM-TP. Das Messintervall betrug $60\frac{s}{LFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0-90°)$ gemessen.

7.3.2.2 Ergebnisse

Die folgenden Abbildungen zeigen die Ergebnisse der zweiten Messreihe. Als Gebäudeplanreferenz zur Darstellung der GSM-RPs wurde derselbe Plan wie bei Messreihe 1 verwendet, mit den GSM-RPs in blau. Im Unterschied zur Messreihe 1 wurden in der RTP die GSM-RPs nicht beliebig gewählt, sondern jeweils in einem Abstand von $15\,m$ gemessen.

Abbildung 7.13 zeigt die Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 16 GSM-RPs für alle Stockwerke. Auf der x-Achse aufgetragen sind die GSM-RPs.

Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen Miss (rot) dargestellt. Die Anzahl der Treffer ist an fast allen GSM-RPs sehr gering und GSM-RPs werden nur vereinzelt erkannt. Abbildung 7.14 zeigt die Anzahl der Treffer bei der Stockwerkerkennung an den aufgenommenen 16 GSM-RPs. Auf der x-Achse aufgetragen sind die 16 GSM-RPs. Die gesamte Anzahl durchgeführter Stockwerkerkennung wird analog zur Anzahl der Treffer durch Total (blau), Hit (grün) und Miss (rot) dargestellt. Die Stockwerkerkennung funktioniert wesentlich besser als die Positionsbestimmung. Liegen die gemessenen GSM-RPs im Erdgeschoss oder im zweiten Stock, ist die Anzahl der Treffer wesentlich höher als im ersten Stockwerk. Dies kann damit begründet werden, dass die GSM-RPs im Erdgeschoss und dem zweiten Stock eine wesentlich größere Distanz zueinander aufweisen als die GSM-RPs vom Erdgeschoss beziehungsweise des zweiten Stockwerks im Vergleich zum ersten Stockwerk.

Abbildung 7.15 zeigt die Zuverlässigkeit der Positionsbestimmung in Prozent für alle 16 GSM-RPs auf allen Stockwerken. Die Zuverlässigkeit ist mit $15,95\,\%$ sehr niedrig und deutet auf eine zu geringe Messdichte hin. Abbildung 7.16 stellt die Zuverlässigkeit der Stockwerkerkennung in Prozent für alle 16 GSM-RPs dar. Die Zuverlässigkeit beträgt durchschnittlich $49,48\,\%$. Die Abbildung zeigt aber auch, dass diese noch wesentlich höher wäre, wenn man die GSM-RPs aus dem ersten Stock vernachlässigen und sich nur die Werte aus dem Erdgeschoss und zweiten Stocks zur Auswertung heranziehen würde.

Interessant ist hingegen die in Abbildung 7.17 dargestellte mittlere Abweichung bei der Positionsbestimmung der 16 GSM-RPs in der RTP in Metern. Diese beträgt durchschnittlich $13\,m$ und ist besser als nach den Auswertungen der Zuverlässigkeit zu erwarten. Die Berechnung des Abstands wurde wie in Messreihe 1 7.3.1 stockwerkübergreifend durchgeführt. Bis auf GSM-RP 10 und GSM-RP 12 ist Genauigkeit gleichbleibend.

Die Anzahl der empfangenen APs aller aufgenommenen 16 GSM-TPs für alle Stockwerke in der TRP ist in Abbildung 7.18 dargestellt. Auf der x-Achse aufgetragen sind die GSM-TPs. Auffällig ist die Verteilung der Anzahl empfangener APs in Bezug auf die Lage der GSM-TPs. GSM-TPs, die im oberen oder Mittelteil der Installation aufgenommen wurden empfangen in der Regel 3–4 Zellen pro GSM-TP, während die GSM-TPs 8, 9, 10, 11, 12, 17, 24, 25, 26, 27, 28, 41, 42 und 43 eine stark erhöhte Anzahl der empfangenen Zellen aufweisen. All diese GSM-TPs sind übereinander angeordnet und befinden sich durchweg im unteren Teil der Installation. An allen 39 GSM-TPs wurden insgesamt 19 verschiedene Zellen empfangen. Abbildung 7.19 zeigt, wie oft eine Zelle, zu der das ME aktiv verbunden war, an allen 39 GSM-TPs empfangen wurde. Abbildung 7.20 zeigt dagegen, wie oft die Nachbarzellen an allen 39 GSM-TPs empfangen wurden. Auf der x-Achse sind jeweils die CIDs der aufgenommenen Zellen eingetragen. Bei den aktiven Zellen wurden an den meisten Trainingspunkten 2 CIDs empfangen, während bei den Nachbarzellen 4 CIDs am häufigsten empfangen wurden. Bei den anderen empfangenen Zellen scheint es sich um weiter entfernte Zellen zu handeln, die nur sporadisch empfangen werden konnten.

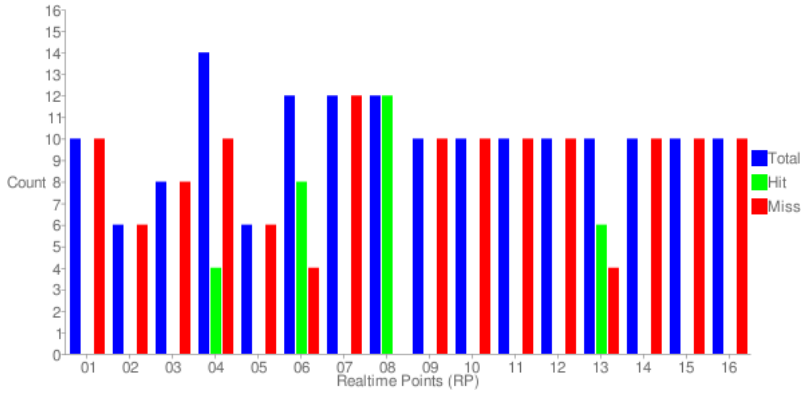


Abbildung 7.13: Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 16 GSM-RPs für alle Stockwerke. Auf der x-Achse aufgetragen sind die GSM-RPs. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen durch Miss (rot) dargestellt.

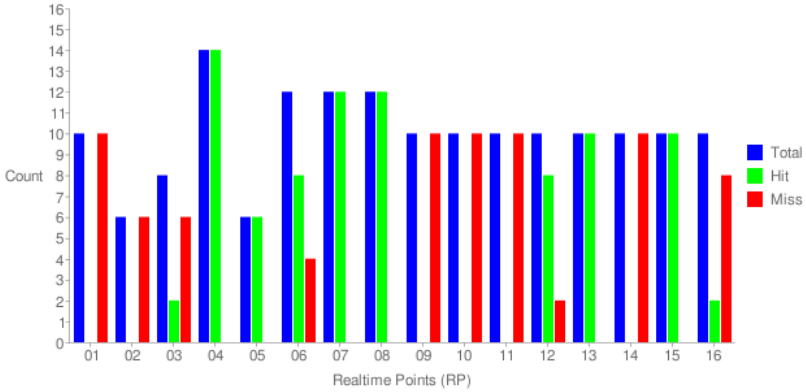


Abbildung 7.14: Anzahl der Treffer bei der Stockwerkerkennung an den aufgenommenen 16 GSM-RPs. Auf der x-Achse aufgetragen sind die GSM-RPs. Die gesamte Anzahl durchgeführter Stockwerk-Erkennung wird durch Total (blau), die Anzahl erfolgreicher Stockwerk-Erkennung durch Hit (grün) und die Anzahl falsch bestimmter Stockwerke durch Miss (rot) dargestellt.

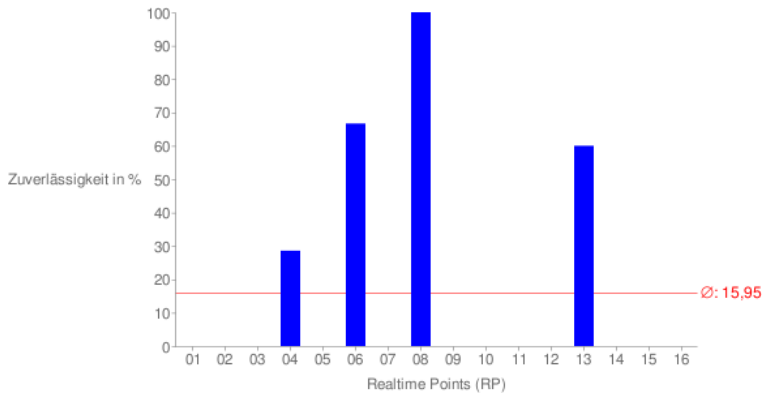


Abbildung 7.15: Zuverlässigkeit der Stockwerkerkennung in Prozent für alle 16 GSM-RPs auf allen Stockwerken. Auf der x-Achse aufgetragen sind die GSM-RPs. Der Durchschnitt der Zuverlässigkeit der Positionsbestimmung aller GSM-RPs ist als horizontale Linie eingetragen.

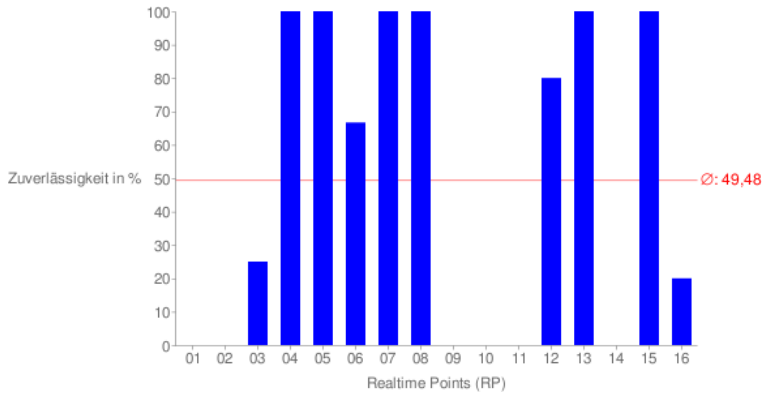


Abbildung 7.16: Zuverlässigkeit der Stockwerkerkennung in Prozent für alle 16 GSM-RPs auf allen Stockwerken. Auf der x-Achse aufgetragen sind die GSM-RPs. Der Durchschnitt der Zuverlässigkeit der Stockwerk-Erkennung aller GSM-RPs ist als horizontale Linie eingetragen.

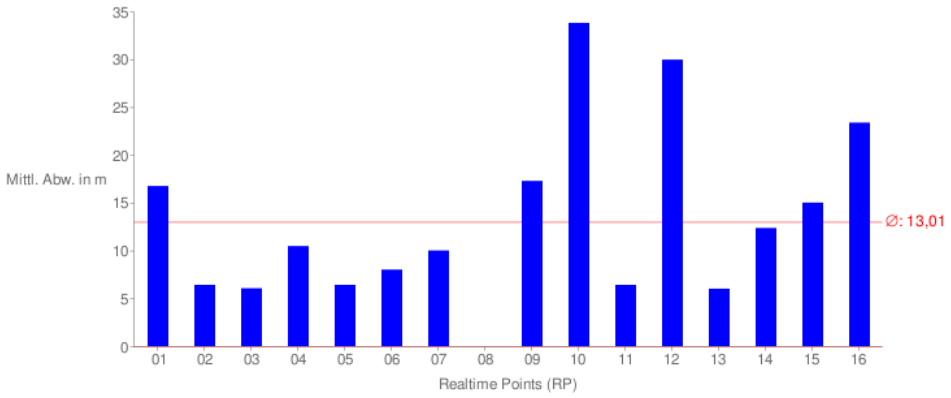


Abbildung 7.17: Mittlere Abweichung bei der Positionsbestimmung der 16 GSM-RPs in der RTP in Metern. Auf der x-Achse aufgetragen sind nur GSM-RPs, die direkt an einem GSM-TP aufgenommen wurden. Die mittlere Abweichung ist als horizontale Linie eingetragen.

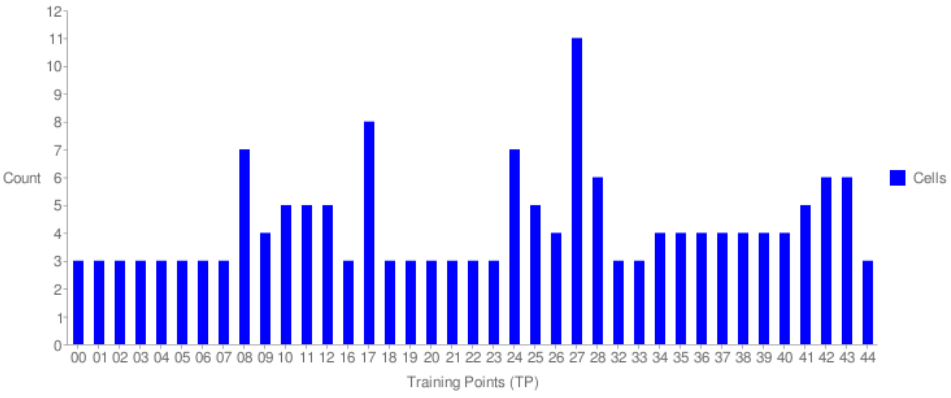


Abbildung 7.18: Anzahl der empfangenen APs (blau) aller gemessenen 16 GSM-TPs für alle Stockwerke in der TRP. Auf der x-Achse aufgetragen sind die GSM-TPs.

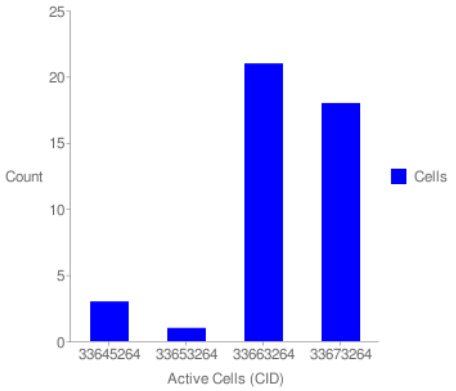


Abbildung 7.19: Anzahl, wie oft eine aktive GSM-Zelle, zu der das ME verbunden war, an 39 GSM-TPs auf allen Stockwerken empfangen wurde. Auf der x-Achse sind die IDs (CID) der einzelnen Zellen aufgetragen.

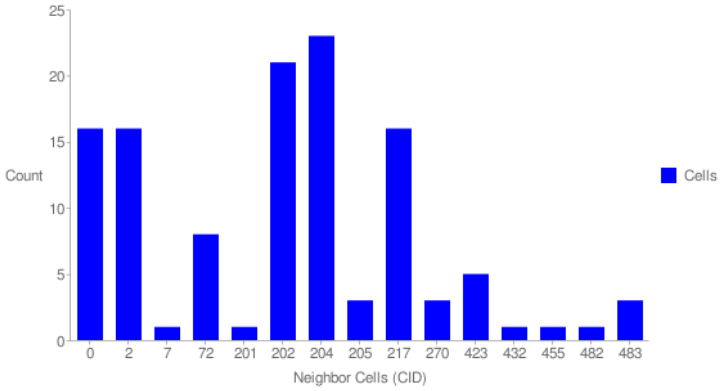


Abbildung 7.20: Anzahl, wie oft eine GSM-Nachbarzelle an 39 GSM-TPs auf allen Stockwerken empfangen wurde. Auf der x-Achse sind die IDs (CID) der einzelnen Zellen aufgetragen.

7.3.3 Messreihe 3 – GSM, 5 GSM-TPs, 50x50 m, 3. Stockwerk

Aufgrund der Ergebnisse aus Messreihe 2, mit einer mittleren Abweichung der GSM Positionsbestimmung von $13,01\,m$ wurde ein weiterer Versuch durchgeführt, um die Genauigkeit eines GSM-ILS, welches auch in der Praxis ohne Probleme eingesetzt werden kann, zu bestimmen. So wurden die Messungen nicht wie bei den Messreihen 1 und 2 im hinteren Teil des Sand 13 durchgeführt, sondern verteilt über das gesamte WSI. Der Gebäudeplan der Messung ist in Abbildung 7.21 dargestellt. Die Rastergröße wurde für diesen Versuch mit großen Abständen zwischen $40\,m - 80\,m$ gewählt. Auf dem Gebäudeplan sind – wie bei den ersten beiden Messreihen – die GSM-TPs eingetragen. GSM-RPs sind im Gebäudeplan nicht separat dargestellt, weil bei der Durchführung der Messung in der RTP die Positionsbestimmung an allen GSM-TPs durchgeführt wurde. Die Systemparameter wurden wie folgt festgelegt:

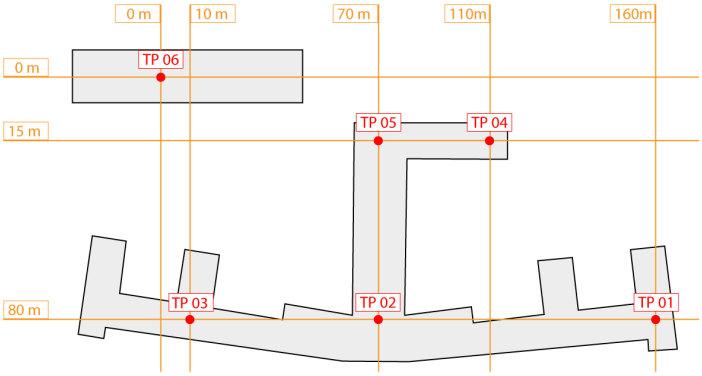


Abbildung 7.21: Gebäudeplan des gesamten WSI, eingezeichnet sind die GSM-TPs und das gewählte Raster durch die orangenen Linien.

7.3.3.1 Aufbau

7.3.3.1.1 Installation Die Größe der Installation betrug $12.800\,m^2$, als Messdichte wurde ein Raster von ca. $40\,m - 80\,m$ gewählt. Aufgezeichnet wurde die GSM-RSS der aktuellen CID, sowie die RSS aller verfügbarer Nachbar CIDs in dBm.

7.3.3.1.2 TRP Die Anzahl der GSM-TPs in der TRP betrug 6, an jedem GSM-TP wurden 4 LFPTs aufgezeichnet. Das ergibt für die Anzahl der Messungen 100 ($M = \#LFPT * \#samples * bS$, mit $\#LFPT = 4, samples = 4, bS = 5$). Das Messintervall betrug $30\frac{s}{LFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0 - 90°)$ aufgezeichnet.

7.3.3.1.3 RTP Die Anzahl der GSM-RPs in der RTP betrug 6 und alle lagen direkt auf einem GSM-TP. Das Messintervall betrug $60\frac{s}{RTFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0 - 90°)$ aufgezeichnet.

7.3.3.2 Ergebnisse

Abbildung 7.22 zeigt die Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 6 GSM-RPs. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen durch Miss (rot) dargestellt. Die GSM-RPs konnten an allen GSM-TPs korrekt erkannt werden. Abbildung 7.23 zeigt die Zuverlässigkeit der Positionsbestimmung der Messreihe in Prozent. Da alle Positionen eindeutig zugeordnet werden, beträgt die Zuverlässigkeit 100 %

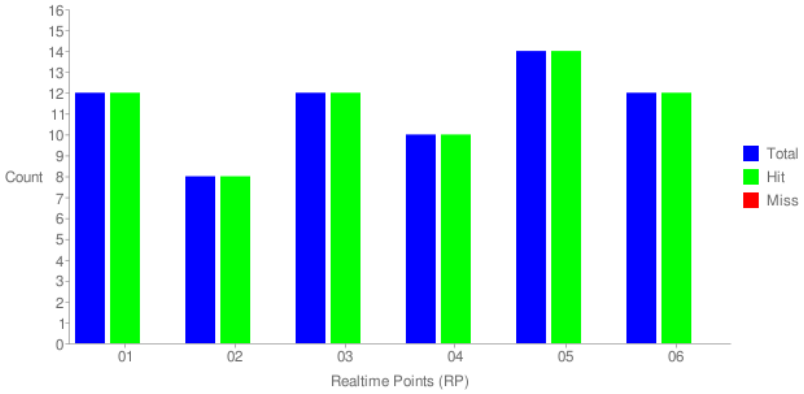


Abbildung 7.22: Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 6 GSM-RPs. Auf der x-Achse aufgetragen sind die GSM-RPs. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen durch Miss (rot) dargestellt.

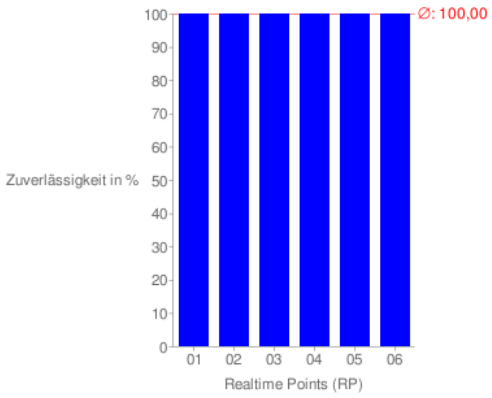


Abbildung 7.23: Zuverlässigkeit der Positionsbestimmung in Prozent für alle 16 GSM-RPs auf allen Stockwerken. Auf der x-Achse aufgetragen sind die GSM-RPs. Der Durchschnitt der Zuverlässigkeit der Positionsbestimmung aller GSM-RPs ist als horizontale Linie eingetragen.

7.3.4 Messreihe 4 – WLAN, 15 WLAN-TPs, 5x5 m, 3. Stockwerk

Die vierte Messreihe hatte zum Ziel, ein möglichst realitätsnahes Alltagsszenario nachzustellen. Vor dem Hintergrund, dass das SSF vor allem von einer breiten Masse an Nutzern in ihren eigenen Räumlichkeiten eingesetzt werden soll, sollte die Performanz in einem realitätsnahen Szenario untersucht werden. In der Regel wird ein Nutzer des SSF dieses dafür nutzen, unterschiedliche Räume oder Orte innerhalb bestimmter Räume zu unterscheiden. Zur Simulation eines solchen Szenarios wurden die Messungen im dritten Stockwerk des hinteren Sands durchgeführt. Der Grund ist, dass auf diesem Stockwerk neben den Gängen auch Messungen in Räumen wie Küche, Besprechungsraum und Büros durchgeführt werden konnten. Dies war im Erdgeschoss und im ersten Stockwerk leider nicht möglich. Die Installation kommt aber einem realen Einsatz in einem Wohnhaus oder einer Eigentumswohnung schon sehr nahe. Die Rastergröße wurde mit $5x5\,m$ entsprechend gewählt, um dieses Szenario so gut wie möglich nach-

zustellen. Der Gebäudeplan sowie die WLAN-TPs und WLAN-RPs sind in Abbildung 7.24 dargestellt. WLAN-TPs und WLAN-RPs sind nicht getrennt voneinander aufgeführt, weil in dieser Messreihe in der RTP alle WLAN-TPs als WLAN-RPs aufgenommen wurden. Die Systemparameter wurden wie folgt festgelegt:

Abbildung 7.24: Gebäudeplan für die WLAN Messreihe im zweiten Stockwerk des hinteren Sand 13. Eingezeichnet sind die WLAN-TPs (rot). Die WLAN-RPs sind nicht separat eingezeichnet. Die Messungen wurden an allen WLAN-TPs ausgeführt. Das in der Messreihe verwendete Raster ist durch die orangenen Linien gekennzeichnet.

7.3.4.1 Aufbau

7.3.4.1.1 Installation Die Größe der Installation betrug $133\,m^2$, als Messdichte wurde ein Raster von $5x5\,m$ gewählt. Aufgezeichnet wurde die WLAN-RSS aller verfügbarer APs in dBm.

7.3.4.1.2 TRP Die Anzahl der WLAN-TPs in der TRP betrug 13, an jedem WLAN-TP wurden 5 LFPTs aufgezeichnet. Das ergibt für die Anzahl der Messungen 125 ($M = \#LFPT * \#samples * bS$, mit $\#LFPT = 5, samples = 5, bS = 5$). Das Messintervall betrug $30\frac{s}{LFPT}$ und es wurde bei einer Orientierung von $ori0$ $(0-90°)$ aufgezeichnet.

7.3.4.1.3 RTP Die Anzahl der WLAN-RPs in der RTP betrug 13, davon lagen alle direkt auf einem WLAN-TP. Das Messintervall betrug $60\frac{s}{RTFPT}$ und es wurde bei einer Orientierung von

*ori*0 ($0 - 90°$) aufgezeichnet.

7.3.4.2 Ergebnisse

Die folgenden Abbildungen zeigen die Ergebnisse der vierten Messreihe. Abbildung 7.25 zeigt die Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 13 WLAN-RPs auf dem dritten Stockwerk. Auf der x-Achse aufgetragen sind die WLAN-RPs. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen Miss (rot) dargestellt. Die Werte zeigen eine gute Trefferquote an allen WLAN-RPs. Zu einer leicht verringerten Trefferquote kommt es lediglich an den WLAN-RPs 00, 01, 02 und 03. Dies ist aber darauf zurückzuführen, dass bei der Aufnahme der WLAN-TPs das ME aufgrund eines leeren Akkus an einen Laptop angeschlossen werden musste. Die ersten vier Punkte in der RTP (WLAN-RP 00 bis 03) wurden jedoch ohne angeschlossenes Laptop aufgenommen und erst wieder ab WLAN-RP 04 aufgrund eines erneut schwachen Akkus wieder mit angeschlossenem Laptop.

Abbildung 7.26 stellt die Zuverlässigkeit der Positionsbestimmung in Prozent für alle 13 WLAN-RPs auf dem dritten Stockwerk dar. Auf der x-Achse aufgetragen sind die WLAN-RPs. Der Durchschnitt der Zuverlässigkeit bei der Positionsbestimmung ist mit 81 % sehr gut. Zu sehen ist die verminderte Zuverlässigkeit im Bereich der WLAN-RPs 00–03. Der mögliche Grund könnte wie bereits besprochen der fehlende Anschluss an einen Laptop darstellen.

Die mittlere Abweichung der Positionsbestimmung ist in Abbildung 7.27 dargestellt. Auf der x-Achse aufgetragen sind nur WLAN-RPs, die direkt an einem WLAN-TP aufgenommen wurden. Zur Berechnung der Distanz zwischen der physikalischen Position des aktuellen WLAN-RPs wurde jeweils der Abstand zwischen dem vom System erkannten WLAN-RPs und dem vom System erwarteten WLAN-TP genommen. Die daraus resultierende mittlere Abweichung beträgt $1,41\,m$. Aufgrund der beschriebenen Problematik mit einem leeren Akku sind die Werte der WLAN-RPs 00–03 entsprechend erhöht.

Auch bei diesem Versuch wurde die Anzahl der empfangenen APs aller gemessenen 13 WLAN-TPs, in der TRP erfasst. Auf der x-Achse aufgetragen sind die WLAN-TPs. Abbildung 7.28 zeigt die Ergebnisse. Die Anzahl der empfangenen APs entspricht in etwa der, der in Messreihe 1 7.3.1 empfangenen Anzahl an APs. Die Anzahl der APs nimmt ab, je weiter man sich dem unteren Ende des hinteren Sands 13 nähert. Die WLAN-RPs 10 und 12 weisen mit 23 und 30 APs die geringste Dichte auf. Diese WLAN-RPs befinden sich in Räumlichkeiten des Lehrstuhls. Hier können deutlich weniger APs empfangen werden wie im Gang oder in den Räumen am oberen Ende des hinteren Sand 13. Lässt man die WLAN-RPs 0–3 aus besagten Gründen außen vor, ist die verminderte Anzahl an APs auch bei WLAN-RP 05 vorhanden. Der Grund hierfür ist, dass die APs auf dem Sandgebäude in der Regel in den Gängen und nicht in den Räumen angebracht sind. Hierfür spricht, dass diese Ausprägung in Messreihe 1 7.3.1 nicht aufgetreten ist.

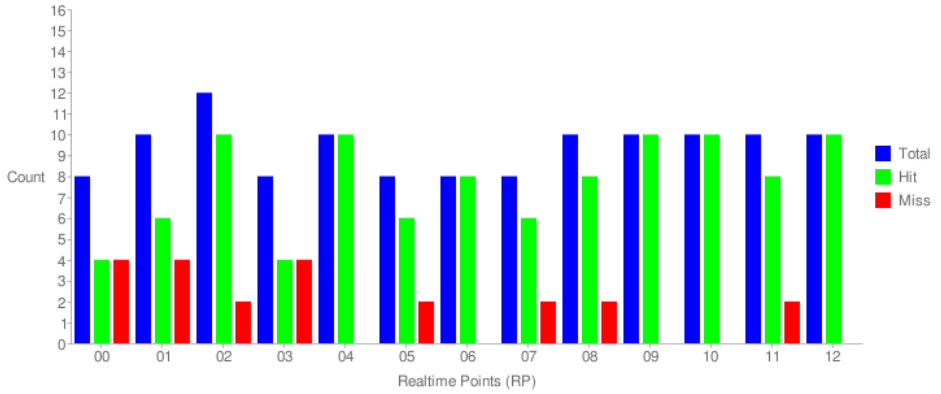


Abbildung 7.25: Anzahl der Treffer bei der Positionsbestimmungen an den aufgenommenen 13 WLAN-RPs auf dem dritten Stockwerk. Auf der x-Achse aufgetragen sind die WLAN-RPs. Die gesamte Anzahl durchgeführter Positionsbestimmungen wird durch Total (blau), die Anzahl erfolgreicher Positionsbestimmungen durch Hit (grün) und die Anzahl falsch bestimmter Positionen Miss (rot) dargestellt.

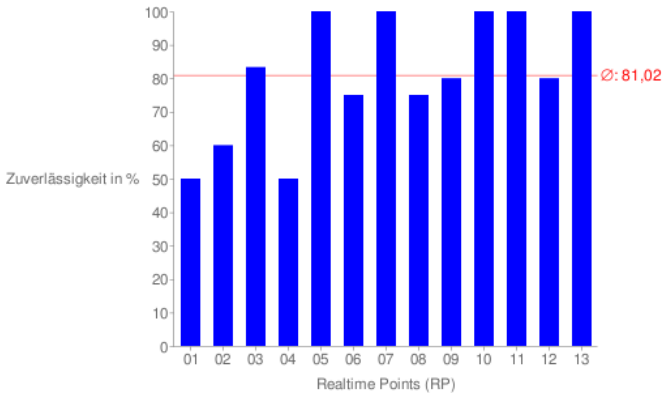


Abbildung 7.26: Zuverlässigkeit der Positionsbestimmung in Prozent für alle 13 WLAN-RPs auf dem dritten Stockwerk. Auf der x-Achse aufgetragen sind die WLAN-RPs. Der Durchschnitt der Zuverlässigkeit der Positionsbestimmung aller WLAN-RPs ist als horizontale Linie eingetragen.

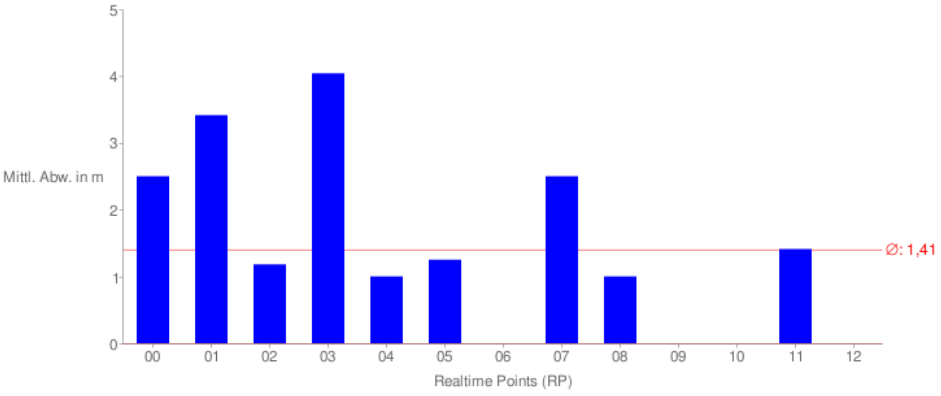


Abbildung 7.27: Mittlere Abweichung bei der Positionsbestimmung der WLAN-RPs in der RTP in Metern. Auf der x-Achse aufgetragen sind nur WLAN-RPs, die direkt an einem WLAN-TP aufgenommen wurden. Die mittlere Abweichung ist als horizontale Linie eingetragen.

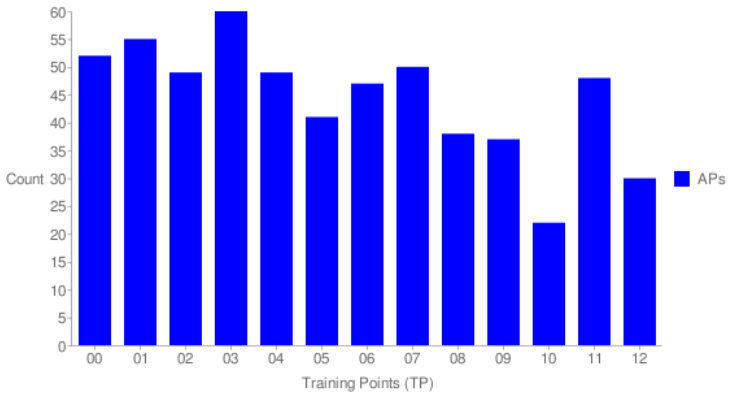


Abbildung 7.28: Anzahl der empfangenen APs aller gemessenen 13 WLAN-TPs in der TRP im dritten Stockwerk. Auf der x-Achse aufgetragen sind die WLAN-TPs.

7.3.5 Messgeschwindigkeit WLAN

Zusätzlich zu den Performanz-Metriken der Genauigkeit und Zuverlässigkeit wurde die Lokalisierungsgeschwindigkeit des Systems in der RTP bestimmt. Abbildung 7.29 zeigt die Ergebnisse der Messungen. Die Werte zeigen deutlich, dass die Zeit zur Lokalisierung des ME exponentiell mit der Anzahl der WLAN-TPs ansteigt. Die durchschnittliche Anzahl an APs hat ebenfalls Auswirkungen auf die Lokalisierungsgeschwindigkeit. Je höher die Anzahl an APs pro WLAN-TP, desto länger dauerte die Lokalisierung. Tabelle 7.3.5 zeigt zusätzlich noch die benötigte Zeit in Sekunden, die pro AP jeweils benötigt wurde, um die Berechnung für genau einen WLAN-TP durchzuführen. Diese Werte sind, wie erwartet, gleichbleibend, wenn die gleiche Anzahl an APs pro WLAN-TP verwendet wird, steigt aber deutlich an, wenn die Anzahl der APs pro WLAN-TP erhöht ist.

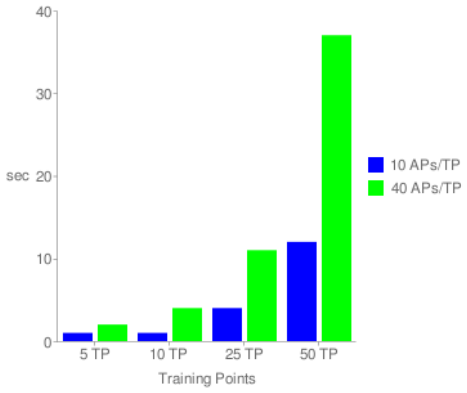


Abbildung 7.29: Absolute Lokalisierungszeit in Sekunden in der RTP anhand der Anzahl der RM-Größe und unter Verwendung einer unterschiedlichen Anzahl an APs. Auf der x-Achse aufgetragen ist die Anzahl der verwendeten WLAN-TPs. Die Anzahl der zur Lokalisierung verwendeten APs ist bei 10 APs pro WLAN-TP in blau, bei 40 APs pro WLAN-TP grün dargestellt.

#AP	#TP	sec/TP
10	5	0.15
10	10	0.2
10	25	0.16
10	50	0.24
40	5	0.4
40	10	0.4
40	25	0.44
40	50	0.74

Tabelle 7.2: Durchschnittliche Lokalisierungsgeschwindigkeit in Sekunden, in der RTP anhand der RM-Größe und AP-Anzahl

8 Diskussion

In diesem letzten Kapitel soll das Vorgehen bei der Erstellung der Diplomarbeit noch einmal repetiert werden und ein Rückblick sowie eine Diskussion über die Erstellung der gesamten Arbeit erfolgen. Ebenso werden die Erkenntnisse, die sich im Verlauf der Arbeit ergeben haben, diskutiert und schließlich die erreichten Ziele zusammengefasst und erörtert. Abschließend wird anhand der gewonnenen Erkenntnisse ein Ausblick auf zukünftige Erweiterungen und Potentiale der Arbeit gegeben.

8.1 Vorgehen

Die Idee zu dieser Diplomarbeit, ein ILS zu entwerfen, das jeder einfach und unkompliziert auch in seinen eigenen Räumlichkeiten installieren kann, entstand lange vor Beginn der eigentlichen Arbeit. Erst die technologische und gesellschaftliche Entwicklung, vor allem der letzten drei Jahre, ermöglichte überhaupt eine Auseinandersetzung und Umsetzung, wie sie in dieser Arbeit vorzufinden ist, durchzuführen.

Am Anfang der Diplomarbeit standen zunächst allgemeine Überlegungen die Möglichkeiten der Indoor-Lokalisierung auf ME, im besonderen des Form-Faktors, Smartphone genauer zu untersuchen. Hierzu musste eine Analyse bereits bekannter und implementierter ILS und neuester Ansätze aktueller Forschungen in diesem Bereich durchgeführt werden. Daraus wurde dann ein Konzept abgeleitet, welches eine Indoor-Lokalisierungs Implementierung auf einem gängigen ME aus dem Massenmarkt ermöglichen sollte. Dabei ging es von Anfang an weniger darum, wohlbekannte Techniken und Verfahren zu verbessern, sondern im Vordergrund stand, mit bestehenden Verfahren ein System zu schaffen, welches von der breiten Masse auf einer gängigen Hardware verwendet werden kann. Danach sollte von der konkreten Implementierung eine allgemeingültige Abstraktion induziert werden und diese in Form eines Frameworks ausgeliefert werden. Dabei war es wesentlich, dass das Framework einfach von anderen Entwicklern benutzt werden kann, und um neue Komponenten wie Sensoren, Filter und Lokalisierungsalgorithmen erweiterbar sein soll. Um dieses Ziel zu erreichen, wurde die Arbeit in drei Phasen unterteilt:

1. Auswahl eines geeigneten MEs zur Umsetzung und Implementierung eines ILS und Einarbeitung in die Konzepte und Funktionsweise mobiler Betriebssysteme, im Speziellen in das in der Arbeit verwendete Android Betriebssystem und dessen Entwicklungsumgebung, bestehend aus SDK, NDK und das Eclipse-Plugin ADT.

2. Literaturrecherche, um einen umfassenden Überblick über die Technologien, Techniken und Verfahren von ILS zu bekommen. Im Anschluss daran folgte die Gegenüberstellung und Auswertung der in der Literaturrecherche gewonnenen Erkenntnisse. Schließlich die Identifizierung der am besten geeignetsten Kombination der untersuchten Technologien, Techniken und Verfahren und die Erstellung eines geeigneten Konzeptes zur Realisierung eines ILS auf einem Android basierten Endgerät.

3. Design und Implementierung eines konkreten ILS-Prototypen auf Basis von Google Android. Anschließend folgte die Abstrahierung und Umsetzung eines allgemeingültigen Frameworks aus der konkreten Implementierung, welches von anderen Entwicklern der Android Plattform in ihren Android Applikationen verwendet werden kann.

8.1.1 Auswahl des ME

In der ersten Phase der Diplomarbeit musste ein geeignetes ME zur Umsetzung des geplanten ILS gefunden werden. Die Kriterien bei der Auswahl des MEs wurden dabei folgendermaßen definiert:

- Open-Source Betriebssystem,
- Qualität des Software Development Kits (SDK) und der Entwicklungsumgebung,
- API-Unterstützung diverser Sensoren und Funktechnologien,
- Zukunftsaussichten und Position am Markt.

Anhand dieser Kriterien wurde eine Evaluation des Marktes für MEs durchgeführt. Die Betriebssysteme Windows Mobile, das von Nokia entwickelte Symbian[1] und PalmWebOS wurden aufgrund ihres nicht quelloffenen Betriebssystems von vornherein ausgeschlossen. Das iPhone wurde aufgrund seines modernen Betriebssystems und seiner Relevanz am Markt, trotz der proprietären Implementierung des Betriebssystems, als Plattform in Erwägung gezogen. Das iPhone-OS bietet ein hoch qualitatives SDK, welches Unterstützung für diverse Sensoren und Funktechnologien bietet. Des weiteren war es zu Beginn der Arbeit schon weit am Markt verbreitet und sehr innovativ. Der Grund für die Entscheidung gegen den Einsatz des iPhones als ME war die proprietäre, nicht quelloffene Software des iPhone-OS und die Restriktionen, die bei der Entwicklung mit dem iPhone-OS von Apple auferlegt werden. Zudem hätte für die Arbeit eigens ein iPhone und ein Macintosh Rechner angeschafft werden müssen, da nur auf Rechner mit MacOS Betriebsystem für das iPhone entwickelt werden kann. Die geräteübergreifende Alternative in Form von Java Microedition (JavaME) musste ebenfalls ausgeschlossenen werden, da sie auf den MEs aus Sicherheitsgründen nur in einer Sandbox ausgeführt werden kann und so der Zugriff auf die Sensoren und den verbauten Funktechnologien nicht in vollem Umfang möglich ist. Die weiteren am Markt verfügbaren Open-Source Betriebssysteme beschränkten sich auf Maemo, Openmoko und die Android Plattform. Maemo und Openmoko basieren auf Linux und erfüllen beide die Kriterien im Bezug auf die Quelloffenheit und ermöglichen ebenfalls einen vollen Zugriff auf alle auf dem ME verfügbaren Sensoren und Funktechnologien. Sie haben jedoch beide den Nachteil, dass sie nur auf wenigen ME, wie dem Neo oder dem Nokia 810, eingesetzt werden und die Zukunft, im Vergleich zu der damals noch jungen aber bereits sehr viel versprechenden Google Android Plattform, nicht klar war.

Der Grund warum letztendlich die Android Plattform ausgewählt wurde ist, dass sie die im Vorfeld festgelegten Kriterien am besten erfüllte. Das Android Betriebssystem steht, bis auf einige gerätespezifische Treiber, komplett als Open-Source Implementierung zur Verfügung. Diese Quelloffenheit ermöglichte tiefe Einblicke in die Funktionsweise des Betriebssystems und stellte weiterhin ein gut durchdachtes SDK zur Verfügung, dessen API den Zugriff auf

[1] Symbian war zum Zeitpunkt, als diese Diplomarbeit begonnen wurde, noch nicht Open-Source und wurde erst am 4.2.2010 vollständig quelloffen.

zahlreiche Sensoren unkompliziert ermöglichte. Zudem müssen alle Geräte, die das Android Betriebssystem verwenden, gewisse Standards bezüglich der in den Geräten verbauten Funktechnologien und Sensoren einhalten. So besaß das verwendete Testgerät HTC G1 neben einer GSM-, einer WLAN- und einer Bluetooth-Schnittstelle auch IMUs zur Beschleunigungs-, Lage- und Orientierungsmessung, sowie eine Kamera. Außerdem stand bereits damals fest, dass es zukünftig viele weitere Geräte[2] auf Basis des Android Betriebssystems geben würde und dass die Android Plattform großen Rückhalt der führenden Technologie- und Software-Firmen, darunter Google, HTC, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile und Nvidia haben würde. Somit war die Hardware und die Softwarebasis für die Implementierung des SSF gefunden und es konnte mit der Einarbeitung in die Konzepte, Architektur und den APIs begonnen werden, um vor der letztendlichen Umsetzung einen guten Überblick über die Möglichkeiten der Plattform zu erhalten.

8.1.2 Literaturrecherche, Technologie Auswahl und Konzeption

In der zweiten Phase wurde eine gründliche Literaturrecherche zum Thema Indoor-Lokalisierung durchgeführt. Dabei wurde recherchiert, welche Technologien, Techniken, Verfahren und Systeme im Bereich der Indoor-Lokalisierung bereits implementiert wurden. Anschließend wurden diese im Hinblick auf die weitere Realisierung einer adäquaten Indoor-Lokalisierung auf Android basierten MEs überprüft. Nach der durchgeführten Recherche wurden deren Ergebnisse gegenübergestellt und abschließend bewertet. Am Ende des Prozesses standen die Entscheidungen, WLAN und GSM und die IMUs als Technologien zu verwenden und als Lokalisierungsverfahren das tabellenbasierte Location-Fingerprinting Verfahren zu verwenden.

Die Entscheidung zugunsten der verwendeten Technologien wurde aufgrund der in der Literaturrecherche gewonnenen Erkenntnisse und der angestrebten Anwendungsszenarien getroffen. Hierbei spielte die ubiquitäre Verfügbarkeit der Infrastruktur der GSM- und WLAN-Netze eine große Rolle bei der Auswahl dieser Technologien. Zudem wurden diesen Technologien in der Literatur bereits für den Aufbau mehrerer ILSs anhand des Location-Fingerprint Verfahrens verwendet. Dagegen wurde die dritte auf den Android basierten MEs zur Verfügung stehende Funktechnologie Bluetooth zwar ebenfalls untersucht, aber vorerst nicht in die Auswahl der zu implementierenden Sensoren mit einbezogen. Bluetooth kann keine flächendeckende Abdeckung wie WLAN und GSM aufweisen und steht, wenn vorhanden, meist nicht in Form stationärer Landmarken, sondern in Form von mobilen Sendern wie Smartphones oder Notebooks zur Verfügung. Deshalb kann es nur bei Installation mit eigens errichteter Infrastruktur adäquat verwendet werden. In keinster Weise kann es aber für eine Lokalisierung, wie sie in dieser Arbeit durchgeführt werden sollte, herangezogen werden. So müsste zur Verwendung von Bluetooth innerhalb eines ILS erst ein Netz aus Bluetooth Landmarken installiert werden. Diese müssten permanent mit Strom versorgt werden und würden aufgrund der recht eingeschränkten Reichweite von ca. 10 Metern, bei einer durchschnittlichen Leistungsaufnahme von $2,5mw$, schon in relativ kleinen Installationen zur Einrichtung von mehreren Bluetooth Landmarken führen. Eine Errichtung einer solchen Infrastruktur durch einen durchschnittlichen Nutzer eines Smartphones ist indes nicht zu erwarten, alleine aufgrund der entstehenden Kosten. Auch die Verwendung von Bluetooth Landmarken in der Umgebung ist wegen der geringen Reichweite von Bluetooth eher unwahrscheinlich. Es wird aber dennoch nicht ausgeschlossenen, Bluetooth nach Fertigstellung dieser Arbeit in das Smartspace Framework zu integrieren,

[2] Es war im Rahmen der Diplomarbeit möglich, auch einige Versuche mit dem Google Nexus One durchzuführen, welches erst nach der Implementierung des SSF erschienen ist.

um Entwicklern die Möglichkeit zu bieten, auch dieses Verfahren in ihren Applikationen auf Basis des Smartspace Frameworks anbieten zu können. Es soll an dieser Stelle aber trotzdem betont werden, dass Bluetooth großes Potential zur Umsetzung von ILS bietet. Die Potentiale liegen aber eher in der Verwendung, in Verbindung mit einer Installation, mit aktiv errichteter Infrastruktur und können dort bedeutende Genauigkeiten im Zentimeterbereich liefern.

Das Location Fingerprint Verfahren wurde gewählt, weil es den Anforderungen am besten entsprochen hat. Ausschlaggebend für die Entscheidung war die Tatsache, dass zur Indoor-Lokalisierung mittels dieses Verfahrens keinerlei aktive Infrastruktur benötigt wird und auch keine aktive Verbindung zu den Infrastrukturkomponenten wie WLAN-APs oder GSM-BTS aufgebaut werden muss. So kann das Location Fingerprint Verfahren rein passiv eingesetzt werden, indem die Feldstärken aller verfügbarer Funktechnologien als Fingerabdruck, einer bestimmten Position im Raum, in einer Datenbanktabelle abgespeichert werden und bei der Lokalisierung durch Abgleich mit der Datenbank die aktuelle Position des MEs bestimmt werden kann. Dies führt dazu, dass eine Lokalisierung mittels dieses Verfahren in einer sehr leicht gewichtigen Lösung implementiert werden kann, die keinerlei externe Infrastrukturkomponenten benötigt. Zudem ermöglicht es die Lokalisierung direkt auf dem ME durchzuführen und passt sehr gut zu dem gewählten Anwendungsszenario. So können Nutzer des Systems zum Beispiel nicht nur ihre eigenen WLAN-APs zu einer Lokalisierung verwendenden, sondern alle in der Umgebung verfügbaren WLAN-APs, wie zum Beispiel die ihrer Nachbarn. Weiterhin hatte das Verfahren den Vorteil, dass durch die Lokalisierung auf dem Endgerät ein Tracking des Nutzers des ILS so gut wie unmöglich ist. Bei der Literaturrecherche wurde zudem deutlich, dass das Leistungsverhalten bezüglich Genauigkeit, Zuverlässigkeit und Geschwindigkeit bei der Lokalisierung völlig ausreichend ist, um ein ILS umzusetzen, das jeder Nutzer zu Hause in seinen eigenen Räumlichkeiten installieren kann. Der einzige Nachteil des Verfahrens liegt darin, dass vor Verwendung des ILS in einem ersten Schritt eine initiale Erstellung eines Innenraum-Modells durch Aufnehmen von Fingerprints an diversen Positionen im Raum, erfolgen muss. Erst dann kann eine Lokalisierung durchgeführt werden. Dieser Aufwand ist aber im Vergleich zur Errichtung einer aktiven WLAN-Infrastruktur, der Verwendung von Hochleistungsrechnern, wie diese zum Beispiel bei der WLAN Laufzeitmessung benötigt werden, oder dem Installieren von zahlreichen Bluetooth Landmarken in der Umgebung, eher gering. Abschließend wurde basierend auf der Technologie und Verfahrensauswahl eine Evaluation bereits implementierter, existierender LFPT-ILS untersucht und daraus ein Konzept für die Implementierung eines konkreten Prototypen auf Android Basis erstellt. Dafür wurde zunächst analysiert, welche Anforderungen zum einen ein ILS allgemein hat und zum anderen, was für Anforderungen das Location Fingerprinting mit sich bringt.

8.1.3 Prototyp- und Framework Implementierung

Als das Konzept und die primären Anforderungen festgelegt waren, wurde mit der Implementierung der ersten Version des Smartspace Framework Prototypen begonnen. Der implementierte Prototyp unterstützte zunächst nur das WLAN-Fingerprinting Verfahren und wurde schließlich im Rahmen des zweiten Ambisense Workshops am Wilhelm-Schickard-Institut für Informatik in Tübingen zum ersten Mal vorgestellt. Die ersten Tests und Messungen im Rahmen des Workshops waren sehr vielversprechend und haben gezeigt, dass das Prinzip sehr gut funktioniert. Daraufhin wurde in der zweiten Implementierungsphase aus dem konkreten Prototypen ein allgemeines Framework abgeleitet. Hierzu wurde die Software von Grund auf neu entworfen. Es flossen vor allem die Ergebnisse aus dem ersten Teil der Implementierung mit

in das Design ein. So wurde aufgrund von Performanzproblemen, die bei der Implementierung des Prototypen auftraten, von vorn herein ein Design gewählt, was von Grund auf aufwendiges Multithreading unterstützte. Hierdurch sollten die rechenintensiven Vorgänge der Datenaufzeichnung, der Datenverarbeitung und der Positionierung, sowie die Persistenz stets mit optimaler Ausnutzung der auf dem ME verfügbaren Ressourcen durchgeführt werden. Die Implementierungsdetails des Smartspace Framework können im Detail in Kapitel 5 nachgelesen werden. Im weiteren Verlauf der Arbeit wurde das Framework dann um weitere Technologien wie das GSM-Fingerprinting und die Bewegungserkennung mittels den IMU Sensoren des ME erweitert.

Das fertige Smartspace Framework implementiert eine flexible Architektur und ist beliebig um neue Sensoren, Filter zur Datenverarbeitung und Lokalisierungsalgorithmen erweiterbar. Durch die Multithreading Architektur weist es eine durchweg gute Performanz auf und zusätzlich können alle Systemparameter über ein Konfigurationsobjekt gesetzt werden. Der Verwendung des Frameworks durch andere Entwickler der Android Plattform, in ihren Applikationen, steht somit nichts mehr im Wege.

8.1.4 Performanz-Analyse

Nach Fertigstellung der Implementierung wurde eine Leistungsanalyse des Frameworks durchgeführt. So wurde die Genauigkeit und Zuverlässigkeit des Systems im Hinblick auf drei Kriterien untersucht: Verwendete Technologien, Umgebung, Lokalisierungsgeschwindigkeit.

Die verwendeten Technologien WLAN und GSM sollten vor allem im Hinblick auf das Verhalten beim Einsatz in verschiedenen Umgebungen getestet werden. Die einzelnen Testreihen sollten die System-Performanz im Hinblick auf einstöckige und mehrstöckige Installation untersuchen. Zudem sollte die Messung in der einstöckigen Umgebung in einer sehr realitätsnahen Umgebung durchgeführt werden, um die Performanz in Bezug auf ein realistisches Alltags-Szenario messen zu können. Dazu mussten zunächst geeignete Testumgebungen gefunden werden. Diese sollten nach Möglichkeit mehrere Räume und Stockwerke umfassen und über eine ausreichende Abdeckung mit WLAN-APs und GSM-Zellen verfügen. Dies war für einfache Messungen zunächst die private Wohnung und schließlich wurde für die weiteren in dieser Arbeit vorgestellten Ergebnisse eine Testumgebung mit mehreren Räumen und Stockwerken am Wilhelm-Schickard-Institut für Informatik, in Tübingen im Sand Gebäude 13, ausgewählt. Die Ergebnisse dieser Untersuchungen werden in Kapitel 7 ausführlich vorgestellt.

Zusammenfassung der Ergebnisse

Die Ergebnisse zeigen, dass die Lokalisierung beim Einsatz des WLAN-Fingerprintings in einer mehrstöckigen Installation, mit einer Rastergröße von $5x5$ Metern, bei einer Genauigkeit von ungefähr zwei bis drei Metern liegt und eine Zuverlässigkeit von 70 bis 80% aufweist. Die Stockwerkerkennung mittels WLAN funktioniert ebenfalls sehr gut. Die Zuverlässigkeit bei der Stockwerkerkennung in der mehrstöckigen Installation lag in der Messreihe bei 96%. Die Ergebnisse der WLAN Messreihe zur Simulation eines realistischen Alltagsszenarios in einer einstöckigen Installation im dritten Stockwerk des Sand 13 ergaben sogar eine noch höhere Zuverlässigkeit bei der Lokalisierung von über 80%. Die mittlere Abweichung der Genauigkeit konnte mit $1,41$ Metern im Vergleich zu der dreistöckigen Installation sogar noch gesteigert

werden. Für die Lokalisierung mittels GSM-Fingerprinting konnte gezeigt werden, dass mit einem groben Raster im Bereich von 30 bis 50 Metern eine Zuverlässigkeit von 100% gewährleistet werden kann und dass GSM durchaus zur Unterscheidung von Stockwerken eingesetzt werden kann. In der Messreihe in der dreistöckigen Installation ergab sich eine Genauigkeit von 13 Metern bei der Lokalisierung. Die weiterhin durchgeführten Analysen zur Lokalisierungsgeschwindigkeit sind ebenfalls sehr viel versprechend. Sie zeigen zwar, dass die Zeit zur Lokalisierung eines ME exponentiell mit der Anzahl der Trainingspunkte in der Trainingsphase ansteigt und dass die durchschnittliche Anzahl an APs oder Zellen ebenfalls Auswirkungen auf die Lokalisierungsgeschwindigkeit hat. Nichts desto trotz dauert die Lokalisierung in einer häuslichen Installation mit zehn Trainingspunkten, bei zehn empfangenen APs pro Trainingspunkt, weniger als eine Sekunde. Die Ergebnisse der einzelnen Messreihen werden zum besseren Vergleich nochmals in Tabelle 8.1.4 in der Übersicht dargestellt.

Messreihe	**Rastergröße**	**APs/TP**	**Genaugkeit**	**Zuverlässigkeit**
1: WLAN 3 Stockwerke	5x5	39,72	$2,55m$	$72,45\%$
2: GSM 3 Stockwerke	5x5	4,26	$13,01m$	$15,95\%$
3: GSM 3. Stock WSI	50x50	-	$0,00m$	$100,00\%$
4: WLAN 3. Stock Räume	5x5	44,69	$1,41m$	$81,02\%$

Tabelle 8.1: Übersicht über die Ergebnisse der in der Arbeit durchgeführten Messreihen.

8.2 Erkenntnisse

Weiterhin konnten diverse weitere Erkenntnisse und Herausforderungen aus den in Kapitel 7 beschriebenen Messreihen abgeleitet werden. Bei den durchgeführten WLAN Messreihen war sowohl in der dreistöckigen, als auch in der einstöckigen Installation zu beobachten, dass neben der guten Genauigkeit und Zuverlässigkeit, selbst bei einer fehlerhaften Lokalisierung eines RPs in der RTP, der falsch erkannte RP in der Regel in unmittelbarer Nähe der zu erwarteten Position lag. Die falsch erkannten RPs lagen dabei in der dreistöckigen Messreihe auch durchgehend im selben Stockwerk. Ferner konnte bei der Auswertung der aufgenommenen Messdaten ein Problem identifiziert werden, was so bei der Implementierung nicht beachtet wurde, aber keine Auswirkung auf die Lokalisierung hatte. Die bei den WLAN Messreihen 1 und 4 gemessene Anzahl an WLAN-APs pro TP in der TRP dürfte weit unter der wirklich empfangenen Anzahl an WLAN-APs liegen. Grund hierfür ist, dass die meisten der empfangenen WLAN-APs dem WLAN Netz des Zentrums für Datenverarbeitung (ZDV), der Uni Tübingen zugeordnet werden konnten. Diese Router des ZDV scheinen pro AP über drei SSIDs, „Belwue“ „Guest“ „802.x“ zu verfügen, um jeweils einen anderen Zugang zum Uni WLAN Netz mit VPN und 802.x Authentifizierung und Autorisierung zu gewährleisten. Die BSSIDs (MAC-Adressen) dieser Router unterscheiden sich nur in den letzten beiden hexadezimalen Stellen voneinander. Die Feldstärken von diesen verschiedenen BSSIDs sind aber pro Router relativ gleich bleibend und unterscheiden sich nur geringfügig voneinander. Deshalb kann angenommen werden, dass diese Netze von derselben Hardware erzeugt oder zumindest in einem Gehäuse vereint sind. Dies hatte zwar keinerlei Auswirkungen auf die Genauigkeit und Zuverlässigkeit in der RTP, führte aber aufgrund der stark erhöhten Anzahl an WLAN-APs zu einem signifikanten Einbruch bei der Lokalisierungsgeschwindigkeit in der RTP. Die Ursache hierfür ist, dass die tatsächliche Anzahl an aussagekräftigen WLAN-APs um rund ein Drittel

geringer war. So betrug die tatsächliche Anzahl an WLAN-APs in der Messreihe 1 an TP0 sogar 50 anstatt gerade einmal 26 WLAN-APs.

Aus den GSM-Messreihen konnten ebenfalls weitere Erkenntnisse bezüglich der GSM-Infrastruktur und seiner zukünftigen Verwendung abgeleitet und wichtige Rückschlüsse auf die zukünftige Verwendung in adaptiven Algorithmen des Smartspace Frameworks gezogen werden. Die empfangenen CIDs des GSM-Netzes deuten darauf hin, dass die GSM-BTSs am Sand 13 des WSI in vier Sektoren aufgeteilt sind. So konnten nach Evaluierung der Daten der Messreihe 2, die empfangenen CIDs: 0,2,7 einer GSM-BTS zugeordnet werden, genauso wie die CIDs: 201, 202, 204 und 205. Je nach relativer Position zum WSI können zudem 2-3 Zellen pro BTS empfangen werden. Bei den anderen empfangenen Zellen scheint es sich um weiter entfernte Zellen zu handeln, die nur sporadisch empfangen werden konnten. Trotz dieser Ergebnisse konnte aber keine eindeutige Zuordnung zum Empfang einzelner Zellen in einem bestimmten Teil der Installation ausgemacht werden. Dies ist letztendlich ein erwartetes Verhalten, da eine GSM-BTS meistens 3-4 Sektoren besitzt, durch die verschiedene Winkel abgedeckt werden. Dieses Verhalten ist aber durchaus interessant und muss weiter untersucht werden, weil die Sektoren einer GSM Zelle in Zukunft durchaus mit in die Berechnung der Position mit einbezogen werden könnten. So ist es denkbar, unter Zuhilfenahme der empfangenen CID, Rückschlüsse auf die grobe Position der BTS-Sektoren und somit auf die grobe Richtung, in der sich das ME befindet[3] zu ziehen. Diese Daten könnten dann weiterhin mit den Daten, die der elektrische Kompass eines ME liefert, korreliert und fusioniert werden, und dazu genutzt werden, bestimmte Zellen an manchen RPs in der RTP von vornherein auszuschließen und weder bei der Erstellung des Innenraum-Modells in der TRP als auch in der RTP nicht mit abzuspeichern. Hierzu müsste aber vor der Erstellung des eigentlichen Modells eine genaue Analyse anhand von Testmessungen erstellt werden. Diese Beobachtung könnte bei näherer Untersuchung aber durchaus die Genauigkeit und Zuverlässigkeit des GSM-Fingerprinting Verfahren erhöhen, weil zusätzlich zur Messung der Feldstärken eine Optimierung durch Einbeziehung der Zell-Sektoren zur Lokalisierung herangezogen werden könnte.

Des weiteren haben die Ergebnisse der Messreihe 3 gezeigt, dass GSM zur einer groben Lokalisierung auf Granularität zum Beispiel eines Arbeitsbereichs, wie der Lehrstuhls für Rechnerarchitektur, mit einer sehr hohen Genauigkeit bestimmt werden kann. Diese Tatsache könnte zu einem ganz neuen Einsatz der GSM Technologie innerhalb eines multimodalen Lokalisierungssystems aus WLAN und GSM ausgenutzt werden. So könnte anhand der durch GSM verfügbaren Positionen jeweils eine bereichsspezifische RM in einer größeren Installation geladen werden. So könnte die Größe der RM signifikant verringert werden, was einen hohen Performanz-Zuwachs bei der Lokalisierung zur Folge hätte, weil immer nur die WLAN-RM der unmittelbaren Umgebung geladen würde. Diese Bereichszuordnung von RMs wäre auch mit WLAN umsetzbar. Letztendlich ist es egal welche Technologie für diesen Ansatz verwendet wird. Es ergeben sich dadurch ganz neue Möglichkeiten das Smartspace Framework in großen Installationen mit hunderten von TPs zu verwenden und trotzdem ein gutes Leistungsverhalten bezüglich der Lokalisierungsgeschwindigkeit zu erzielen.

Interessant ist auch die hohe Stabilität, die bei der Lokalisierung bei der Auswertung der GSM-Messung festgestellt wurde. Trat es bei den WLAN-Messreihen immer wieder auf, dass es während der RTP an einem RP zu einer Änderung der erkannten Positionen kam, war dies bei GSM überhaupt nicht zu beobachten. Nach mehr als zwei Messungen änderte sich die

[3] Mit Richtung ist hier der Sektor der Antenne der BS gemeint, indem sich das Gerät befindet.

erkannte Position nicht mehr und blieb auch nach längeren Messungen konstant. Dieses Verhalten konnte an allen aufgenommenen GSM-TPs beobachtet werden und deutet darauf hin, das GSM im Vergleich zu WLAN ein sehr stabiles Netz ist, welches eine geringere Varianz bezüglich der Signalcharakteristika aufweist. Dies wird auch durch Messreihe 3 bestätigt, welche bei der Indoor-Lokalisierung mittels des GSM-LFPT-Verfahren sehr konstante Ergebnisse lieferte, wenn das Raster entsprechend gewählt wurde.

Nach der Auswertung der Ergebnisse zur Performanz des Smartspace Frameworks lässt sich zunächst feststellen, dass sich mit dem Smartspace Framework eine zuverlässige Lokalisierung, basierend auf WLAN, bezüglich der Anwendungsszenarien zum Aufbau eines eigenen ILS mit einer Granularität von Arbeitsräumen oder Zimmern in einer Wohnung durchführen lässt und die Lösung absolut alltagstauglich ist. Ebenso ist die Genauigkeit bei der Verwendung von GSM ausreichend, um ein System zu implementieren, welches den Aufenthaltsort, zum Beispiel eines Mitarbeiters, immerhin mit der Genauigkeit eines Arbeitsbereiches innerhalb eines großen Betriebes bestimmen kann. In den eigenen vier Wänden könnte hiermit unterschieden werden, ob man sich im Erdgeschoss, dem ersten Stock, dem Garten oder der Garage befindet.

Überdies lässt sich feststellen, dass eine feingranulare Lokalisierung im Zentimeterbereich, wie sie für einige der in der Einführung vorgestellten Echtzeit-Navigationssysteme benötigt wird mit dem Smartspace Frameworks, nicht durchführbar ist. Dies ist für viele Anwendungsfälle aber nicht relevant und ist eine der Erkenntnisse, welche im Lauf der Diplomarbeit gewonnen werden konnte. So gibt es viele Anwendungsfälle, in denen das Zentimeter genaue Lokalisieren eines ME überhaupt nicht erforderlich ist. So reicht es in den meisten Anwendungsfällen, in denen eine Indoor-Lokalisierung auf Basis eines ME durchgeführt werden soll, vollkommen aus, dass ME auf drei bis fünf Meter genau lokalisieren zu können. vor allem vor dem Hintergrund, dass zur Verwendung des Smartspace Frameworks keinerlei Funk- oder Serverinfrastruktur zur Lokalisierung mittels des ME installiert werden muss. Soll eine Anwendung zum Beispiel dazu eingesetzt werden, Teilnehmer auf einer Konferenz zu lokalisieren, reicht es vollkommen aus, wenn die Teilnehmer auf Raum-Granularität lokalisiert werden können. Ähnlich verhält es sich für das in der Einführung beschriebene Anwendungsszenario in einem Krankenhaus, bei dem Patienten im Notfall innerhalb einer Klinik lokalisiert werden sollen. Dasselbe gilt für den Einsatz des Smartspace Framework in den Räumlichkeiten von privaten Personen. Bei allen Anwendungsszenarien ist eine Genauigkeit von einigen wenigen Metern völlig ausreichend. Genau in diesen Anwendungsszenarien liegt nach Evaluation der Messreihen auch die Stärke des implementierten Smartspace Framework: Es bietet eine ausreichende Genauigkeit, um in vielen Indoor-Lokalisierungs-Szenarien eingesetzt werden zu können und ist dabei sehr einfach in den jeweiligen Umgebungen zu installieren.

9 Ausblick

Diese Arbeit zeigt, dass eine Indoor-Lokalisierung auf Android basierten mobilen Endgeräten funktioniert und effizient umgesetzt werden kann. Aufgrund der Tatsache, dass SSF als Open-Source Implementierung jedem zur freien Verfügung steht, sollen hier mögliche Ideen und Anregungen zur Erweiterung des Frameworks gegeben werden. So könnten zukünftige Weiterentwicklungen des SSF darin bestehen, neue Technologien zu integrieren, neue Verfahren zur Verarbeitung der Sensor Daten zu entwickeln und intelligente Algorithmen bezüglich der verwendeten Technologien und einer geeigneten Sensor Fusion, je nach Kontext, zu entwerfen.

Die Integration neuer Technologien in das SSF ist aufgrund der Implementierung als ein erweiterbares Framework ohne Weiteres möglich. So könnten in zukünftigen Arbeiten neue Technologien wie Bluetooth und RFID in das SSF integriert werden. Vorstellbar wäre zum Beispiel die Integration eines Bluetooth-RFID-Scanner, wie der von der Firma Baracoda [Bar] entwickelte TagRunners-Bluetooth-RFID-Scanner, über die Bluetooth API des Android Application Frameworks. Ebenfalls könnte die Bluetooth Technologie der mobilen Endgeräte in das Framework integriert werden und Näherungs-, Fingerprinting- und Partikel Filter-Implementierung beider Technologien umgesetzt oder auf das SSF portiert werden. Zusätzlich könnten in zukünftigen Versionen weitere Verfahren zur Datenverarbeitung in das Framework hinzugefügt werden. So könnte das vorgestellte Signal-Strength-Difference Verfahren vollständig implementiert werden. Darüber hinaus ist vorstellbar, die Komponente zur Bewegungserkennung, um ein Modul zur Schritterkennung unter Zuhilfenahme der IMU-Sensoren zu erweitern, um das in Kapitel 3 beschriebene relative Dead-Reckoning Positionierungsverfahren zu implementieren und eine Evaluation bezüglich des Leistungsverhaltens durchzuführen.

Den wichtigsten Teil stellt aber sicherlich die Implementierung neuer Algorithmen zur Lokalisierung dar. Auf Basis der bereits implementierten Technologien WLAN und GSM könnten so die bestehenden Verfahren um neue Algorithmen, wie den vorgestellten Similarity-Matching Algorithmus, ergänzt werden. Hierdurch könnten Fingerprints besser miteinander verglichen und damit auch deutlicher unterschieden werden. Dabei müssten insbesondere die Auswirkungen von zahlreichen schwachen APs auf die Genauigkeit und Stabilität eines LFPTs anhand weiterführenden Messungen untersucht werden. Ferner könnten ganz neue Algorithmen entwickelt werden, welche in Echtzeit intelligente Entscheidungen anhand des Umgebungszustandes wie Genauigkeit, Verfügbarkeit, Indoor-Outdoor-Position, bezüglich der verwendeten Technologien und Lokalisierungsverfahren treffen. Ein primitiver Algorithmus könnte zum Beispiel das Verhalten implementieren, falls kein WLAN verfügbar ist, automatisch eine Lokalisierung mittels GSM durchzuführen und umgekehrt. Ein weiterer Algorithmus könnte Optimierungen bezüglich der Lokalisierungsgeschwindigkeit implementieren und durch eine weiterführende Kombination von WLAN- und GSM-Fingerprints, GSM, wie in der Diskussion beschrieben, nur als Näherungsverfahren eingesetzt werden, um die grobe Position zu bestimmen. Anhand dieser wird dann auf dem mobilen Endgerät die für diese Position relevante Radio Map geladen und verwendet. Durch diese und andere Vorgehensweisen könnte die Lokalisierung des SSF kontinuierlich verbessert werden. Auch das in Kapitel 5.3.5 vorgestellte Prinzip des GEO-

Mappings könnte in weiteren Arbeiten vollständig implementiert werden und einen saumenlosen Übergang zwischen Indoor- und Outdoor-Lokalisierung realisieren.

Es bleibt zu hoffen, dass das SSF, in Verbindung mit dem Potential der inzwischen 180.000 Entwickler umfassenden Android Gemeinde, eine völlig neue Art von standortbezogenen Diensten hervorbringen wird. Entwickler könnten eine saumenlose Integration von Indoor- und Outdoor-Lokalisierung in ihren Applikationen bereitstellen und somit völlig neue Anwendungsszenarien im Geschäftsbereich als auch in der privaten Nutzung implementieren.

A Abkürzungsverzeichnis

AOA	Angle of Arrival
AAF	Android Application Framework
AIDL	Android Interface Definition Language
AIL	Asynchrones Interval Labeling
AP	WLAN Access Point
AR	Augmented Reality
BER	Bit Error Rate
BCCH	Broadcast Control Channel
BSC	Base Station Controller
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CBR	Consumer Based Routing
CDMA	Code Division Multiple Access
CID	Cell ID
CoF	Core Framework
COO	Cell of Origin
DVM	Dalvik Virtual Machine
DLOS	Direct Line of Sight
DNS	Domain Name System
DTOA	Differential TOA
DDP	Dominant Direct Path
DGPS	Differential GPS
FDM	Freuquency Division Multiplexing
FPT	Fingerprint
FSM	Finite State Machine
GC	Geocoding
GBS	Gelocation Basis Station
GCS	Geolocation Control Station
GML	Geography Markup Language
GPS	Global Positioning System
GPX	GPS Exchange Format
GSM	Global System for Mobile Communications
HHM	Hidden Markov Modell
HTC	High Tech Computer
IBR	Interest Based Routing
ILS	Indoor Lokalisierungs System
IMU	Inertial Measurement Units
IPC	Inter Process Communication
IrDA	Infrared Data Association
JVM	Java Virtual Machine
KML	Keyhole Markup Language

KNNSS	K Nearest Neighbor in Signal Space
LFPT	Location Fingerprint
LOC	Lines of Code
LBS	Location Based Services
LOS	Line of Sight
LS	Learning State
MAC	Medium Access Control
MBA	Mobile Based Architecture
MCC	Mobile Country Code
MNC	Mobile Network Code
ME	Mobiles Endgerät
MT	Mobile Tagging
MS	Motion State
NBA	Network Based Architecture
NDDP	Nondominant Direct Path
NLOS	No Line of Sight
NNSS	Nearest Neighbor in Signal Space
NSS	Network Switching Subsystem
OHA	Open Handset Alliance
OSI	Open System Interconnet
OSS	Operation Subsystem
POI	Point of interest
QR	Quick Response
RF	Radio Frequency
RFID	Radio Frequency Identification
RM	Radio Map
RP	Realtime Punkt
RSS	Received Signal Strength
RTLS	Realtime Location Systems
RTFPT	Realtime Fingerprint
RTP	Realtime Phase
RTS	Realtime State
SBS	Strongest Base Station
SDK	Software Development Kit
SMA	Similarity Matching Algorithmus
SNR	Signal to Noise Ratio
SSD	Signal Strength Diffenrence
SSF	SmartSpace Framework
TDM	Time Division Multiplexing
TOA	Time of Arrival
ToF	Time of Flight
TRP	Trainings Phase
TP	Trainings Punkt
SeF	Sensing Framework
SID	Symbolische Identifier
SSF	Smartspace Framework
UDP	Undetected Direct Path
UMTS	Universal Mobile Telecommunications System

VAP	Virtual Access Point
WEP	Wired Equivalent Privacy
XML	Extensible Markup Language
WiMAX	Worldwide interoperability for Microwave Access
WLAN	Wireless Local Area Network
WPS	Wireless Positioning Services
WPA	WiFi Protected Access
WSI	Wilhelm Schickard Institut
ZDV	Zentrum für Datenverarbeitung

B Grundlagen verwendeter Entwurfsmuster

Bei der konkreten Beschreibung der Implementierung des SSF ist es nützlich, ein paar grundlegende Begriffe der verwendeten objektorientierten Entwurfsmuster zu kennen. Dies soll vor allem dem besseren Verständnis dienen, aber auch die Kommunikation über die einzelnen Framework-Komponenten erleichtern. In jeder Ingenieurswissenschaft gibt es eine Reihe von „Best Practices“, eine Sammlung an Lösungen für immer wieder auftretende Probleme, auf die ein Ingenieur bei Bedarf zurückgreifen kann. So auch in der Softwaretechnik: Entwurfsmuster sind erprobte Herangehensweisen an Problemstellungen, die bei der Programmierung immer wieder auftreten. Sie verleihen dem Code hohe Robustheit und Eleganz und sparen Zeit, weil ein Entwickler nicht für jedes Problem das Rad neu erfinden muss. Durch Einsatz des richtigen Entwurfsmusters an der richtigen Stelle kann die Erstellung vereinfacht sowie Wartung und Erweiterbarkeit von Code deutlich erhöht werden. Des Weiteren sind Entwurfsmuster insbesondere in großen Teams und großen Projekten eine solide Basis für ein einfaches, grundlegendes, gemeinsames Vokabular. Dadurch, dass Entwurfsmuster im Code erkannt werden können und dass man leicht über sie sprechen kann, erhöhen sie die Lesbarkeit und fördern somit die schon angesprochene Wartbarkeit des Codes.

Bei der Erstellung des SSF wurde darauf geachtet, dass diese nichtfunktionalen Eigenschaften der Programmierung stets berücksichtigt werden, insbesondere vor dem Hintergrund des angestrebten Einsatzes des SSF. Dies erleichtert die Arbeit für Entwickler, die das SSF erweitern und über die bereitgestellte API hinaus verwenden möchten und führt nicht zu einer unnötigen Frustration beim Versuch, die Implementierung nachzuvollziehen.

Im Verlauf der Arbeit wurden die Begrifflichkeiten der Entwurfsmuster ohne weitere Erläuterung verwendet. Im Folgenden werden die wichtigsten Entwurfsmuster, die im Zusammenhang mit dem SSF verwendet wurden, zum besseren Verständnis kurz vorgestellt.

B.1 Composite

Das Composite-Entwirfsmuster [GHJV95] erlaubt es, Bäume, eine spezielle Art von Graphen, in einer objektorientierten Datenstruktur abzubilden. Es besteht aus zwei Komponenten: den inneren Knoten, den sogenannten Composites, und den Blättern. Composites sind in der Lage, andere Composite-Komponenten oder die zweite Art der Komponenten, die sogenannten Blätter, aufzunehmen. Blätter können keine weiteren Elemente aufnehmen und bilden somit das Ende der Baumstruktur. Einsatz findet das Composite Entwurfsmuster im SSF bei der Organisation und Persistenz eines LFPTs, welcher als Baumstruktur abgebildet ist.

B.2 Visitor

Das Visitor-Entwurfsmuster [GHJV95] steht in direkter Beziehung zum gerade vorgestellten Composite-Entwurfsmuster, und dient der Traversierung von objektorientierten Baumstrukturen. Das Entwurfsmuster gibt zwei Schnittstellen vor, die implementiert werden müssen, um eine Traversierung zu ermöglichen: Zum einen die Visitor Schnittstelle, welche von konkreten Visitorklassen implementiert werden muss, zum anderen eine Schnittstelle, welche von jedem Knoten im Baum implementiert werden muss, um „besucht“ werden zu können. Wie diese Schnittstellen im SSF im Detail aussehen, wurde im Kapitel 6.2.1.4 beschrieben. Es ist wichtig, sich die Funktionsweise von Visitoren gut zu verdeutlichen, da ein Visitor sich durch sein hohes Maß an Entkopplung von im Fall des SSF dem LFPT und seiner Manipulation sehr gut eignet, um ein flexibles auslesen der LFPTs zu gewährleisten.

B.3 Template-Method

Das Template-Method-Entwurfsmuster [GHJV95] gibt in einer als final deklarierten Methode das Gerüst eines Algorithmus vor. Genauer: die Aufrufreihenfolge von Methoden einer Klasse wird vorgegeben. Dies erlaubt es, die Struktur von Algorithmen beizubehalten, aber jederzeit die Implementierung auszutauschen zu können.

B.4 Reactor

Das Reactor-Entwurfsmuster [BHS07] ist der Klasse der „Event Demultiplexing und Dispatch Patterns“ zuzuordnen. Das Entwurfsmuster erleichtert den Umgang mit eventgetriebener Software, in dem es ein Demultiplexing der ankommenden Events vornimmt und die Events an einen geeigneten Event-Handler weiterleitet, welcher den ankommenden Event verarbeitet. Die Verarbeitung der ankommenden Events findet im Reactor synchron statt, das heißt der Reactor wartet ab, bis der Handler zurückkehrt und verarbeitet erst dann den nächsten Event.[1] Das Reactor-Entwurfsmuster wird im SSF eingesetzt, um die ankommenden Daten der verschiedenen Sensoren zu demultiplexen und an ihre entsprechenden Handler weiterzuleiten, welche sich um die Verarbeitung der Daten durch einen entsprechenden Dienst kümmern.

B.5 Strategy

Das Strategy-Entwurfsmuster [GHJV95] ermöglicht, ein austauschbares Verhalten von Objekten unabhängig von der Implementierung zu realisieren und dient der Umsetzung unterschiedlicher, beziehungsweise alternativer Algorithmen in einem Objekt. Eine Strategie kapselt eine Familie von Operationen, die sich zusammen ändern, in Form einer Schnittstelle. Die Verwendung der Strategie durch eine andere Klasse funktioniert so, dass die Klasse eine Instanz des gewünschten Strategieobjektes hält und die Methodenaufrufe nicht selber implementiert, sondern an das Strategieobjekt delegiert. Dies ermöglicht es der Klasse, durch Austausch der Instanz des Strategieobjekts ein völlig anderes Verhalten zu exprimieren. Eingesetzt wird das

[1] Die synchrone Verarbeitung von Events kann zu Problemen führen, wenn der durch den Handler ausgeführte Service lange Rechenzeit beansprucht und nicht direkt zurückkehrt. In diesem Fall kann auf das Proreactor-Entwurfsmuster [BHS07] zurückgegriffen werden, welches eine asynchrone Verarbeitung der ankommenden Events bereitstellt.

Strategie-Entwurfsmuster von der Finite State Machine (FSM) des SSF, um je nach aktuellem Zustand eine andere Strategie zum Ansteuern der Framework-Komponenten zu verwenden.

B.6 Singelton

Das Singelton-Entwurfsmuster [GHJV95] sichert zu, dass es von einer Klasse immer nur genau eine Instanz gibt und stellt einen globalen Zugriff auf dieses statische Objekt bereit.

B.7 Registry

Ein Registry-Objekt nach dem Registry-Entwurfsmuster [GHJV95] erlaubt das Ablegen von globalen Applikations-Objekten und -Daten anhand eines eindeutigen Schlüssels. Der Zugriff auf die globalen Variablen erfolgt in der Regel über statische Methoden auf der Registry. Sinnvoll ist der Einsatz einer Registry vor allem in Schichtenarchitekturen, in denen mehrere Schichten auf dieselben globalen Objekte und Daten zugreifen müssen. SSF implementiert eine Registry, um einen eindeutigen Zugriffspunkt auf diverse Systemkomponenten, wie zum Beispiel einen von einem Sensor definierten Event-Handler und andere Datenstrukturen zu gewährleisten. Die Datenstrukturen sind über die Registry im ganzen Framework abrufbar und dienen als Datenstruktur-Prototypen, von denen über die Registry eine Kopie angefordert werden kann.

B.8 Fassade und Proxy

Das Fassade- und auch das Proxy-Entwurfsmuster [GHJV95] vereinfachen die Benutzung eines Systems durch Zusammenfassen mehrerer Klassen oder Schnittstellen in einem Objekt. Die Fassade dient damit als die exklusive öffentliche Schnittstelle eines Subsystems nach außen, während die komplexen Klassen des Subsystems die eigentliche Funktionalität implementieren.

C Die Android Plattform

Android ist ein modernes Betriebssystem für MEs. Android wurde 2003 von der Firma Android Inc. in Palo Alto (Kalifornien) ins Leben gerufen und im Jahre 2005 von Google Inc. aufgekauft. Die Android-Plattform ist freie Software und wurde 2007 unter der Apache-Lizenz 2.0 [OSI] von der Open Handset Alliance (OHA)[1] veröffentlicht. Der Quellcode des Betriebssystems ist frei verfügbar und kann auf der Webseite der OHA heruntergeladen werden [Alld].

C.1 System Architektur

Android wurde speziell für die Anforderungen heutiger MEs entwickelt. Zu den Anforderungen zählen eine permanente Internetverbindung über WLAN- oder Mobilfunknetze, die Positionsbestimmung durch ein GPS Modul und die Möglichkeit, über zusätzliche Sensoren, wie zum Beispiel den Beschleunigungs-, Lage- und Orientierungssensoren, Informationen über aktuelle Kontexte des ME zu erhalten. MEs, die mit einem Android Betriebssystem ausgestattet sind, müssen deshalb gewisse Hardware Anforderungen erfüllen, um den Software-Stack von Android zu unterstützen. Eine Übersicht der gesamten Architektur findet sich in Abbildung C.1.

Die Architektur von Android [Allf] ist in mehrere Schichten aufgeteilt. So verwendet Android einen Standard-Linux-Kernel als Hardware-Abstraktionsschicht für Speichermanagement, Prozess-Scheduling und Input/Output. In der darüber liegenden Schicht werden ergänzend mehrere Open-Source-Bibliotheken und Frameworks, wie zum Beispiel SQLite [Conb], Webkit [Web], OpenGL [Ope] und OpenCORE [Vid] zur Anwendungsentwicklung verwendet.[2] Den Kern der Android Laufzeitumgebung bildet die speziell auf mobile Endgeräte zugeschnittene virtuelle Maschine, die Dalvik Virtual Machine (DVM) [Bor], welche maßgeblich von Dan Bornstein, einem Mitarbeiter von Google, entwickelt wurde. Die DVM basiert auf der quelloffenen Java Virtual Machine [Mic] (JVM) Apache-Harmony [Fou] und arbeitet im Vergleich zur stackbasierten JVM registerbasiert. So kann die DVM den Vorteil der meist registerbasierten ARM [ARM] Prozessoren der MEs besser ausnutzen. Weiterhin verwendet die DVM ein optimiertes Bytecode Format namens DEX [Ret]. Hierzu wird der Java-Quellcode einer Android-Anwendung mit dem Standard-Java-Compiler zuerst in .class Dateien übersetzt und dann mit Hilfe des DX-Cross-Compilers in .dx Dateien übersetzt. Anwendungen werden in Android mit Hilfe des Software Development Kit (SDK) [Allc] entwickelt. Das SDK enthält alle Software-Tools und Java-Bibliotheken, die zu der Erstellung einer Android-Anwendung benötigt werden. Hierzu zählen unter anderem die Pakete:

- java und javax (JDK),
- android (Android-Application-Framework),

[1] Die OHA ist ein Konsortium, bestehend aus 50 führenden Technologie- und Softwarefirmen, darunter Google, HTC, Intel, Motorola, Qualcomm, Texas Instruments, Samsung, LG, T-Mobile und Nvidia.

[2] Es kommt hierbei auch eine angepasste Version der Standard libc, mit einer Größe von nur 200kb, zum Einsatz.

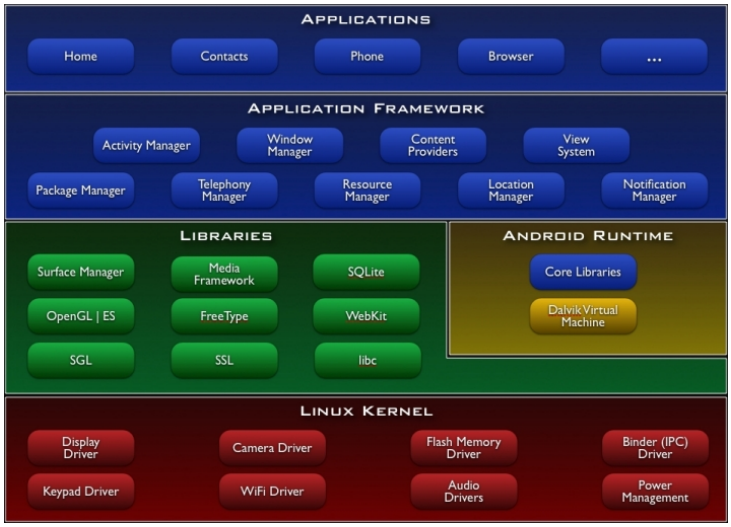


Abbildung C.1: System Architektur nach [Allf]

- junit (Testframework),
- org.apache.commons (Apache Commons Projekt),
- org.json und org.w3c.dom (Verarbeitung/Manipulation von JSON beziehungswiese W3C-DOM).

Anwendungen werden komplett in Java programmiert, wobei das Android-Application-Framework (AAF) für geschwindigkeitskritische Operationen auf die oben erwähnten, direkt in C oder C++ geschriebenen nativen Bibliotheken zurückgreift. Entwickler können für spezielle Anforderungen aber auch das Native Development Kit (NDK) [Allb] nutzen, welches Anwendungsprogrammierung in den hardwarenäheren Sprachen C und C++ ermöglicht.

C.2 Komponenten

Alle Anwendungen, die auf der Android-Plattform ausgeführt werden, sind gleichberechtigt. Das heißt, jede Anwendung läuft in ihrem eigenen Linux Prozess, in einer eigenen DVM. Jede Anwendung hat eine eindeutige ID innerhalb des Systems, welche dazu dient, mit Hilfe der Unix-Rechte-Verwaltung, unberechtigte Zugriffe auf die Daten anderer Anwendungen zu unterbinden. Anwendungen selber können nur gegen die API des AAFs implementiert werden. Die API ist eine javabasierte Abstraktionsschicht, welche die Interaktion mit dem Betriebssystem bereitstellt. Android Anwendungen bestehen aus verschiedenen Komponenten, davon gibt es in Android genau folgende vier:

Activities stellen genau einen Bildschirm in der Benutzerschnittstelle dar. Activities sind die Komponenten einer Anwendungen, die für die direkte Interaktion mit dem Nutzer zuständig sind. Darüber hinaus sind sie verantwortlich für die Darstellung der Daten in der Benutzerschnittstelle, der Entgegennahme von Nutzereingaben und der Zustandsverwaltung von Komponenten. Activities werden vom Betriebssystem durch ein spezielles Lebenszyklus-Management verwaltet. Eine Anwendung enthält im Normalfall mehrere miteinander verknüpfte Activities, zwischen denen je nach zu erledigender Aufgabe hin und her geschaltet werden kann.

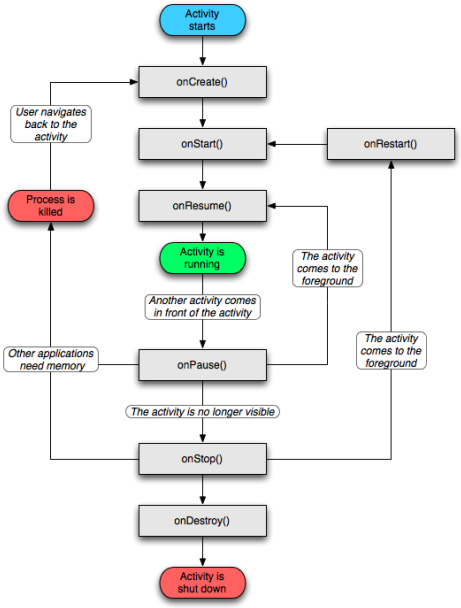


Abbildung C.2: Lebenszyklus einer Android Anwendung nach [Alla]

Services sind den Activities sehr ähnlich und unterscheiden sich nur darin, dass sie keine eigene Benutzerschnittstelle haben. In Services werden Aufgaben ausgeführt, die im Hintergrund laufen sollen. So wird in dieser Arbeit ein Service zum Beispiel dazu verwendet, Sensor-Daten im Hintergrund aufzuzeichnen und zu verarbeiten. Werden Services in Verbindung mit einer Activity benutzt, kann die Kommunikation transparent über ein IPC ähnliches Konzept namens Binder erfolgen. Services können hierbei sowohl als Thread im Prozess der Anwendung oder auch als eigenständiger separater Prozess laufen. Bei letzterem muss aber auf den Android spezifischen Inter-Process-Communication (IPC) Mechanismus zurückgegriffen werden, der die Android-Interface-Definition-Language (AIDL) nutzt, um die Objektserialisierung über Prozessgrenzen hinweg zu gewährleisten.

Broadcast Receiver stellen die Kommunikation auf der Anwendungsschicht bereit. Durch die komponentenbasierte Architektur von Anwendungen ist ein Mechanismus notwendig, durch den die Kommunikation zwischen Betriebssystem und Anwendung, beziehungsweise die Kommunikation von Komponenten untereinander gesteuert wird.

Content Provider Die Verwendung einzelner Komponenten führt zu einer losen Kopplung von Teilen der Anwendung. Dies bietet ein hohes Maß an Wiederverwendung, bedarf aber eines Kommunikationsmechanismus und eines intelligenten Prozessmanagements.

C.3 Prozess-Management

Die Android Plattform verwendet ein prozessorientiertes Lebenszyklusmanagement für Anwendungen. Aufgrund der Tatsache, dass mobile Endgeräte oft nur über limitierte Hardwareressourcen verfügen, muss das Betriebssystem in die Lebensdauer von Prozessen eingreifen, um im Fall von knappen Ressourcen einige Prozesse beenden zu können. Welcher Prozess beendet wird entscheidet sich anhand der Priorität des Prozesses. So hat zum Beispiel eine Activity,

die gerade angezeigt wird, eine sehr hohe Priorität und wird vom Prozessmanagement nur in Ausnahmefällen beendet, während eine Activity, die aktuell als Hintergrundprozess läuft, weil Sie nicht angezeigt wird oder keine laufenden Services enthält, eher eine geringe Priorität zugewiesen wird und deshalb beendet werden kann. Diese Art des Prozessmanagements hat den Vorteil, dass die Kontrolle über eine einzelne Anwendungen beim Betriebssystem selber liegt und die Anwendungen nur über Betriebssystem-Events informiert werden und ihr somit die Möglichkeit gibt, entsprechend zu reagieren.

Das Lebenszyklusmanagement hat somit mehrere Implikationen auf die Entwicklung von Komponenten. Komponenten müssen zu jedem Zeitpunkt damit rechnen, beendet oder in den Hintergrund versetzt zu werden. Damit hierbei keine Daten verloren gehen, werden im Rahmen des Lebenszyklusmanagements nacheinander mehrere Methoden auf der aktuellen Komponente aufgerufen, welche der Anwendung die Möglichkeit bietet, sich in einen konsistenten Zustand zu versetzen. Das gesamte Lebenszyklusmanagement der Android-Plattform ist in Abbildung C.2 dargestellt.

C.4 Apps und App-Stores

Applikationen werden auf heutigen mobilen Plattformen in Form von Miniprogrammen, sogenannten „Apps“ ausgeliefert. Apps sind hierbei in sich geschlossene Komponenten, die als funktionelle Erweiterungen der Plattform zu verstehen sind. Die Apps laufen in der Android-Plattform in einer Laufzeitumgebung, welche das Lebenszyklusmanagement übernimmt und die Applikation durch Verwendung des Callback-Entwurfsmusters beim Eintreten bestimmter Events zurückruft, um Informationen, die zum Beispiel durch einen Hardwaresensor aufgezeichnet wurden, der Applikation mitzuteilen. Diese Apps werden von Entwicklern auf der ganzen Welt geschrieben und in ein sogenanntes App-Store eingestellt. Ein App-Store wird dann von den Endnutzern verwendet, um die Apps finden und installieren zu können.

C.5 HTC G1 Hardware Spezifikation

Zur Umsetzung des Frameworks und zur Durchführung der Testmessungen wurde in dieser Arbeit ein Android basiertes Testgerät vom Typ G1 verwendet. Das bereits in Abbildung 7.2 dargestellte Gerät wurde vom taiwanischen Hersteller High Tech Computer (HTC) unter dem Codenamen HTC Dream 100 entwickelt. Tabelle C.5 gibt einen detaillierten Überblick der G1-Hardware-Spezifikation.

Betriebsystem	Android 1.6 (Linux Kernel 2.6)
Processor (CPU)	Qualcomm MSM7201A Architecture: 32-bit RISC (65nm) Clock rate: 528 MHz Core: ARM1136EJ-S Instruction set: ARMv6
Memory	Samsung Multichip Memory MCP K5E2G1GACM ROM: 256-MB NAND Flash RAM: 196-MB DDR SDRAM (64 MB baseband chip)
Display	Sharp Colored transflective TFT display Size: 3.2-inch TFT-LCD flat touch-sensitive screen Resolution: 320 x 480 (HVGA) Touchscreen Controller: Synaptics 1007A (Synaptics Clearpad Capacitive) Colors: 16 bit/pixel (65536 scales) Dot Pitch: 0.1404 millimetre/pixel
Input Device	Keyboard: Slide-out 5-row QWERTY keyboard Trackball: EVQWJN Jog Ball Module from Panasonic
Radio	Transceiver: Qualcomm RTR6285 Downlink: up to 7.2 Mbps (HSDPA) Uplink: up to 2 Mbps (HSUPA) HSDPA: 3GPP release 5 compliant, UE category 8 HSUPA: 3GPP release 6 compliant, UE category 5 Quad-band GSM/GPRS/EDGE/UMTS: GSM850, GSM900, GSM1800 GSM1900, UMTS1700, UMTS2100 Datenverbindungen: CSD, GPRS, EDGE, UMTS, HSDPA, HSUPA Antenna: internal MCC-MNC-UC-ID SAR (head) GSM1900: 0.547 W/kg SAR (body) UMTS1700: 0.732 W/kg
GPS	Integrated into Baseband Processor (Qualcomm MSM7201A) Protocol: NMEA 0183 (Supports Version 3.0 or above) Antenna: internal GPS Service: Assisted GPS (A-GPS), QuickGPS GPS Chipset: Qualcomm MSM7201A gpsOne Sensitivity: -145 dBm cold start, and -155 dBm for tracking Hot start: 8 sec. Warm start: 60 sec. Cold start: 75 sec. Update rate: once/1sec Accuracy Position: $< 15m95\,\%$ typical Accuracy Velocity: $0,05ß, m/sec$ steady state
Bluetooth	Texas Instruments BRF6300 Connectivity 802.15 Bluetooth 2.0 with Enhanced Data Rate Bluetooth profiles: GAP, RFCOMM, HFP, HSP
Wifi	Texas Instruments WL1251B WLAN IC Power Amplifier: WL1251FE Supported Standards: IEEE 802.11b, IEEE 802.11g, 54 Mbit/s
USB	HTC ExtUSB (11-pin mini-USB 2.0 and audio jack in one) USB 2.0 client, Hi-Speed (480 Mbit/s) USB Series: Mini-B (mini-USB) Connector: SMSC USB3316

Tabelle C.1: HTC G1 Hardware Spezifikation

D Weitere Code-Listings

Listing D.1: Beispielimplementierung zur Verwendung des SSF Service in einer Android Activity.

```
public class IndoorLocatorExample extends Activity {

  private static final String LOG_TAG = "IndoorLocatorExample";

  protected static final int MSG_SSF_LOC = 0x0101;
  protected static final int MSG_LOG = 0x0102;

  private SSFLocationManager locationMngr;
  private SmartSpaceFramework.SSFBinder ssfBinder;
  private SmartSpaceFramework ssf;
  private IGeoPoint currentLocation = new IGeoPoint("default", 0, 0,
      0);
  private Handler handler = new Handler() {
    @Override
    public void handleMessage(final Message msg) {
      switch (msg.what) {
        case MSG_SSF_LOC :
          IGeoPoint currentlocation = (IGeoPoint) msg.obj;
          // ...Implementierung...
          break;
        case MSG_LOG :
          // ...Implementierung...
          break;
        default :
          break;
      }
    }
  };

  private iLocationListener iLocL = new iLocationListener() {
    @Override
    public void onLocationChanged(IGeoPoint location, Accuracy acc)
        {
      handler.sendMessage(handler
          .obtainMessage(MSG_SSF_LOC, location));
    }

    @Override
    public void onStateChanged(int state) {
    }
```

```
    @Override
    public void onLocationChanged(List<IGeoPoint> list, Accuracy acc
        ) {
    }

  };

  private LogListener logL = new LogListener() {

    @Override
    public void onLogMessage(String tag, String msg) {
      String logMsg = tag + "␣" + msg;
      handler.sendMessage(handler.obtainMessage(MSG_LOG, logMsg));
    }
  };

  private ServiceConnection ssfServiceConnection = new
     ServiceConnection() {
    @Override
    public void onServiceDisconnected(ComponentName name) {
      ssfKill();
      ssfBinder = null;
      ssf = null;
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder
       service) {
      ssfBinder = (SmartSpaceFramework.SSFBinder) service;
      ssf = ssfBinder.getSSF();
    }
  };

  /**
   * Lebenszyklus Methode onCreate(). Wird vom AAF beim starten
     einer Activity
   * aufgerufen.
   */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    // ...Implementierung...
    Intent ssfServiceIntent = new Intent(getApplicationContext(),
        SmartSpaceFramework.class);
    bindService(ssfServiceIntent, ssfServiceConnection,
        Context.BIND_AUTO_CREATE);
    ssfStart();
  }

  private void ssfStart() {
    try {
      ssf.registerLogListener(logL);
```

```
      ssf.start();
      locationMngr = ssf
          .getLocationManager(SmartSpaceFramework.
             INDOOR_POSITION_PROVIDER);
      locationMngr.registerListener(iLocL);
    } catch (Exception e) {
      showMessage(e.getMessage());
      Log.d(LOG_TAG, "Unexpected␣error␣-␣Here␣is␣what␣I␣know:␣"
          + e.getMessage());
    }
  }

  /**
   * Lebenszyklus Methode onPause(). Wird vom AAF aufgerufen wenn
      eine den Fokus
   * verliert
   */
  @Override
  protected void onPause() {
    // ...Implementierung...
    unbindService(ssfServiceConnection);
  }

  private void ssfKill() {
    locationMngr.unregisterListener(iLocL);
    ssf.unregisterLogListener(logL);
    ssf.kill();
    try {
      Thread.sleep(1000);
    } catch (InterruptedException e) {
      e.printStackTrace();
    }
  }

  public void useQRInputMethod() {
    Intent intent = new Intent(
        "de.ambisense.smartspace.android.QRInputMethod");
    startActivityForResult(intent, 0);
  }

  // ...Implementierung...
}
```

Listing D.2: Beispielhafte KML Datei zur Auszeichnung von POIs

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <name><![CDATA[Wifi coverage Tuebingen, Germany]]></name>
    <Style id="wifiOpenPOI">
      <IconStyle>
        <Icon>
```

```
          <href>http://www.ambisense.uni-tuebingen.de/de/smartspace/
             wifi-open.png</href>
        </Icon>
      </IconStyle>
    </Style>
    <Style id="wifiWepPOI">
      <IconStyle>
        <Icon>
          <href>http://www.ambisense.uni-tuebingen.de/de/smartspace/
             wifi-wep.png</href>
        </Icon>
      </IconStyle>
    </Style>
    <Style id="wifiWpaPOI">
      <IconStyle>
        <Icon>
          <href>http://www.ambisense.uni-tuebingen.de/de/smartspace/
             wifi-wpa.png</href>
        </Icon>
      </IconStyle>
    </Style>
    <Placemark>
    <name><![CDATA[SSID: Helios]]></name>
    <description><![CDATA[BSSID: <b>00:13:49:f3:c5:a1</b><br/>
       Capabilities: <b></b><br/>Frequency: <b>2437</b><br/>Level: <
       b>-69</b><br/>Timestamp: <span>1271604688634</span><br/>Date:
        <span>Apr 18, 2010 5:31:28 PM</span>]]></description><
       styleUrl>#wifiOpenPOI</styleUrl>
    <Point>
    <coordinates>9.048678874969482,48.52129518985748,391.0</
       coordinates>
    </Point>
  </Placemark>
  <Placemark>
    <name><![CDATA[SSID: TOBI]]></name>
    <description><![CDATA[BSSID: <span>00:19:5b:dd:89:fe</span><br/>
       Capabilities: <span>[WEP]</span><br/>Frequency: <span>2427</
       span><br/>Level: <span>-76</span><br/>Timestamp: <span
       >1271604670729</span><br/>Date: <span>Apr 18, 2010 5:31:10 PM
       </span>]]></description><styleUrl>#wifiWepPOI</styleUrl>
    <Point>
    <coordinates>9.048528671264648,48.52178871631622,386.0</
       coordinates>
    </Point>
  </Placemark>
  <Placemark>
    <name><![CDATA[SSID: WLAN-001F3F924AA3]]></name>
    <description><![CDATA[BSSID: <span>00:1f:3f:92:4a:a3</span><br/>
       Capabilities: <span>[WPA-PSK-TKIP][WPA2-PSK-CCMP]</span><br/>
       Frequency: <span>2422</span><br/>Level: <span>-92</span><br/>
       Timestamp: <span>1271604665791</span><br/>Date: <span>Apr 18,
```

```
        2010 5:31:05 PM</span>]]></description><styleUrl>#wifiWpaPOI
       </styleUrl>
    <Point>
    <coordinates>9.0485018491745,48.52186918258667,385.0</
       coordinates>
    </Point>
  </Placemark>
  </Document>
</kml>
```

Literaturverzeichnis

[Adm10] Admob: *Admob mobile metrics report*. Website, March 2010. Available online at http://metrics.admob.com/2010/04/march-2010-mobile-metrics-report/; visited on May 10th 2010.

[Alla] Alliance, Open Handset: *Android Application Fundamentals*. Website. Available online at http://developer.android.com/guide/topics/fundamentals.html; visited on December 9th 2009.

[Allb] Alliance, Open Handset: *Android NDK*. Website. Available online at http://developer.android.com/sdk/ndk/1.6_r1/index.html; visited on December 9th 2009.

[Allc] Alliance, Open Handset: *Android SDK*. Website. Available online at http://developer.android.com/sdk/index.html; visited on December 9th 2009.

[Alld] Alliance, Open Handset: *Android Source Code*. Website. Available online at http://source.android.com/; visited on December 9th 2009.

[Alle] Alliance, Open Handset: *Android System Architektur*. Website. Available online at http://developer.android.com/reference/android/location/Geocoder.html; visited on December 9th 2009.

[Allf] Alliance, Open Handset: *Android System Architektur*. Website. Available online at http://developer.android.com/guide/basics/what-is-android.html; visited on December 9th 2009.

[Amba] Ambisense: *Ambiente Vernetzung mobiler Systeme*. Website. Available online at http://http://www.ambisense.uni-tuebingen.de/; visited on July 4th 2009.

[Ambb] Ambisense: *WLAN Trilateration with Goodtry*. Website. Available online at http://www.ambisense.uni-tuebingen.de/en/wlan-trilateration/; visited on July 4th 2009.

[Appa] Apple, Inc.: *iPhone Mapkit Framework*. Website. Available online at http://developer.apple.com/iphone/library/documentation/MapKit/Reference/MapKit_Framework_Reference/index.html; visited on February 8th 2010.

[Appb] Apple, Inc.: *iPhone OS*. Website. Available online at http://www.apple.com/de/iphone/softwareupdate/; visited on February 8th 2010.

[ARM] ARM, Ltd.: *ARM Processor Architecture*. Website. Available online at `http://infocenter.arm.com/help/index.jsp`; visited on December 9th 2009.

[Ass] Association, Infrared Data: *Infrared*. Website. Available online at `http://www.irda.org/`; visited on May 14th 2009.

[Bar] Baracoda, Inc.: *TagRunners Bluetooth RFID Scanner*. Website. Available online at `http://code.google.com/apis/documents/docs/3.0/developers_guide_protocol.html#OCR`; visited on February 19th 2010.

[BBG04] Bhasker, Ezekiel S., Steven W. Brown und William G. Griswold: *Employing User Feedback for Fast, Accurate, Low-Maintenance Geolocationing*. In: *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, Seite 111, Washington, DC, USA, 2004. IEEE Computer Society.

[BBP00] Bahl, P., A. Balachandran und V. Padmanabhan: *Enhancements to the RADAR User Location and Tracking System*, 2000.

[BBV02] Battiti, Roberto, Mauro Brunato und Alessandro Villani: *Statistical learning theory for location fingerprinting in wireless LANs*. Technischer Bericht, 2002.

[BCLN05] Borriello, Gaetano, Matthew Chalmers, Anthony LaMarca und Paddy Nixon: *Delivering real-world ubiquitous location systems*. Commun. ACM, 48(3):36–41, 2005.

[BHS07] Buschmann, Frank, Kevlin Henney und Douglas C. Schmidt: *Pattern-Oriented Software Architecture, Volume 4: A Pattern Language for Distributed Computing*. Wiley, Chichester, UK, 2007.

[bke] bkennish@chromium.org: *QR-ome Chrome Extension*. Website. Available online at `https://chrome.google.com/extensions/detail/nkhgpliefpgnkmoeidcfchbbkmnpbmeh`; visited on April 14th 2010.

[Bla] Blau, Samsung Mobilers Team: *Android Conference App*. Website. Available online at `http://www.theconferenceapp.com/start/de/index`; visited on February 15th 2010.

[Bor] Bornstein, Dan: *Dalvik Virtual Machine*. Website. Available online at `http://www.dalvikvm.com/`; visited on December 9th 2009.

[BP00] Bahl, Paramvir und Venkata N. Padmanabhan: *RADAR: An In-Building RF-based User Location and Tracking System*. Microsoft Research, Seite 10, 2000.

[BPCL09] Bolliger, Philipp, Kurt Partridge, Maurice Chu und Marc Langheinrich: *Improving Location Fingerprinting through Motion Detection and Asynchronous Interval Labeling*. In: Quigley, Aaron und Tanzeem Choudhury (Herausgeber): *Fourth International Symposium on Location- and Context-Awareness (LoCA 2009), 7-8 May 2009, Tokyo, Japan*, LNCS, Berlin Heidelberg New York, Mai 2009. Springer.

[Buna] Bundesamt, Statistisches: *Erhebung über private Nutzung von Informations- und Kommunikations-technologien*. Website. Available online at http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Content/Publikationen/Qualitaetsberichte/WirtschaftsrechnungenZeitbudget/IKT2009,property=file.pdf; visited on February 19th 2010.

[Bunb] Bundesamt, Statistisches: *Erhebung über private Nutzung von Informations- und Kommunikations-technologien*. Website. Available online at http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2008/04/PD08__163__52911,templateId=renderPrint.psml; visited on February 19th 2010.

[Cam] Cambridge, AT&T Laboratories: *The Bat Ultrasonic Location System*. Website. Available online at http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/; visited on July 14th 2009.

[CCKM01] Castro, Paul, Patrick Chiu, Ted Kremenek und Richard R. Muntz: *A Probabilistic Room Location Service for Wireless Networked Environments*. In: *UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing*, Seiten 18–34, London, UK, 2001. Springer-Verlag.

[CGL93] Christ, T. W., P. A. Godwin und R. E. Lavigne: *A prison guard Duress alarm location system*. Seiten 106–116, 1993.

[Cona] Consortium, Open Geospatial: *Geography Markup Language (GML)*. Website. Available online at http://www.opengeospatial.org/standards/gml#overview; visited on March 10th 2010.

[Conb] Consortium, SQLite: *SQLite*. Website. Available online at http://sqlite.org/; visited on December 9th 2009.

[Cora] Corp., Intel: *Moblin Platform*. Website. Available online at http://moblin.org/; visited on February 8th 2010.

[Corb] Corp., Nokia: *Maemo OS*. Website. Available online at http://maemo.org/; visited on February 8th 2010.

[CWKS02] Crow, B. P., I. Widjaja, L. G. Kim und P. T. Sakai: *IEEE 802.11 Wireless Local Area Networks*. Communications Magazine, IEEE, 35(9):116–126, August 2002.

[DAS01] Dey, Anind K., Gregory D. Abowd und Daniel Salber: *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*. Human-Computer Interaction, 16(2):97–166, 2001.

[DoCSUtUoC] California, Los Angeles Department of Computer Science UCLA the University of: *The Nibble Location System*. Website. Available online at http://mmsl.cs.ucla.edu/nibble/; visited on May 2001.

[DW] Denso-Wave, Inc.: *QR-Code Format*. Website. Available online at http://www.denso-wave.com/qrcode/aboutqr-e.html; visited on February 17th 2010.

[DZ02] Dornbusch, Peter und Maximilian Zuendt: *Realisierung von Positionsortungen in WLAN*. In: *ITG-Fachtagung Technologie und Anwendungen für die mobile Informationsgesellschaft Dresden*. VDE Verlag, 2002.

[Ekaa] Ekahau: *HeatMapper - The Free Wi-Fi Coverage Mapping Site Survey Tool*. Website. Available online at http://www.ekahau.com/products/heatmapper/features-a-screen-shots.html; visited on April 12th 2010.

[Ekab] Ekahau: *WLAN Real Time Location System*. Website. Available online at http://www.ekahau.com/; visited on July 9th 2009.

[ESKF05] Ekiz, Nasıf, Tara Salih, Sibel Küçüköner und Kemal Fidanboylu: *An Overview of Handoff Techniques in Cellular Networks*, 2005.

[FBSR08] Filho, Julio Oliveira, Ana Bunoza, Jürgen Sommer und Wolfgang Rosenstiel: *Self-Localization in a Low Cost Bluetooth Environment*. In: *UIC '08: Proceedings of the 5th international conference on Ubiquitous Intelligence and Computing*, Seiten 258–270, Berlin, Heidelberg, 2008. Springer-Verlag.

[FKZL03] Feldmann, Silke, Kyandoghere Kyamakya, Ana Zapater und Zighuo Lue: *An Indoor Bluetooth-Based Positioning System: Concept, Implementation and Experimental Evaluation*. Seiten 109–113, 2003.

[Fou] Foundation, Apache: *Apache Harmony*. Website. Available online at http://harmony.apache.org/; visited on December 9th 2009.

[Gar] Gartner: *10 Mobile Technologies to Watch in 2010 and 2011*. Website. Available online at http://www.gartner.com/it/page.jsp?id=1328113; visited on November 10th 2009.

[Geo] Geonames: *Reverse Geocoding Webservices*. Website. Available online at http://www.geonames.org/export/reverse-geocoding.html; visited on July 15th 2009.

[GHJV95] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[Gooa] Google, Inc.: *Google Charttools API*. Website. Available online at http://code.google.com/intl/de-DE/apis/charttools/; visited on February 19th 2010.

[Goob] Google, Inc.: *Google Docs OCR-API*. Website. Available online at http://code.google.com/apis/documents/docs/3.0/developers_guide_protocol.html#OCR; visited on February 19th 2010.

[Gra] Grassmuck, Volker: *Die japanische Schrift und ihre Digitalisierung*. Website. Available online at `http://waste.informatik.hu-berlin.de/Grassmuck/Texts/jp-schrift.html`; visited on February 17th 2010.

[Guo07] Guochang, Xu: *GPS: Theory, Algorithms and Applications*. Springer-Verlag, Berlin, Federal Republic of Germany(DEU), 2007.

[HB01] Hightower, Jeffrey und Gaetano Borriello: *A Survey and Taxonomy of Location Systems for Ubiquitous Computing*. Technischer Bericht, IEEE Computer, 2001.

[HBB02] Hightower, Jeffrey, Barry Brumitt und Gaetano Borriello: *The Location Stack: A Layered Model for Location in Ubiquitous Computing*. Mobile Computing Systems and Applications, IEEE Workshop on, 0:22, 2002.

[Her06] Hermersdorf, Marion: *Indoor Positioning with a WLAN Access Point List on a Mobile Device*. Proceedings of Workshop on World-Sensor-Web at the 4th ACM Conference on Embedded Networked Sensor Systems (WSW'06 at Sensys'06), oct 2006.

[HHS+99] Harter, Andy, Andy Hopper, Pete Steggles, Andy Ward und Paul Webster: *The anatomy of a context-aware application*. In: *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, Seiten 59–68, New York, NY, USA, 1999. ACM.

[HW08] Hoene, Christian und Jörg Willmann: *Four-way TOA and Software-Based Trilateration of IEEE 802.11 Devices*. In: *IEEE PIMRC*, Cannes, September 2008.

[IETa] IETF: *A Uniform Resource Identifier for Geographic Locations (geo URI) draft-mayrhofer-geopriv-geo-uri-01*. Website. Available online at `http://tools.ietf.org/html/draft-mayrhofer-geopriv-geo-uri-01`; visited on May 3th 2010.

[IETb] IETF: *Uniform Ressource Identifier RFC3986*. Website. Available online at `http://tools.ietf.org/html/rfc3986`; visited on February 17th 2010.

[IETc] IETF: *VCard*. Website. Available online at `http://datatracker.ietf.org/wg/vcarddav/`; visited on May 14th 2009.

[Inca] Inc., Google: *Google data protocol (GData) - Google Maps Data API*. Website. Available online at `http://code.google.com/intl/de-DE/apis/maps/documentation/mapsdata/developers_guide_protocol.html`; visited on April 13th 2010.

[Incb] Inc., Google: *Google Maps Navigation*. Website. Available online at `http://www.google.com/mobile/navigation/`; visited on April 13th 2010.

[Incc] Inc., Yahoo: *Yahoo Geocoding API!* Website. Available online at `http://developer.yahoo.com/maps/rest/V1/geocode.html`; visited on April 13th 2010.

[Jef] Jeffrey: *SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength*.

[Kae05] Kaemarungsi, Kamol: *Design of indoor positioning systems based on location fingerprinting technique*. Doktorarbeit, Pittsburgh, PA, USA, 2005. Adviser-Krishnamurthy, Prashant.

[KK03] Kourogi, Masakatsu und Takeshi Kurata: *Personal Positioning based on Walking Locomotion Analysis with Self-Contained Sensors and a Wearable Camera*. In: *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, Seite 103, Washington, DC, USA, 2003. IEEE Computer Society.

[KK04] Kaemarungsi, Kamol und Prashant Krishnamurthy: *Properties of Indoor Received Signal Strength for WLAN Location Fingerprinting*. Mobile and Ubiquitous Systems, Annual International Conference on, 0:14–23, 2004.

[KKH+06] King, Thomas, Stephan Kopf, Thomas Haenselmann, Christian Lubberger und Wolfgang Effelsberg: *COMPASS: A probabilistic indoor positioning system based on 802.11 and digital compasses*. In: *WiNTECH '06: Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization*, Seiten 34–40, New York, NY, USA, 2006. ACM.

[kowa] kowoma: *GPS-Monitor*. Website. Available online at `http://www.kowoma.de/gps/gpsmonitor/gpsmonitor.php`; visited on May 13th 2010.

[kowb] kowoma: *Umlaufbahn der GPS Satelliten*. Website. Available online at `http://www.kowoma.de/gps/Umlaufbahnen.htm`; visited on May 14th 2010.

[LBR+02] Ladd, Andrew M., Kostas E. Bekris, Algis Rudys, Guillaume Marceau, Lydia E. Kavraki und Dan S. Wallach: *Robotics-based location sensing using wireless ethernet*. In: *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, Seiten 227–238, New York, NY, USA, 2002. ACM.

[LBR+05] Ladd, Andrew M., Kostas E. Berkis, Algis Rudys, Lydia E. Kavraki und Dan S. Wallach: *Robotics-based location sensing using wireless Ethernet*. Wirel. Netw., 11(1-2):189–204, 2005.

[M. 07] M. Hossain, H. Nguyen Van, Y. Jin and W.-S. Soh: *Indoor Localization using Multiple Wireless Technologies*. In: *Proc. IEEE MASS*, Pisa, Italy, Oct. 2007.

[Mah36] Mahalanobis, P. C.: *On the generalized distance in statistics*. Seiten 49–55, 1936.

[Mic] Microsystems, Sun: *JVM Specification*. Website. Available online at `http://java.sun.com/docs/books/jvms/`; visited on December 9th 2009.

[Mut09] Muthukrishnan, Kavitha: *Multimodal localisation : analysis, algorithms and experimental evaluation*. Doktorarbeit, Enschede, September 2009.

[NC] Nokia Corp., Intel Corp.: *Meego Platform*. Website. Available online at `http://meego.com/`; visited on February 8th 2010.

[Ope] OpenGL: *Open GL Project*. Website. Available online at `http://openGL.org`; visited on December 9th 2009.

[oS] Singapore, National University of: *Fragmentation of mobile applications*. Website. Available online at `http://www.comp.nus.edu.sg/~damithch/df/device-fragmentation.htm`; visited on March 24th 2010.

[OSI] OSI, Open Source Initiative: *Apache License, Version 2.0*. Website. Available online at `http://www.opensource.org/licenses/apache2.0.php`; visited on December 9th 2009.

[OVLL05] Otsason, Veljo, Alex Varshavsky, Anthony Lamarca und Eyal De Lara: *Accurate GSM Indoor Localization*. In: *In the proc. of UbiComp 2005*, Seiten 141–158, 2005.

[Pac04] Paciga, Mark: *Herecast: An Open Infrastructure for Location-Based Services using Wi-Fi*, 2004.

[Pal] Palm, Inc.: *Palm Web OS*. Website. Available online at `http://developer.palm.com/`; visited on February 8th 2010.

[PCB00] Priyantha, Nissanka B., Anit Chakraborty und Hari Balakrishnan: *The Cricket location-support system*. In: *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, Seiten 32–43, New York, NY, USA, 2000. ACM.

[PKB98] Pahlavan, K., P. Krishnamurthy und A. Beneat: *Wideband radio propagation modeling for indoor geolocation applications*. Communications Magazine, IEEE, 36(4):60–65, 1998.

[PKL07] Phillips, Shaun, Michael Katchabaw und Hanan Lutfiyya: *WLocator: An Indoor Positioning System*. In: *WIMOB '07: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, Seite 33, Washington, DC, USA, 2007. IEEE Computer Society.

[PLM02] Pahlavan, K., Xinrong Li und J. P. Makela: *Indoor geolocation science and technology*. Communications Magazine, IEEE, 40(2):112–118, 2002.

[PLY+00] Pahlavan, Kaveh, Xinrong Li, Mika Ylianttila, Ranvir Chana und Matti Latva-aho: *An Overview of Wireless Indoor Geolocation Techniques and Systems*. In: *NETWORKING '00: Proceedings of the IFIP-TC6/European Commission International Workshop on Mobile and Wireless Communication Networks*, Seiten 1–13, London, UK, 2000. Springer-Verlag.

[Rad] Radar, Oreilly: *Internet OS*. Website. Available online at http://radar.oreilly.com/2010/03/state-of-internet-operating-system.html; visited on March 29th 2010.

[RDM03] Randell, Cliff, Chris Djiallis und Henk Muller: *Personal Position Measurement Using Dead Reckoning*. In: *ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, Seite 166, Washington, DC, USA, 2003. IEEE Computer Society.

[Ret] Retrodev: *Dalvik Virtual Machine Bytecode Format*. Website. Available online at http://www.retrodev.com/android/dexformat.html; visited on December 9th 2009.

[Sal] Sales, Jane: *Location 101: breaking down the market for location-based apps*. Website. Available online at http://www.visionmobile.com/blog/2010/02/14/; visited on February 15th 2010.

[Sch03] Schiller, Jochen: *Mobilkommunikation : Techniken für das allgegenwärtige Internet*. Pearson Studium, München, 2. Auflage, 2003.

[SHTH07] Saito, Shigeru, Atsushi Hiyama, Tomohiro Tanikawa und Michitaka Hirose: *Indoor Marker-based Localization Using Coded Seamless Pattern for Interior Decoration*. In: *VR*, Seiten 67–74, 2007.

[SK08] Swangmuang, Nattapong und Prashant Krishnamurthy: *Location Fingerprint Analyses Toward Efficient Indoor Positioning*. Pervasive Computing and Communications, IEEE International Conference on, 0:100–109, 2008.

[SR00] Shyy, Dong-Jye und B. Rohani: *Indoor location technique for 2G and 3G cellular/PCS networks*. Local Computer Networks, Annual IEEE Conference on, 0:264, 2000.

[SSZ+09] Subramanian, Suguna P., Juergen Sommer, Frank-Peter Zeh, Stephen Schmitt und Wolfgang Rosenstiel: *PBIL PDR for scalable Bluetooth Indoor Localization*. In: *NGMAST '09: Proceedings of the 2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies*, Seiten 170–175, Washington, DC, USA, 2009. IEEE Computer Society.

[Staa] Stanely, Morgan: *Internet Trends*. Website. Available online at http://www.morganstanley.com/institutional/techresearch/pdfs/Internet_Trends_041210.pdf; visited on May 17th 2010.

[Stab] Stanely, Morgan: *Internet Trends*. Website. Available online at http://www.scribd.com/doc/21364028/Morgan-Stanley-Economy-Internet-Trends; visited on November 10th 2009.

[Ste] Stender, Michael Werner: *GSM 21 - SDMA + FDMA = Zellulare Netze*. Website. Available online at http://www.ip-mobil.de/GSM-Grundlagen/SDMA-%20FDMA/index.php; visited on March 15th 2010.

[Tra89] Traenkler, H.R.: *Taschenbuch der Meßtechnik, mit Schwerpunkt Sensortechnik*. Oldenbourg Verlag, 1989.

[Vid] Video, Packet: *Open Core Media Framework*. Website. Available online at `http://www.packetvideo.com/products/android/index.html`; visited on December 9th 2009.

[War99] Ward, Andrew Martin Robert: *Sensor-driven Computing*. Doktorarbeit, University of Cambridge, 1999.

[WBM02] Wirfs-Brock, Rebecca und Alan McKean: *Object Design: Roles, Responsibilities, and Collaborations*. Pearson Education, 2002. Foreword By-Jacobson, Ivar and Foreword By-Vlissides, John.

[Web] Webkit: *Webkit Browser*. Website. Available online at `http://webkit.org/`; visited on December 9th 2009.

[wgs04] *Department of Defense World Geodetic System 1984 – Its Definition and Relationships with Local Geodetic Systems*. Technischer Bericht, Bethesda, MD, 2004.

[WHFG92] Want, Roy, Andy Hopper, Veronica Falc ao und Jonathan Gibbons: *The active badge location system*. ACM Trans. Inf. Syst., 10(1):91–102, 1992.

[WJH97] Ward, Andy, Alan Jones und Andy Hopper: *A New Location Technique for the Active Office*. IEEE Personal Communications, 4(5):42–47, 1997.

[WL98] Werb, Jay und Colin Lanzl: *Designing a positioning systems for finding things and people indoors*. IEEE Spectr., 35(9):71–78, 1998.

[XSC+04] Xiang, Z., S. Song, J. Chen, H. Wang, J. Huang und X. Gao: *A wireless LAN-based indoor positioning technology*. IBM J. Res. Dev., 48(5/6):617–626, 2004.

[YA03] Youssef, Moustafa und Ashok Agrawala: *On the Optimality of WLAN Location Determination Systems*. In: *In Communication Networks and Distributed Systems Modeling and Simulation Conference*, Seiten 205–218, 2003.

[YA05] Youssef, Moustafa und Ashok Agrawala: *The Horus WLAN location determination system*. In: *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, Seiten 205–218, New York, NY, USA, 2005. ACM.

[YAS03] Youssef, Moustafa, Ashok Agrawala und A. Udaya Shankar: *WLAN Location Determination via Clustering and Probability Distributions*. In: *In IEEE PerCom 2003*, 2003.

[YASN02] Youssef, Moustafa, Ashok Agrawala, A. Udaya Shankar und Sam H. Noh: *A Probabilistic clustering-based indoor location determination system*. In: *Technical Report UMIACS-TR 2002-30 and CS-TR 4350*, 2002.

[YZN07] Yeung, Wilson M., JunYang Zhou und Joseph K. Ng: *Enhanced Fingerprint-Based Location Estimation System in Wireless LAN Environment*. In: *EUC Workshops*, Seiten 273–284, 2007.

[Zai] Zaielacademic: *Line of Sight*. Website. Available online at `http://zaielacademic.net/networking_wireless/wireless_antennas.htm`; visited on December 7th 2009.

[ZG04] Zhao, Feng und Leonidas Guibas: *Wireless Sensor Networks: An Information Processing Approach (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann, July 2004.

[ZXIa] ZXING: *A barcode scanner for the android platform*. Website. Available online at `http://code.google.com/p/zxing/wiki/BarcodeContents`; visited on February 17th 2010.

[ZXIb] ZXING: *QR-Code GEO Format*. Website. Available online at `http://code.google.com/p/zxing/wiki/BarcodeContents`; visited on February 17th 2010.

[ZYN08] Zhou, Junyang, Wilson Man-Chung Yeung und Joseph Kee-Yin Ng: *Enhancing Indoor Positioning Accuracy by Utilizing Signals from Both the Mobile Phone Network and the Wireless Local Area Network*. Advanced Information Networking and Applications, International Conference on, 0:138–145, 2008.